Multi-View Video Surveillance of Outdoor Traffic

P. T. Duizer & D. M. Hansen

Multi-View Video Surveillance of Outdoor Traffic Master Thesis — August 2007

Aalborg University — Laboratory of Computer Vision and Media Technology — 2007

AALBORG UNIVERSITY DENMARK

Faculty of Engineering, Science and Medicine

Aalborg University

Laboratory of Computer Vision and Media Technology

Title:
Multi-View Video Surveillance of Outdoor Traffic

Theme:
Computer Vision

Semester:
9th - 10th semester

Project period:
September 5th 2006 - August 16th 2007

Group members:
Paul Toft Duizer
Dennis Mølholm Hansen

Supervisors:
Thomas Moeslund
Mohan Trivedi
Sangho Park

Issues:	5
Main report:	125
Appendix:	53
Total:	178

Abstract

One of the goals for performing traffic monitoring is to avoid traffic accidents. It would not be feasible to use a human operator to monitor a traffic scene because accidents are rare events. There is a growing interest in automating this process, and visual surveillance systems are paid much attention due to their non-intrusive nature.

This thesis addresses the tracking issue, which is a cornerstone of all visual surveillance systems. The overall goal is to use the tracking information to detect potential traffic accidents before they occur. A requirement is thus that the system must be able to track both vehicles and humans reliably. There is only a limited amount of work reported on tracking of both vehicles and humans.

The developed system is a multi-view tracking system based on the planar homography. Foreground segmentation for each view is performed using the codebook method, which is capable of adapting to illumination changes. The tracking of objects is performed in each view using bounding box overlap, and occlusion situations are resolved by probabilistic appearance models. The following correspondence of tracks between views is carried out by combining and modifying prominent methods for humans and vehicles. In the human case, the principal axis method is extended to handle groups. In the vehicle case, the footage region is applied, and special attention has been put on solving occlusion situations.

Due to the use of multiple views and the correspondence of tracks it is possible to calculate an accurate view invariant representation of the objects. This representation is suitable for performing event recognition and assessment of the danger level of the situation. The goal of this is to detect an accident before it occurs, and an alarm is raised as a first step of preventing the accident.

The developed system is tested over several hours of unconstrained data on different times of the day, under different illuminations and different camera configurations. The system gives a solid foundation for tracking objects, and demonstrations using analysis based on the view invariant representation of objects show that the system is able to detect dangerous situations, e.g. near collisions between vehicles and humans.

Preface

This thesis documents the work performed by Paul Toft Duizer and Dennis Mølholm Hansen for the degree of M.Sc.EE during the 9th and 10th semester at the specialization of Computer Vision and Graphics at Aalborg University, Denmark. The research for this thesis was mainly carried out at the Computer Vision and Robotics Research laboratory (CVRR lab) at University of California, San Diego (UCSD) in USA. The study abroad lasted seven months from November 8^{th} 2006 to June 10^{th} 2007. From June 12^{th} 2007 to August 16^{th} 2007 the research was completed at Aalborg University.

The thesis covers both 9^{th} and 10^{th} semester, but it was not until the middle of January 2007 that the subject of the thesis was established in co-operation with the CVRR lab. The final thesis definition was established in the middle of March 2007.

In the two months period from arriving at the CVRR lab to establishment of the thesis subject we worked on a different subject. We worked on a continuation of our 8^{th} semester project where the subject was to perform automatic annotation of humans in indoor surveillance video recordings. The developed system was extended by new methods, existing methods was refined and new tests of the system was performed. As a part of the work a dataset was developed, which is now available for download at `http://www.cvmt.dk/projects/Hermes/head-data.html`. Our work resulted in the publication "Automatic Annotation of Humans in Surveillance Video", which was submitted and accepted for oral presentation at The First International Workshop on Video Processing and Recognition (VideoRec'07). The publication is enclosed as Appendix G.

Reading Guidelines

Literature references are in the thesis indicated with brackets, e.g. [Hu et al., 2006]. The position of the reference determines which specific part the reference relates to. If the reference is positioned before a period, it relates to the previous sentence. If the reference is positioned after a period, it relates to the previous section. The bibliography with references to all the literature used can be found at the end of the thesis.

Equations and formulas use the notation $\mathbf{a}$ for vectors and $\mathbf{A}$ for matrices. All equations are supplied with a number in the right margin for reference.

Videos, test results, source code and other project related files are located on the attached DVD. A reference to a file on the DVD is given as e.g. (◎ */application/config_ decription.doc*). A reference to a folder on the DVD is given as e.g. (◎ */application/*).

Paul Toft Duizer

Dennis Mølholm Hansen

Acknowledgements

We would like to thank to the following foundations for financial support:

Aalborg Universitets Internationale Puljer

Fabrikant Vilhelm Pedersen og Hustrus Legat

Henry og Mary Skovs Fond

Ingeniørforeningen i Danmarks Låne og Hjælpefond

Jørgen Esmers Mindelegat

Knud Højgaards Fond

Reinholdt W. Jorck og Hustrus Fond

Nordea Danmark Fonden

Otto Mønsteds Fond

Ingeniør Alexandre Haynman og hustru Nina Haynmans Fond

Julie Damms Studiefond

Brandkassefonden Sydvest

The work could not have been made without the help, guidance and support from the following group of people, which we would like to send a special thanks:

Professor Mohan Trivedi for accepting us at the CVRR lab and always being enthusiastic, positive and inspiring as our supervisor abroad.

Dr. Thomas Moeslund for initiating the contact with the CVRR lab and making our stay possible and for providing critical supervision and inspiration.

Dr. Sangho Park for being an enthusiastic and inspiring supervisor helping to understand and overcome challenges.

The people at CVRR lab for helping us out whenever we needed it and for a good time.

We also give personal thanks to our families for being supportive, visiting and helping with various practical issues.

Contents

Chapter 1

Introduction

With an increasing amount of vehicular traffic the number of traffic related accidents is expected to increase unless measures are taken to enhance traffic safety. Traffic related deaths are currently the fourth most occurring reason for reduction in years of life in Denmark [Færdselssikkerhedskommisionen, 2000]. In [Færdselssikkerhedskommisionen, 2000] the national Danish goals for traffic safety are defined for the period of 2001 - 2012. Based on the accident rate for 1998, the overall goal is a 40 percent reduction of traffic related deaths and injuries. To achieve this goal, a strengthening of the research within traffic safety is essential. One of the research themes with highest priority is the applicability of intelligent technology to increase traffic safety [Færdselssikkerhedskommisionen, 2007].

The use of cameras and computer vision to monitor a traffic area also referred to as visual surveillance can be used to detect illegal acts and detect accidents before they happen and take needed measures to prevent the accident. A visual surveillance system has several advantages. The cost of the cameras is relatively low and with only little disruption of the traffic during installation and maintenance [Kastrinaki et al., 2003]. Furthermore, cameras are a non-intrusive measure to monitor the traffic activities.

A single camera sensor has the ability to cover a large area, but covering all the traffic e.g. in an entire city would require many cameras. Applying only passive visual surveillance would involve many human operators to monitor the traffic. This is a very tedious task and an inefficient solution. The aim of visual surveillance using computer vision is to accomplish the entire surveillance task as automatically as possible. Given that the visual surveillance system is operational 24 hours a day, a large amount of data describing the traffic is generated. This data can be used to obtain new knowledge about traffic safety, but could also be used in a non-safety related situation, e.g. using the data to do traffic flow analysis for traffic planning. Furthermore, the stored data and especially the video could be used by the police in crime related cases.

The use of computer vision as an intelligent technology to help increase the traffic safety is the goal of this thesis. The initial problem in this thesis can therefore be formulated as follows:

> ***How can computer vision-based visual surveillance be applied in order to increase traffic safety?***

Chapter 2

Problem Analysis

This problem analysis investigates the problem area of traffic surveillance and the use of a computer vision system to monitor the traffic. The analysis introduces the concept of situation awareness and a general framework for visual surveillance systems. Computer vision challenges and the related work are addressed afterwards. This results in the thesis definition.

2.1 Situation Awareness

Before describing the problem area the term *situation awareness* is introduced. Other work may use a different term such as "intelligent" or "smart-space" for basically the same concept. A simple explanation of situation awareness is "knowing what is going on around you". The research within this area studies how humans perceive, structure and use knowledge from the immediate environment. The most established definition is given by [Endsley, 1988], where situational awareness is divided into three levels. Level 1 is to perceive the elements in the environment. In level 2 the knowledge of these elements is used to create a holistic picture to understand the significance of objects and events. Level 3 is using this understanding to do a projection of future scenarios, which makes it possible to take a certain preventive action.

This is a general and abstract definition. An example of situational awareness taken from everyday life is when a pedestrian is crossing the street. First of all, the person about to cross the street has to locate the road he is about to cross and any potential cars on the road. By using his knowledge about the cars (how fast they drive and their location on the street) he can decide whether or not to cross the street. He utilizes situation awareness by stopping and waiting for the cars before crossing. A computer vision system for visual surveillance should be able to perform similar reasoning, if it should be used for increasing traffic safety.

This requires that the computer vision system is able to identify the objects and know what they represent. Level 3 situation awareness is useful when trying to detect (and following prevent) a car accident before it happens, but this is not achievable without the intermediate steps of level 1 and 2.

2.2 Description of Problem Area

The problem area is given in locations with both vehicles and humans. Interesting locations would typically be urban areas that are busy at some point during the day. According to [Færdselssikkerhedskommisionen, 2000], the risk of a traffic accident in urban areas is three

times larger than in areas outside a city. Furthermore, 80 percent of all accidents involving pedestrians or cyclists happen in urban areas. More specific urban location could be roads with sidewalks, parking lots, intersections and bus stops. In Figure 2.1 some examples of possible sites are given.

FIGURE 2.1: Examples of locations. Left: San Diego, USA. Middle: Reading, England. Right: Barcelona, Spain.

Two overall object types exist within the problem area: humans and vehicles. Humans can further be divided into pedestrians, bicyclists and humans riding a skateboard or roller skates. Vehicles can similarly be divided into sedans, pickups, trucks and busses. More simply this could be small and large vehicles. Other object types exists, such as animals and inanimate objects like a baby carriage, but the focus is kept on humans and vehicles because they appear more frequently and are more interesting within the given problem area.

In a location as described above, the pedestrians are most likely in transit. They may stop for a red light, stop and wait for the bus or stop before crossing the street to get an overlook of the situation. This does not necessarily mean that they follow a fixed movement pattern. They may meet and stop to talk for a longer period of time. They may cross the road at any time or simply turn around and go back. However, it is expected that pedestrians are mainly walking along the sidewalk. Bicyclists and persons on skateboard or roller skates may also be on the sidewalk or close to it. Pedestrians, cyclist and skateboarders interact differently, but in the following the interaction of human objects is focused on the interaction between pedestrians. An example of interaction between pedestrians is when they walk as a group. Two pedestrians may meet and continue as a group or split after walking as a group. Pedestrians may also interact in a more detailed level, e.g. holding hands or fighting.

Vehicles are like pedestrians in transit and are expected to be on the road, but may cross the sidewalk in order to get to a parking space or a driveway. However, the structure of the environment may restrict the movement to only a few paths (assuming drivers of the vehicles obey basic traffic laws). Vehicles interact differently than humans; e.g. they do not form groups like pedestrians, but typically drive one by one in a line. Exceptions from this is e.g. overtaking vehicles. All in all, the movement pattern of vehicles is not as arbitrary and complicated as that of pedestrians.

Interaction between vehicles and pedestrians are also possible. A car may pick up a pedestrian or a bus may stop to drop off several passengers. These kinds of interactions happens when the vehicle in question is standing still. When vehicles are moving, the interaction is more of a preventive character, e.g. a pedestrian stopping and waiting for a car to pass. Of course, the vehicle may also stop allowing people to cross the street. A last example of person-vehicle

interaction is the traffic accident, where the vehicle hits the human. If the goal is to prevent such accidents, the visual surveillance system should be able to exhibit situation awareness by projecting the scenario before it happens. In order to preserve the integrity of the visual surveillance system, the system must be able to distinguish between the different types of person-vehicle interactions and only raise an alarm in an unsafe situation. When an alarm is raised adequate measures must be taken to avoid the accident, but since the focus is on computer vision-based surveillance this is not included in the thesis.

2.3 General Framework of Visual Surveillance Systems

In order to take a deeper look into the structure of a visual surveillance system and the challenges of doing visual surveillance, it is advantageous to have a general framework as a starting point. Several general frameworks are presented in the literature, e.g. in [Hu et al., 2004b], [Moeslund et al., 2006] and [Valera and Velastin, 2005]. The framework presented by [Hu et al., 2004b] is the most extensive of the aforementioned in the sense that it applies to both single view and multi-view systems and covers both vehicles and humans. Furthermore, this survey is extensively used in the reviewed literature. Hence, the framework in [Hu et al., 2004b] is presented in the following and is the main inspiration for the system structure used in the thesis.

The main focus of the survey given in [Hu et al., 2004b] is object motion and object behavior. In the survey, a general framework of visual surveillance in dynamic scenes is presented. This framework is shown in Figure 2.2.

The framework consists of different stages covering low-level vision, intermediate-level vision and high-level vision. A visual surveillance application may not contain all the stages in the framework. Likewise, stages may be added in an application. The stages of the framework are described in the following.

Environment Modelling The process of detecting motion or segmenting foreground objects involves environment modelling, motion segmentation and object classification. To be able to separate background and foreground, a model for the background or environment is needed. The model makes it possible to determine, if a pixel or region in the frame corresponds to background or a foreground object.

Motion segmentation Motion segmentation detects pixels or regions corresponding to foreground objects using the environment or background model. These pixels or regions are the focus for the later stages in the framework. Most segmentation methods make use of either spatial or temporal information in the video sequence.

Object classification The detected regions from the motion segmentation stage may correspond to different targets in the scene. In traffic surveillance applications targets may be humans and vehicles as described in Section 2.2. Hence, it is essential to classify the detected regions. This is often in the literature considered as a standard pattern recognition task.

Tracking Having segmented the foreground objects, visual surveillance systems track these objects from frame to frame. Tracking foreground objects over time typically involves matching objects in consecutive frames using descriptions of e.g. points, lines or blobs.

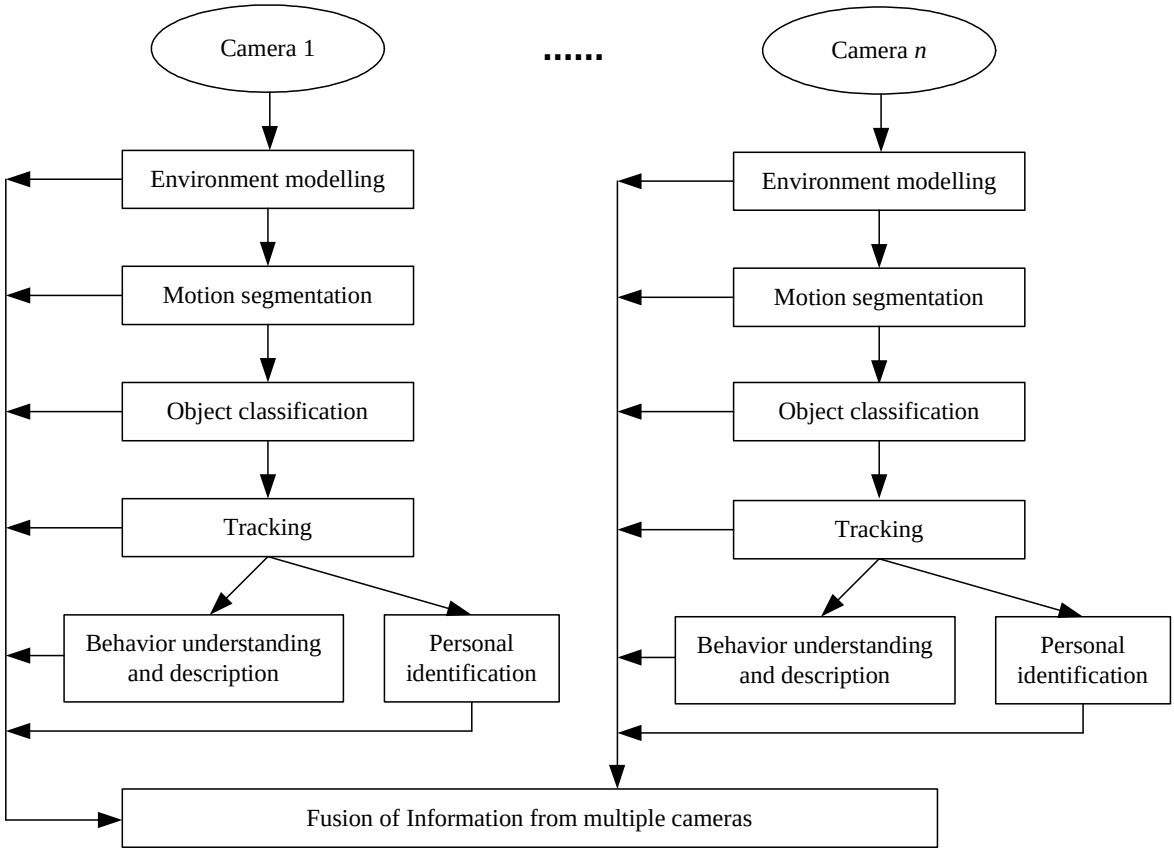


FIGURE 2.2: General Framework of Visual Surveillance Systems. Figure from [Hu et al., 2004b].

The main problem related to the tracking stage is how to handle occlusion. By tracking foreground objects the system is able to supply information on the history of the objects in the video sequence.

Behavior understanding and description Following tracking is the problem of understanding object behavior. In this stage, analysis of motion patterns is performed to produce high-level description of actions and interactions. The object history from the tracking stage can be utilized in the interpretation of the object motion patterns. Understanding of behavior can simply be seen as classification of time varying data. Typically, the behavior is classified as normal or abnormal.

Personal identification This stage is at the same level as behavior understanding and description. In this stage, a known identity is assigned to the tracked object. In case of surveillance of humans, biometric descriptions of e.g. the human face and gait can be used. A different approach is needed when assigning identities to vehicles. Given that the vehicles have a license plate and it is possible to extract an image of the license plates, it is rather straightforward to assign an identity to vehicles.

Fusion of Information from multiple cameras Segmenting foreground objects, object classification, tracking, behavior understanding and personal identification can be accomplished using a single camera. However, using multiple cameras can overcome problems regarding occlusion and depth estimation of the objects. In this stage, the information from the previous stages is collected. The information from the later stages is

at a higher abstraction level and is well suited for information fusion. However, fusion of lower level information may result in higher reliability of the overall system. [Hu et al., 2004b]

2.4 Computer Vision Challenges

Knowing the general structure of a visual surveillance system it is now possible to focus on the challenges related to visual surveillance. The goal for a traffic monitoring visual surveillance system is to be functional without any constraints on the given scene. To achieve this goal in the described problem area, several challenges must be addressed. Dealing with unconstrained environments involves the issues of changing illumination and handling multiple types of objects which interact in multiple different ways causing both partial and fully occluded objects. All together it becomes a highly complex task to interpret and keep track of information in an unconstrained environment. The following lists the main challenges in building a computer vision system for outdoor traffic and pedestrians in an unconstrained environment.

- Camera setup
- Changes in the scene
- Multiple types of moving objects
- Occlusion
- Fusion of information
- Understanding the behavior

In the following, these issues are elaborated.

2.4.1 Camera Setup

The camera setup is very application dependent and at the same time it provides a possibility to solve problems at an early stage. A camera can be stationary or moving, e.g. a PTZ camera. Furthermore, the camera can be mounted on buildings, but work has also been reported on cameras installed inside or outside the vehicle. An example is found in [Trivedi et al., 2005], where a system capable of looking out of the vehicle and analyzing the vehicle's surrounding is presented. Simultaneously, cameras are facing the driver inside the vehicle making it possible to monitor the driver's behavior and intent. However, in the following only stationary cameras mounted on buildings or lamp posts are considered. In Figure 2.3 three different camera setups are shown. The side view setup allows a good view of the persons making it attractive for recognition of detailed human interaction, but an object moving close to the camera would result in complete occlusion of the other objects. Compared to side view, the elevated setup reduces the effect of occlusion while having a good view of the pedestrians' and vehicles' appearance and at the same time gives a larger monitoring area. A camera would typically be elevated to more than 2.5 meters in this kind of setup in order to be out of reach from humans [Andersen et al., 2006]. By using the top view setup the occlusion problem would be close to eliminated. However, the appearance of the pedestrians is lost and installation of a

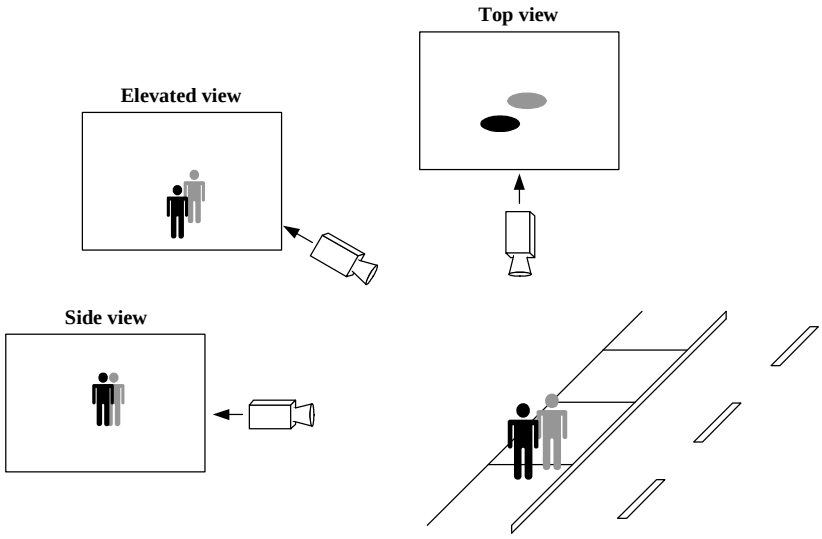


FIGURE 2.3: Possible camera setups. The difference is exemplified by two pedestrians walking in close proximity.

top view camera might be impractical. For these reasons, most existing surveillance systems use elevated cameras or side view cameras.

A second consideration closely related to the camera setup is the number of cameras used. A wider monitoring area can be covered by multiple cameras, and multiple cameras have proven efficient in handling occlusion situations. A bus can cause complete occlusion of several pedestrians in one view, but from another view all the pedestrians are visible.

Using many cameras with overlapping field of views increases the cost of the system and could be impractical to install. It should be assumed, only a few cameras (2-3 cameras) are overlapping. Furthermore, because the computer vision system must be able to handle multiple vehicles and pedestrians over a large monitoring area, a wide view seems more practical than a narrow view. When using multiple cameras it should be considered how the cameras are registered to make collaboration between views possible. If the camera registration can be performed automatically and still yield reliable results the applicability of the system would increase significantly.

Other issues with using multiple cameras is how to match the objects between views, switch between views as the object is leaving one view and how to fuse the data from multiple views. Despite these issues, the advantages of using multiple cameras to cover a larger area and resolve occlusion issues weigh more heavily.

2.4.2 Changes in the Scene

An outdoor surveillance system must handle significant changes in illumination. During daytime the most significant light source is the sun. The illumination of the scene can change drastically within a short period when a cloud passes the sun. Gradual and relatively slow changes in the illumination also occur. Around noon the sun is very bright and causes strong shadows. In the morning and evening the sun is not as bright, which results in longer, but weaker, shadows. The gradual change and corresponding cast shadow is illustrated in Fig-

ure 2.4. Artificial lights are a different kind of light sources, which causes different local illumination. At nighttime artificial light is typically the only significant light source.

(a) 9:10 (b) 13:10 (c) 16:07 (d) 17:53

FIGURE 2.4: The change in illumination over a nine hour period.

The background in the monitored scene is composed of static and non-static background objects. Static background objects could be building, lamp posts and roads. Typical examples of non-static background objects are vegetation, where trees and shrubberies are very likely in the given problem area. Vegetation presents a bigger challenge because they move if it is windy. The motion of vegetation could be confused with a moving region for a pedestrian or a vehicle. Other examples of non-static background objects are advertisements display with changing ads and objects deposited by humans. The latter could be luggage or a car being parked and left by the driver. A visual surveillance system must not identify static or non-static background objects as humans and vehicles.

2.4.3 Multiple Types of Moving Objects

In computer vision it is common to apply models when analyzing objects. In the traffic monitoring case the fact that humans and vehicles are different objects must be taken into consideration. The apparent shape of the objects viewed from a distance is an example of the difference. The human shape is almost the same even though the human object is viewed from the side or from the front. The shape of a vehicle is significantly different compared to the human shape, and the vehicle shape when viewed from the side and from the back differs significantly. Furthermore, vehicles are rigid objects whereas humans are non-rigid objects. This indicates that the models which apply to humans might not hold in the vehicle case.

2.4.4 Occlusion

The occlusion issue is closely related to interaction, since objects often occlude each other during interaction. Occlusion can either be partial or complete, e.g. pedestrians walking in a group often partly occlude each other, while a truck may fully occlude multiple people. Another kind of occlusion is background occlusion, meaning that a static object like a lamp post can cause occlusion if it is placed between the object of interest and the camera. At a more detailed level self occlusion can occur, e.g. when the arm of a person occludes a part of the body. Self occlusion is only an issue if detailed description of body interaction is needed. As mentioned earlier, using multiple cameras can be used in resolving these issues by providing a view where there is little or maybe no occlusion. If the visual surveillance system must be able to obtain situational awareness, the tracking of the objects must not be affected by occlusion.

A problem related to occlusion is track initialization. In the described problem area, it is very likely that pedestrians enter as group and stay as a group throughout the monitored area. This complicates the task of creating tracks for each individual in the group. Many existing systems require manual initialization to correctly track a group of persons. E.g. [Kim and Davis, 2006] and [Qu et al., 2007] explicitly write that tracks are created manually in such a situation.

2.4.5 Understanding the Behavior

In the general framework presented in Section 2.3 behavior understanding and personal identification is on the same level. However, when monitoring traffic the identity of the objects might not be of interest or must be hidden. The identity might only be of interest when analyzing a certain event post mortem, e.g. identify the car in a hit-and-run accident. Behavior understanding is of greater interest in relation to accident prevention. Often the goal is to detect normal or abnormal behavior or to project the states of the objects to detect near collisions. In case of a dangerous situation an alarm should be triggered. This requires some model or representation of the behavior. Furthermore, the behavior is exhibited over a time period between objects and the model or representation must therefore incorporate this temporal relationship.

In a wide camera surveillance system it is most likely only possible to extract the trajectories of the objects. The trajectories then forms the basis for the assessment of the danger level of the situation. The representation of the trajectories should also be considered. The representation can be view variant or view invariant. In a view variant representation a vehicle driving far away from the camera has smaller visual displacement in that view than a vehicle closer to the camera driving at the same speed. A view invariant representation does not have this disadvantage caused by the perspective effect. Therefore, a view invariant representation is the most general representation and well suited for doing analysis of interaction of objects, which is demonstrated in [Park and Trivedi, 2006].

It might be of interest to have an understanding of the objects' behavior on a finer level. A person about to cross the road without looking to either side of the road is a potential dangerous situation. This requires that it is possible to detect if the person has looked to both sides, which might be hard if the camera is located far away from the human object. However, given that it is possible to extract a finer description of the motion (e.g. using multiple cameras offering a good view of the object) this could be incorporated in the assessment of the danger level of the situation, thus reducing the number of falsely triggered alarms.

2.5 Partial Conclusion

In the previous, the problem area has been described, a general framework has been introduced and computer vision challenges have been identified. Because of the many challenges, the related work within visual surveillance is large, and to keep focus the scope must be delimited. It is chosen to focus on systems using stationary cameras mounted on e.g. buildings or lamp posts. Furthermore, systems using multiple views are addressed, because these systems provide a larger monitoring area and is also efficient in resolving occlusion issues.

2.6 Related Work

In this section the related work within visual surveillance of traffic is reviewed. A general framework of visual surveillance system is introduced in Section 2.3. The framework is divided into several stages, and reviewing the work related to each individual stage is a rather comprehensive task; the review of related work to the individual stages is carried out when describing the individual components of the system later in the thesis. The focus is in the following on complete systems and how they resolve some of the challenges in a traffic monitoring application. This is followed by a review of multi-view systems.

The interest in visual surveillance within the research community is large. This is evident by the IEEE workshops, IEEE conferences and special journal issues focusing solely on visual surveillance. Several surveys exists which offer a fine introduction to the subject. In this work, the surveys found in [Hu et al., 2004b], [Moeslund et al., 2006], [Kastrinaki et al., 2003] and [Valera and Velastin, 2005] have been widely used. The survey presented in [Moeslund et al., 2006] focuses on human motion capture and analysis and reviews the related work in the period from 2000 to 2006. Within this period more than 350 research papers have been published on this subject. [Kastrinaki et al., 2003] presents a survey on 128 publications of the research related to traffic applications with the focus on vehicles. The amount of work focusing on either monitoring of humans or monitoring of vehicles is large, but the amount of work focusing on monitoring of humans *and* vehicles simultaneously is relatively small.

The limited research done on monitoring both humans and vehicles is exemplified in the survey by [Valera and Velastin, 2005]. The survey describes the current state-of-the-art in the development of automated visual surveillance systems with the focus on complete systems developed within the academic research community or for commercial use. In this survey only four systems monitor both humans and vehicles. Only two of these systems use multiple views. Other systems reviewed in [Valera and Velastin, 2005] might also track humans and vehicles, but these systems do not assign an object type to the tracks, e.g. "human" and "vehicle". To enhance situational awareness this information is required in order to understand the significance of the objects and events (see Section 2.1). Hence, these systems are of little interest for this application. In the following review of work, the focus is on three visual surveillance systems which monitors both humans and vehicles and are able to distinguish humans from vehicles.

System by [Collins et al., 2000] One of the renowned visual surveillance systems is the Video Surveillance and Monitoring (VSAM) project. The project lasted three years, and the final report is presented in [Collins et al., 2000]. The purpose of the system was to develop automatic video understanding technologies that enable a single human operator to monitor behaviors over complex areas such as battlefields and civilian scenes. The VSAM system models the outdoor environment and segments moving objects from the background by combining adaptive background subtraction with a three-frame differencing technique. All moving objects are classified into four classes: single human, vehicle, human group and clutter. VSAM also performs finer classification of vehicles such as UPS truck, FedEx truck and police car. The moving objects are tracked using a combination of positional and template matching. VSAM performs gait analysis

of humans to distinguish running humans from walking humans. The system utilizes multiple cameras and calculates each object's geolocation using a terrain map. This makes it possible to make a 3D description of the objects, which can be passed from one camera to the next.

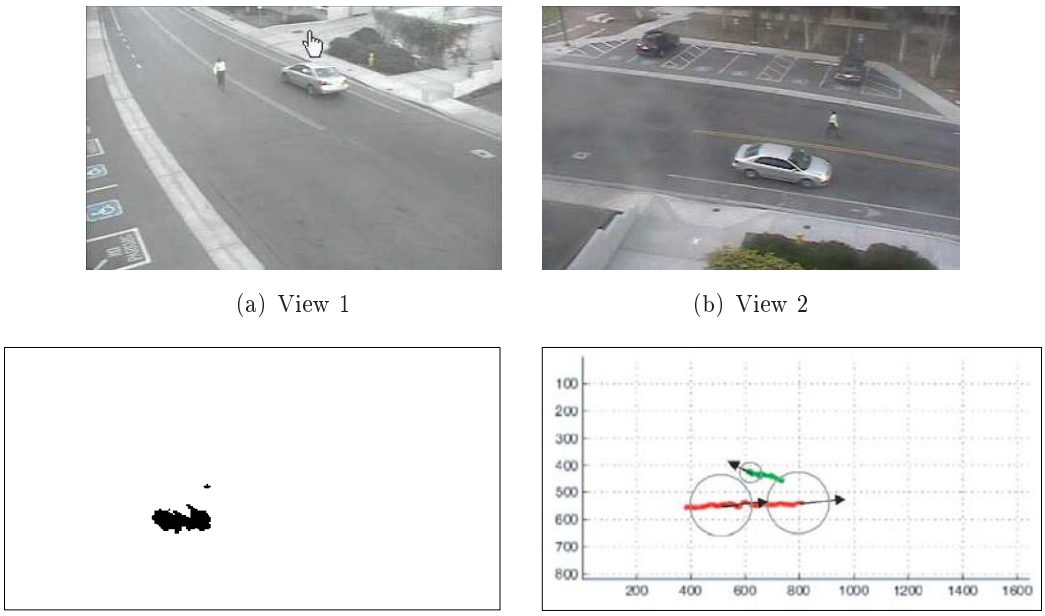
System by [Stauffer and Grimson, 2000] A visual surveillance system monitoring vehicles and humans is presented in [Stauffer and Grimson, 2000]. The goal of the system is to detect patterns of motion and interaction demonstrated by the objects in a site over long time intervals using a single camera. The objects are segmented by creating a model of the background using mixture of Gaussians. The tracking of objects uses a linearly predictive multiple hypotheses tracking algorithm which breaks tracks when objects interact. This is due to the following classification system requirement of tracking sequences containing only a single object. The classification system is based on a co-occurrence matrix to hierarchically classify both objects and behaviors. The system is able to classify traffic activities into directions in the site and also where in the site (e.g. on the road or on the sidewalk). Using the silhouette objects are furthermore classified as people, groups of people, cars or clutter. Preliminary work has been performed on detecting abnormal events.

System by [Park and Trivedi, 2006] The system presented in [Park and Trivedi, 2006] focuses on the interaction between persons and vehicles using multiple cameras. In [Park and Trivedi, 2006] moving objects are segmented from the background using moving window-based multi-frame differencing technique. The resulting foreground masks are transformed into a planar homography domain, where the moving objects are tracked. The planar homography constraint is exploited to extract view-invariant object features such as the footage region and velocity of objects on the ground plane. The system utilizes a spatial-temporal activity space based on the theory of personal space in social psychology in the analysis of the person-person interaction and person-vehicle interaction. An example of person-vehicle interaction using the view invariant representation of the objects is shown in Figure 2.5. The system can be used to detect near collisions and crowd movement analysis in wide view areas.

2.6.1 Review of Multi-View Visual Surveillance Systems

The above focused on traffic monitoring of both humans and vehicles, but as already stated most work focus on either humans or vehicles only. In this section the focus is on multi-view systems which may not cover both object types. Using multiple views is relevant for systems monitoring traffic because it provides a larger viewing area and is efficient in resolving occlusion issues. However, applying multiple views also introduce some issues. According to [Hu et al., 2004b] some of these issues are installation, camera registration, object matching, switching and data fusion. The issue with installation is the task of covering the entire scene with the minimum of cameras. This issue depends heavily on the specific scene.

Some issues are related such as the issue of object matching between views depends on how the cameras are registered. Other object matching approaches can be used if the full camera calibration is available compared to when no information of the cameras are available. In the following, the focus is on how the reviewed systems deal with these issues. The following systems mainly focus on tracking of objects which is the most dominant research area in multi-



(a) View 1 (b) View 2

(c) Planar homography domain (d) Spatial-temporal activity space analysis

FIGURE 2.5: Person-vehicle interaction analysis by [Park and Trivedi, 2006]. The vehicle and human is tracked in the view invariant planar homography domain as two blobs. Near collision detection is achieved using the spatial-temporal activity space.

view systems. Two overall approaches can be identified in the reviewed literature; recognition based methods and geometry based methods. The first mainly utilizes the color of the tracks in matching the objects between views. [Orwell et al., 1999] and [Krumm et al., 2000] are examples on work using color-based tracking where the color histograms is applied when matching the people in different views. This might be the only option when cameras are not overlapping, but even with overlapping cameras the appearance of the same object might be reproduced very differently in the different views.

The second approach is the geometry based methods. An example is found in [Mittal and Davis, 2003], where human tracking in complex scene with severe occlusion is the focus. The humans are segmented using the Bayesian classification rule, and data from multiple cameras are fused to estimate the humans on the ground plane. Another example of the geometry based method is found in [Park and Trivedi, 2006] where tracking is performed in the view invariant planar homography domain. When using geometry based methods, registration of the cameras is required.

Works where the two approaches are combined are presented in the work by [Kang et al., 2003] and [Chang and Gong, 2001]. [Kang et al., 2003] uses both stationary and moving PTZ cameras. This work is based on probabilistic information fusion using color histogram in a polar representation and by predicting the object position using a Kalman filter. [Chang and Gong, 2001] performs tracking of the objects based on Bayesian belief networks where five modalities are fused. The three geometry based modalities use epipolor geometry, homography and landmarks and the recognition based modalities use the height and color of the people.

When reviewing the most recent literature it was found that the work using the geometry based methods are preferred and also presents the best results. However as explained earlier, little work is done on monitoring both humans and vehicles simultaneously, and all the methods mentioned above, except one, only consider the human case. The exception is [Park and Trivedi, 2006] where the objects are tracked using the footage region. Objects occupying a larger area on the ground plane results in a larger footage region in the homography domain. Since humans are significantly smaller than vehicles there are some issues using this method. The footage region is also used in [Khan and Shah, 2006], but here the goal is to track individual persons interacting in dense crowded scenes. According to [Khan and Shah, 2006] the footage region is vulnerable to shadows and might falsely detect a person. The authors solve this issue by adding cameras and use up to four cameras to monitor an area of 5×5 meters.

As stated earlier in this analysis, when monitoring traffic scenes a larger area should be covered, and having four overlapping cameras does not seem plausible. Furthermore, in case the feet of the persons are not detected it is not possible to detect the person by the footage region method. These issues indicate that the footage region is not a robust human tracking method. However, in the vehicle case the footage region performs well as reported in [Park and Trivedi, 2006]. This is because vehicles are significantly larger than humans and occupy a larger area on the ground which makes the footage region more robust in the vehicle case. This is illustrated in Figure 2.5(c), where the large blob is the vehicle and the small blob is the human.

A more robust approach is needed when monitoring humans. Two often referenced systems on tracking of partly occluded humans using multiple cameras is the work by [Mittal and Davis, 2003] and [Dockstader and Tekalp, 2001]. [Dockstader and Tekalp, 2001] fuses independent observations from multiple cameras using a Bayesian network and produces the most likely vector of 3-D state estimates given the available data. Both works require full camera calibration and is only used in indoor scenes. According to [Valera and Velastin, 2005], full camera calibration is a resource-consuming process and a skill-demanding task in practice, especially in outdoor scenes. [Kim and Davis, 2006] is related to the work in [Mittal and Davis, 2003], but does not require the full camera calibration, only the planar homography is required; the planar homography of a scene is easier to recover compared to the full camera calibration. Matching between views in [Kim and Davis, 2006] is performed using the so-called principal axis method and is incorporated into a particle filter framework. The method of the principal axis is presented in [Hu et al., 2006] where it is used differently to track humans. Both [Kim and Davis, 2006] and [Hu et al., 2006] present tracking results in outdoor scenes. Compared to the footage region, the principal axis is robust to some misdetection of the human body and is able to locate the position of people even if they are partly occluded in all views, but the principal axis can not model vehicles.

In conclusion, two interesting approaches have been identified: the footage region method and the principal axis method. Both methods are geometry based methods and only requires the planar homography. There is a tendency in the latest publications towards preferring the use of planar homography instead of the more traditional full camera calibration, because the homography is easier to recover. The footage region method is not as robust in the human case as in the vehicle case. The principal axis method is a robust method in the human case,

but does not apply in the vehicle case. Combining the two methods could therefore increase the performance. Furthermore, such a combination has not been reported in the literature.

2.7 Thesis Definition

The problem analysis has identified several requirements for a visual surveillance system with situation awareness for monitoring traffic. A visual surveillance system consists of several stages as explained using the general framework introduced in Section 2.3. The output from lower-level stages are given to higher level stages making each stage dependent on the output from a lower level stage. Developing a visual surveillance system requires that the higher-level stages are build upon a robust set of lower-level stages. In this thesis the focus is put on building that foundation using multiple cameras which make it possible to cover a large monitoring area and has been efficiently applied in related work to resolve occlusion issues. The main goal for the thesis is given below.

> ***The goal is to enhance automatic situational awareness by building a system capable of robustly tracking humans and vehicles through their activities and interaction in an unconstrained outdoor environment using multiple surveillance cameras.***

Using the level terms of situational awareness introduced in Section 2.1, this work focuses mainly on level 1 and in some aspects level 2. Based on the findings in the problem analysis, a set of secondary goals are given which further defines the scope of the project. The delimitations and assumptions used in the work are listed next.

2.7.1 Secondary Goals

The following lists secondary goals which are derived from the findings in the problem analysis.

- The system must operate with elevated cameras.
- The system must be able to distinguish humans from vehicles.
- People move in groups and it is a goal to be able to distinguish a single human from a group of humans. Furthermore, individuals must be tracked in the group.
- Full camera calibration must not be used.
- The footage region method presented in [Park and Trivedi, 2006] must be combined with the principal axis method presented in [Hu et al., 2006].
- The system must be able to extract a view invariant representation of the objects. This representation is well suited for analyzing the objects' interactions and can be used for detecting an accident situation.
- The tracking result should be available with little or no delay. Otherwise, a possible accident situation is first detected after it has occurred.[1]

[1] This goal is not the same as the system must perform in real-time.

2.7.2 Delimitations and Assumptions

The following lists the delimitations or assumptions used in the thesis.

- Humans are assumed to be walking upright. People on bikes, roller skates or skateboard are not explicitly addressed.
- The ground is assumed to be planar and views are assumed to be overlapping. This restricts the monitored area. The footage region method and principal axis method both uses planar homography for camera registration and hence assumes a planar ground plane.
- Artificial lights are not considered. The system is only expected to track objects during daytime, where the sun is the dominant light source. The reason being that most traffic is present at daytime.

Chapter 3

System Overview

With the task given in the thesis definition, this chapter gives an overview of the applied system structure. The system is divided into modules, and the remaining report reflects the chosen system structure.

3.1 System Overview

This section describes the applied system structure. The structure is based on the general framework presented by [Hu et al., 2004b] and is described in Section 2.3 on page 15. The system consists of four types of modules and is shown in Figure 3.1. For each camera input a foreground segmentation module and a single view tracking module are created. Only a single correspondence module and view invariant module exists in the system. The structure is shown for a two camera setup, which is the setup used in the thesis.

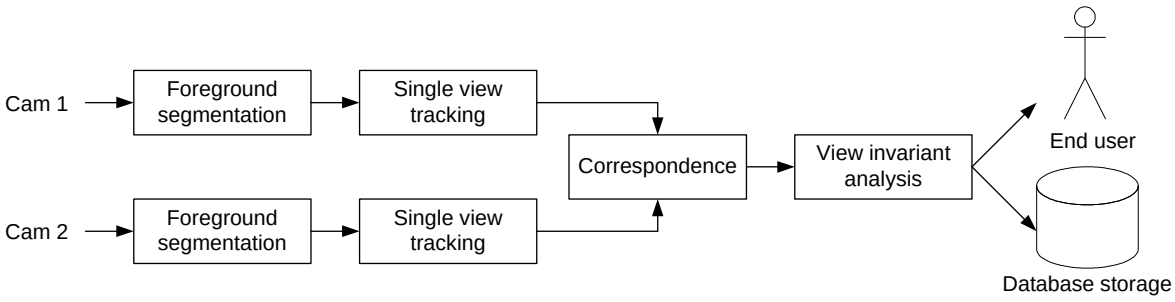


FIGURE 3.1: Overview of the system showing the individual modules.

The following description explains the structure bottom-up, meaning starting from the input from one camera to the output given to the end user and database. The cameras are synchronized. For each camera input moving objects are segmented using the foreground segmentation module, and the moving objects are tracked using the single view tracking module. The tracked objects in each separate view are matched in the correspondence module. The tracking performed in the single view tracking modules yields a view variant representation of the objects, but given that the ground plane coordinates are known the correspondence module provides a view invariant representation of the objects. The view invariant representation can be used to perform view invariant analysis of the objects and their actions and interactions and thus identify normal and abnormal situations. This is carried out by the view invariant analysis module. The output from the system is the track history for each object in the scene along with the result from the view invariant analysis. The track history

includes both the spatial-temporal history and an object classification for each track. The output of the system is displayed on a monitor to an end user who can react on the situation. Furthermore, the output is stored in a database; in the current implementation as a single file for each tracked object.

It is a goal to combine the footage region method and the principal axis method as stated in Section 2.7.1 on page 25. This is reflected in the chosen system structure where single view tracking is performed before fusion of information from other views. This is a prerequisite for using the principal axis method as introduced in [Hu et al., 2006]. The footage region method as used in [Park and Trivedi, 2006] does not apply single view tracking, but is needed in combination with the principal axis method.

3.2 Readers' Guide

In the following, a chapter is devoted to the handling of camera registration issue and a chapter to each module in the system except the view invariant analysis module, reflecting the chosen structure of the system making the order of the following chapters: Camera Registration, Foreground Segmentation, Single View Tracking and Correspondence of Objects. Each chapter describing a module covers analysis, design and test of the module. Following these chapters, the system is tested and the results are discussed in Chapter 8. The task of the view invariant analysis module depends on the specific application. In Chapter 9 event recognition using view invariant analysis of the objects is introduced. The event recognition is used to demonstrate and evaluate the underlying tracking system with regards to detecting a traffic accident before it occurs. The thesis is ended by a conclusion and outlook.

Chapter 4

Camera Registration

The following deals with registration of the cameras for plane-to-plane mapping in a multi-view setup. Initially, an analysis of camera registration approaches is presented. This is followed by conceptual design and presentation of the obtained results.

4.1 Analysis

The task of the camera registration is to do mapping of points on the ground plane between views since this is a prerequisite of both the footage region method in [Park and Trivedi, 2006] and the principal axis method in [Hu et al., 2006]. As stated in Section 2.7.2 on page 26 it is assumed that the ground plane on which objects move is planar. Furthermore, it is specified under the secondary goals in Section 2.7 on page 25 that it is a goal not to use full camera calibration. Therefore, an approach for doing plane-to-plane mapping without full camera calibration must be found.

Plane-to-plane mapping between the image planes for each camera can be calculated algebraic using perspective transformation also called homography mapping [Hartley and Zisserman, 2004]. A sketch of the mapping is presented in Figure 4.1. The figure shows a point $\mathbf{X}_\pi$ on the ground plane π within the shared region of the two cameras. The projection of $\mathbf{X}_\pi$ in each view is mapped between the two image planes using the homography. The homography is represented by a 3×3 matrix $\mathbf{H}$. With $\mathbf{H}_{12}$ matrix determined, the mapping from view 1 to view 2 is straightforward. $\mathbf{H}_{21}$ is obtained by inverting $\mathbf{H}_{12}$ for mapping in the opposite direction.

The homography matrix can be computed from the relative position of the two image planes

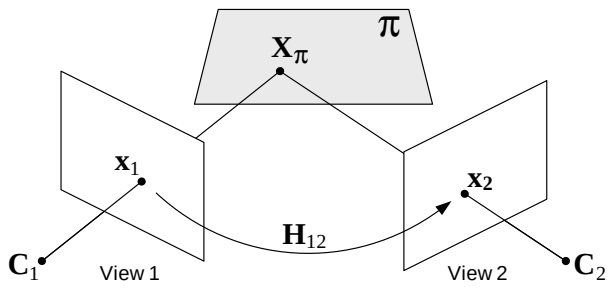


FIGURE 4.1: Mapping of a ground plane point $\mathbf{X}_\pi$ between two views. The projection of $\mathbf{X}_\pi$ in view 1, point $\mathbf{x}_1$, is mapped to the projection of $\mathbf{X}_\pi$ in view 2, point $\mathbf{x}_2$, using the homography matrix $\mathbf{H}_{12}$. $\mathbf{C}_1$ and $\mathbf{C}_2$ are the respective camera centers.

and camera centers [Criminisi et al., 1999]. However, it can also be computed directly using corresponding primitives between the two planes, such as points, lines, polygons, contours, curves or textures [Jain and Jawahar, 2006]. However, since point-to-point correspondence is widely used and successfully applied in related work such as [Hu et al., 2006, Park and Trivedi, 2006] the homography estimation is based on point correspondence.

$\mathbf{H}$ has eight degrees of freedom and is only defined up to a scale [Hartley and Zisserman, 2004]. From each pair of corresponding points two independent linear equations can be derived. This means that each point correspondence $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$ accounts for two degrees of freedom. Since $4 \cdot 2 = 8$ and given that there is no three collinear points on either plane, four points correspondence are sufficient for estimation of the homography [Hartley and Zisserman, 2004]. Four point correspondence is therefore a minimal solution with eight equations, but more points can be added for increased estimation accuracy.

Though the minimal solution allows calculation of an exact solution, it is unlikely to be the case in practise. Perfect image measurements can not be guaranteed as a result of noise in selection of point correspondence e.g. due to camera distortion, human error in selection or quantization due to rounding of coordinates to nearest integer value. Furthermore, a completely flat ground plane can not always be guarantied. The calculation of $\mathbf{H}$ therefore becomes an approximate solution.

The eight equations can be solved by Gaussian elimination, for an inhomogeneous solution. More than four points results in an over-determined set of equations, that can be solved by least-squares minimization [Hartley and Zisserman, 2004]. The inhomogeneous solution is solved by setting one of the entries in the solution vector to the value one. However, in case the selected entry in fact should be zero, no true solution can be found. This can lead to unstable results, and the method is therefore not recommended in general [Hartley and Zisserman, 2004]. To avoid this, a homogeneous solution can be found using the Direct Linear Transformation (DLT) algorithm.

In [Agarwal et al., 2005] a review and comparison of several homography estimation techniques are presented. A tendency for improved performance is observed when the average distance between correspondence points increase. Therefore it makes sense to select widely spread points within the shared region of the two cameras. [Agarwal et al., 2005] also confirms that more correspondence primitives allow a better estimation of the homography [Agarwal et al., 2005]. For more accurate results, algorithms that reduce sensitivity towards outlines like Random Sample Consensus (RANSAC) and Least Median of Squares can be used [Yue et al., 2004].

The calculation of the homography is done off line. Though methods exist for self-calibration [Khan and Shah, 2003, Calderara et al., 2005], it is beyond the scope of this work. However, in future work, such methods could be applied to create a more autonomous system.

4.1.1 Choice of Camera Registration Method

It is decided to base homography estimation on four point correspondences with the possibility of adding more points for increased accuracy. A homogeneous solution is found to the point equations using the DLT algorithm in [Hartley and Zisserman, 2004], as it is recommended not to use an inhomogeneous solution. Furthermore, rather than applying estimation methods like

RANSAC for robustness it is decided to take a more practical approach. By implementation of a tool to ease the homography estimation process, a trial and error approach can be taken when estimating the homography matrix. The developed homography estimation tool is described in Appendix F.3 on page 158.

Since the homography does not model non-linear distortion, it is necessary to consider the effects of radial camera distortion. However, since this work is intended to be used for multiple setups, no specific setup can be assumed. The cameras are therefore assumed to follow the pinhole model which ignores the non-linear radial distortion. Camera distortion must therefore be corrected beforehand if it is considered a significant cause of inaccuracy.

4.2 Conceptual Design

Corresponding points are selected to fulfill the basic requirements; the points must be on the ground plane, no three collinear points and as widely separated as possible. The developed homography estimation tool allows estimation based on video inputs from each view. Point correspondences can therefore be collected from a person moving around in the scene as well as from static landmarks on the ground plane.

As stated under the secondary goals in Section 2.7 on page 25, the system must hold a view invariant representation of objects. This view invariant representation is well suited for behavior analysis, because the size of objects and distances between objects have a meaningful relationship. Furthermore, the same analysis approach can be applied to different camera setups. This view invariant representation is best established using world coordinates. To be able to visualize this view invariant representation of objects, it is therefore very useful to have a view defined from world coordinates, e.g. a virtual top-down view of the scene. Therefore, to allow for view invariant analysis in later modules, a virtual view, ν, is introduced when mapping points between views. This is depicted in Figure 4.2. The direct plane-to-plane mapping between views is preserved, but it is handled transparently by mapping through a virtual view. This requires a homography matrix from each image plane to the virtual view. Points in the virtual view can be selected arbitrary. However, if world coordinates for the ground plane is available, a virtual top-down view of the scene can be generated.

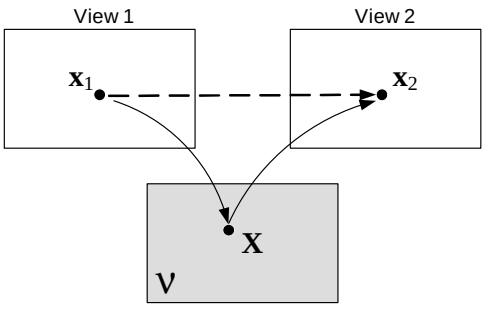


FIGURE 4.2: Mapping between two corresponding image points using a virtual view. Point $\mathbf{x}_1$ in view 1 is mapped to this corresponding point, $\mathbf{x}_2$, in view 2 through the virtual view ν.

The following documents the estimation of the homography based on the choices above. This

is followed by examples of homography estimation results for two scenes based on 4 and 6 point correspondences, respectively.

4.3 Estimating the Homography

The homography estimation is based on the DLT algorithm. The algorithm finds a homogenous solution to a system of linear equations using Singular Value Decomposition (SVD) [Hartley and Zisserman, 2004]. For notation, the entries of $\mathbf{H}$ are represented as:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{4.1}$$

Furthermore, points in the image plane are represented by lower case, $\mathbf{x}$, and the corresponding points in the virtual view are represented by upper case $\mathbf{X}$. All points are represented as homogeneous 3-vectors, $\mathbf{x} = (x, y, 1)^T$ and $\mathbf{X} = (X, Y, 1)^T$. Corresponding points are related by:

$$\mathbf{X} = \mathbf{H}\mathbf{x} \tag{4.2}$$

with "=" meaning equality up to scale [Criminisi et al., 1999]. Equation 4.2 may be expressed as the vector cross product $\mathbf{X} \times \mathbf{H}\mathbf{x} = \mathbf{0}$. From this a linear solution for $\mathbf{H}$ can be derived [Hartley and Zisserman, 2004]. Each pair of corresponding points define two linear independent equations. These can be determined by rewriting the cross product. The equations for a pair of corresponding points are:

$$\begin{aligned} h_{11}x + h_{12}y + h_{13} &= h_{31}xX + h_{32}yX + h_{33}X \\ h_{21}x + h_{22}y + h_{23} &= h_{31}xY + h_{32}yY + h_{33}Y \end{aligned} \tag{4.3}$$

For n point correspondences, $2n$ equations can be obtained. To determine all eight degrees of freedom for $\mathbf{H}$, an equation system with eight unknowns must be solved. This requires $n = 4$ for a minimal solution or $n > 4$ for an over-determined solution. To find the homogeneous solution, a system of linear equations is defined as $\mathbf{A}\mathbf{h} = 0$ [Hartley and Zisserman, 2004]. With $\mathbf{A}$ being a $2n \times 9$ matrix where each of the n point correspondences contribute with two rows, derived from Equation 4.3, in matrix $\mathbf{A}$. $\mathbf{h}$ is the unknown entries in $\mathbf{H}$ rewritten as a vector, $\mathbf{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$. For n correspondences, $\mathbf{A}\mathbf{h} = \mathbf{0}$ becomes:

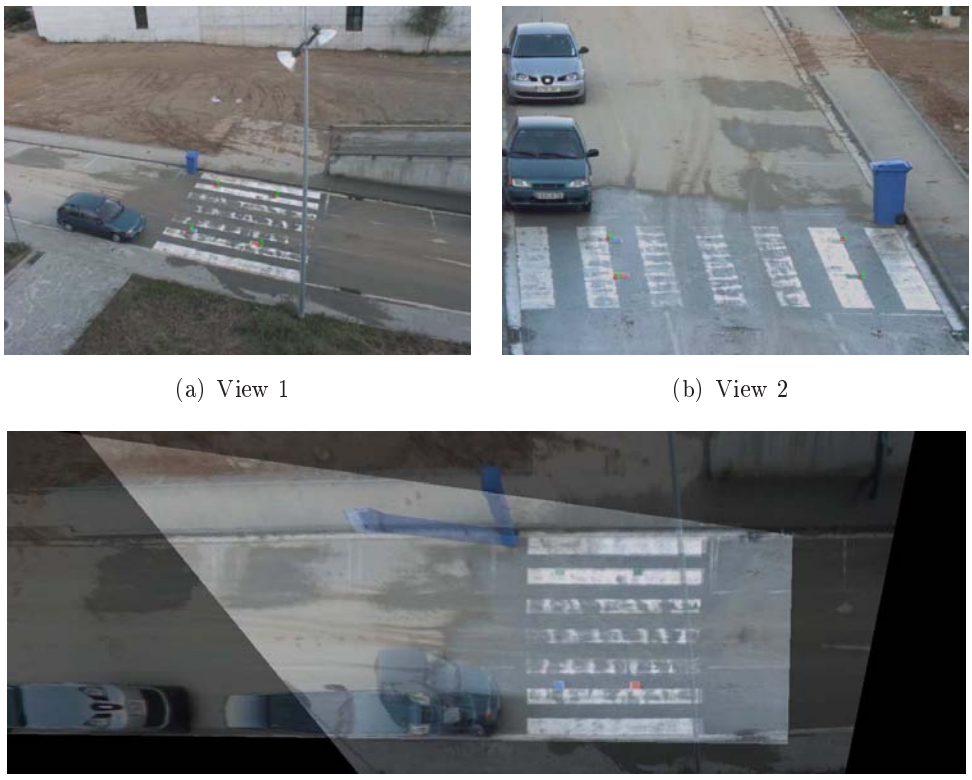
$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & \text{-}x_1X_1 & \text{-}y_1X_1 & \text{-}X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & \text{-}x_1Y_1 & \text{-}y_1Y_1 & \text{-}Y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i & y_i & 1 & 0 & 0 & 0 & \text{-}x_iX_i & \text{-}y_iX_i & \text{-}X_i \\ 0 & 0 & 0 & x_i & y_i & 1 & \text{-}x_iY_i & \text{-}y_iY_i & \text{-}Y_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & \text{-}x_nX_n & \text{-}y_nX_n & \text{-}X_n \\ 0 & 0 & 0 & x_n & y_n & 1 & \text{-}x_nY_n & \text{-}y_nY_n & \text{-}Y_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \mathbf{0} \tag{4.4}$$

Vector $\mathbf{h}$ that minimizes the algebriac residual $||\mathbf{Ah}||$, is obtained directly from the SVD of $\mathbf{A}$ [Criminisi et al., 1999]. Using SVD, $\mathbf{A}$ is factorized into $\mathbf{A} = \mathbf{UDV}^T$. The solution to $\mathbf{h}$ is then given by the column of $\mathbf{V}$ corresponding to the smallest singular value in $\mathbf{D}$ [Hartley and Zisserman, 2004]. Given that the diagonal entries of $\mathbf{D}$ is sorted in descending order, it is also said that the solution to $\mathbf{h}$ corresponds to the rightmost column of $\mathbf{V}$. The SVD also resolves the issue of avoiding a trivial solution with all entries of $\mathbf{h}$ being zero.

4.4 Homography Estimation Results

In the following, two examples of homography estimation using the homography estimation tool are presented. The first setup is from the HERMES dataset and the second is from the Matthews Lane dataset. For an elaborate description of the utilized datasets, please see Appendix D on page 147. To enable a top-down virtual view of the scenes, world coordinates are applied in both cases. In HERMES these are obtained from known markers on the road, and in Matthews Lane these are obtained by a person pacing out the scene.

Point correspondence are selected by clicking in the images with a mouse. The estimation is most sensitive to selection of points in side or wide view setups where a pixel displacement



(a) View 1

(b) View 2

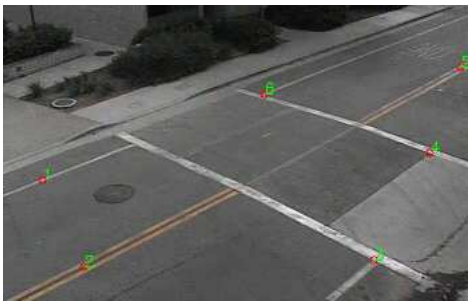
(c) Virtual view

FIGURE 4.3: Camera setup from the HERMES dataset. The plane-to-plane homographies are calculated by point correspondence, as marked by the red circles in the two camera views. The overlapping area between the two views are clearly visible as a bright region in the virtual view.

corresponds to a large distance in world coordinates compared to narrow or top-down view setups. However, in general the precision in manual selection of the correspondence points is sufficient to obtain a good homography estimate.

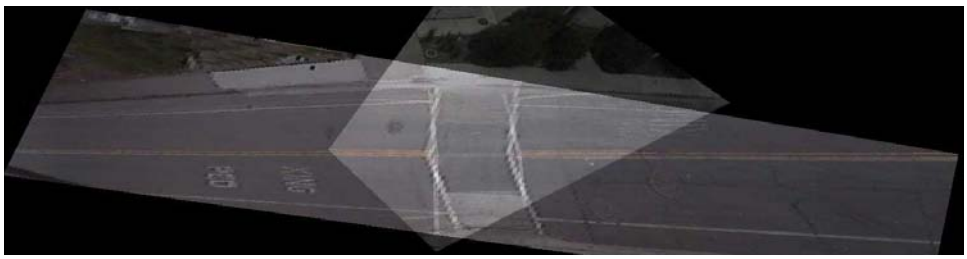
The HERMES setup is shown in Figure 4.3(a) and Figure 4.3(b). The homographies for this scene is estimated using four corresponding points indicated by the red circles. From the estimated homographies a virtual view is generated by mapping both views into the virtual view as depicted in Figure 4.3(c).

The second example is the narrow view setup from Matthews Lane as depicted in Figure 4.4(a) and Figure 4.4(b). The virtual view of this scene is depicted in Figure 4.4(c). The mapping in this setup is not completely accurate. This is seen from the straight lines of the pedestrian crossing which are bend in the virtual view. This is due to violation of the assumption about completely planar surfaces. The inaccuracy occurs due to the profile of the road which is sloping from the middle of the road towards the curbsides. Because the sloping of the road is so pronounced in this setup, six correspondence points are used to find the most accurate estimate of the homographies. The sloping of the road is pronounced due to a more side view setup of the camera in Figure 4.4(b).



(a) View 1

(b) View 2



(c) Virtual view

FIGURE 4.4: Narrow view camera setup from the Matthews Lane dataset. The plane-to-plane homographies are calculated by point correspondence, as marked by the red circles in the two camera views. The overlapping area between the two views are clearly visible as a bright region in the virtual view.

Given that the assumption about a planar ground plane is valid as in the HERMES dataset, the estimation results are very accurate. In spite of the described inaccuracies in the Matthews Lane dataset, experiments showed that the principal axis and footage region method performed well (See Chapter 7 on page 83 for more details). The forthcoming modules are therefore expected to function with the results presented above. This means that any methods

that utilizes the homography mapping must incorporate some robustness towards inaccurate mapping.

4.5 Summary

The main choices and finding made in the work on camera registration are:

- Plane-to-plane mapping is done using homography estimation.
- Homography estimation is based on points correspondence rather that higher order primitives like lines and contours.
- To compute the homography matrix a homogenous solution is found to a set of linear equations by the DLT method using SVD and a minimum of four point correspondence.
- To enable a view invariant representation of the scene a virtual view is introduced.
- Knowledge about world coordinates is not required unless the virtual view should be used for view invariant analysis.
- The homography mapping is in general very accurate, but perfectly accurate estimation can not be guarantied if the surface it not completely planer.

Chapter 5

Foreground Segmentation

The foreground segmentation module is the first step of processing the video sequences from the cameras. The foreground segmentation allows the forthcoming modules to focus their attention on the areas containing objects. This chapter documents the analysis and design of motion segmentation. This is followed by analysis and design of a dedicated shadow suppression method. The chapter is ended with a test of the module.

5.1 Chapter Overview

The purpose of the foreground segmentation module is to segment the objects from the background in the video sequences from the cameras. The challenge lie within ensuring that all the relevant and only the relevant information is segmented as foreground. When having the background removed, the focus in the later modules can be on classifying and tracking objects and analyzing their behavior. The foreground segmentation module's place in the overall system structure is shown in Figure 5.1. The foreground segmentation module receives input from a camera and delivers a foreground mask as output for the forthcoming modules. Binary values denoted by black and white represents background and foreground, respectively.

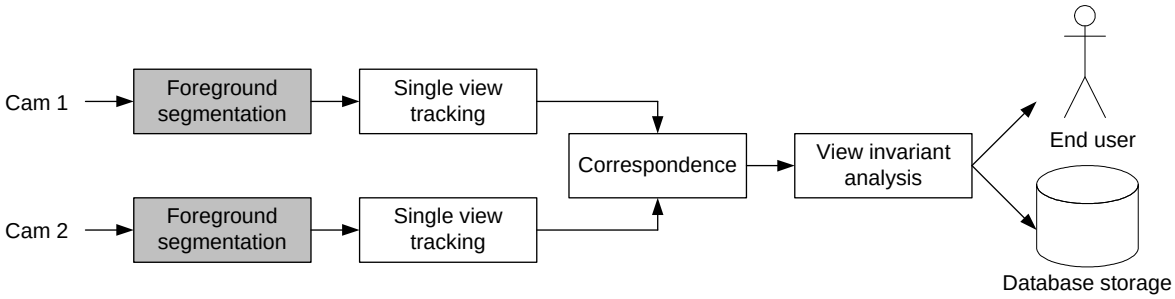


FIGURE 5.1: System overview highlighting the foreground segmentation module.

The foreground segmentation module consists of a motion segmentation submodule followed by a dedicated shadow suppression submodule as depicted in Figure 5.2. Since the shadow suppression submodule is self-contained, the motion segmentation and shadow suppression are documented separately. Methods exist that merge motion segmentation and shadow suppression into one as in [Doshi and Trivedi, 2006], but such methods are not taken into consideration. First, general considerations regarding the scene in which the system is to function are presented. This is followed by analysis and design of the motion segmentation in

Section 5.2 through 5.7. Following that, the analysis and design of the shadow suppression is presented in Section 5.8. As a final step, post-processing is applied in Section 5.9. The test of the foreground segmentation module is presented in Section 5.10.

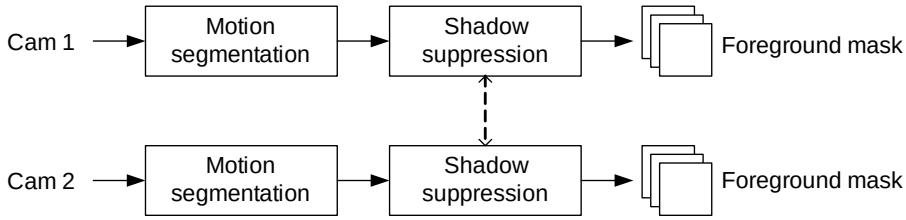


FIGURE 5.2: Overview of the foreground segmentation module.

5.1.1 General Considerations

In the problem analysis in Section 2.4 on page 17 a number of computer vision challenges are listed. The following emphasizes the challenges related to foreground segmentation and defines basic terms used in this chapter:

Object movement In an unconstrained environment it must be assumed that foreground objects move arbitrarily. Both humans and vehicles may move at different speeds and stop for various reasons. This is a challenge because foreground objects can not be assumed to always be moving.

Background camouflage Background camouflage occurs when objects are colored similar to the background or e.g. due to transparent windows in vehicles. A common type of background camouflage is humans with grayish pants walking on a gray road or sidewalk.

Reflections Reflections occur in both foreground and background, which often are caused by the surface of windows or vehicles. Another cause of reflection is due to rain which makes many surfaces reflective. Reflections can cause foreground objects to reflect the background and vice versa.

Illumination Changes Both objects and the scene change appearance during the day as the sun's position changes, as shown in Figure 2.4 on page 19. Moving clouds can also cause sudden illumination changes in the scene.

Shadow In outdoor scenes the sun causes cast shadows from both background and foreground objects. Cast shadow is considered background since it is not a part of the foreground objects. Moving cast shadows from foreground objects are background that are particular difficult to segment, both due to the significant intensity change caused by the shadow and due to similarity with the object's motion, shape and size.

Scene dynamics Vegetation like tree branches moving in the wind should be segmented as background. In general, precipitation and wind can be the cause of both sudden and periodic changes in the scene.

Changes in the scene Any number of things can be added, moved or removed from the scene which changes its appearance like dirt, water, leafs etc. on the road or entrances opening and closing. Inanimate objects placed in the scene must also be handled.

These challenges are used in the following analysis.

5.2 Analysis of Motion Segmentation Methods

To provide a focus of attention in the forthcoming modules, a method of segmenting the motion in a video sequence is very useful. The survey in [Hu et al., 2004b] presents three basic methods for detecting motion. These are described in the following and the choice of method is presented.

Background subtraction Background subtraction detects motion by comparing the current frame with a background model. The method depends on continuously maintaining a good background model. Dynamic updating of the background model is therefore required if it must function in a dynamic and changing environment. The concept is illustrated in Figure 5.3.

Temporal differencing Temporal differencing computes the difference between the current frame and the previous frame(s). A threshold defines which pixels that are classified as foreground. The method is very adaptive to dynamic environments but does not always detect the entire foreground object due to slow motion or uniform coloring as depicted in Figure 5.4.

Optical flow Optical flow calculates flow vectors for each pixel to determine motion based on the similarities between a pixel and its neighbors in a sequence of frames. The method can be used for handling camera motion. But it is in general computationally demanding and sensitive to noise [Hu et al., 2004b]. The concept is illustrated in Figure 5.5.

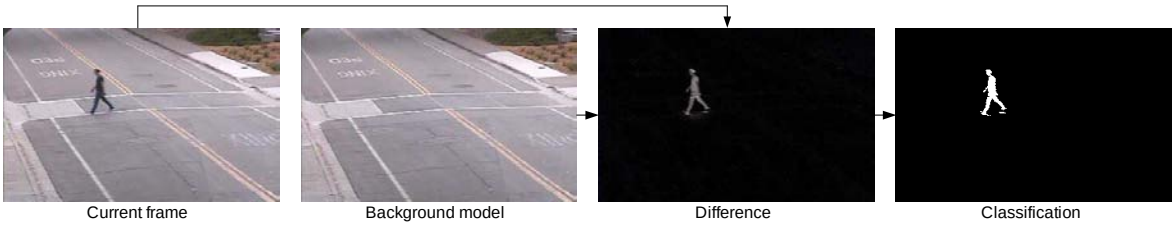


FIGURE 5.3: The concept of background subtraction: the pixel-wise difference between the current frame and the background model is found. Differences above a threshold are classified as foreground.

Discussion and Choice of Motion Segmentation Method

The methods ability to handle the challenges presented in Section 5.1.1 forms basis for the choice of method. The optical flow method gives no direct advantages since handling of moving

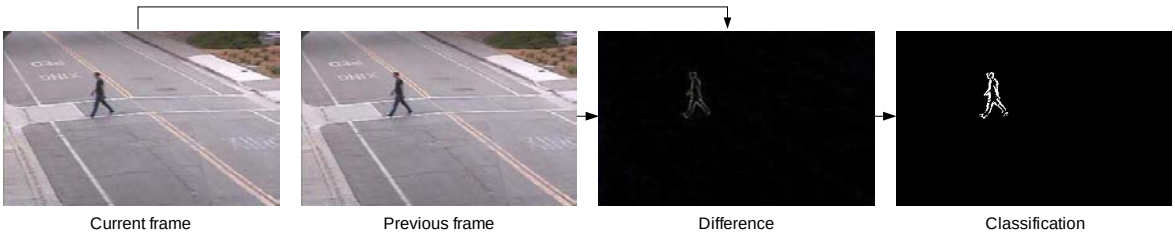


FIGURE 5.4: The concept of temporal differencing: the pixel-wise differences between a number of consecutive frames (usually two or three) is found. Differences above a threshold are classified as foreground.

FIGURE 5.5: The concept of optical flow: for each pixel in a frame a motion vector is calculated. If the motion vector is large enough the pixel is considered foreground.

cameras is not considered an issue. Temporal differencing is invariant towards illumination changes but has the demerits of not detecting the entire object as well as not being able to detect slow object movement. In contrast, stationary and slowly moving objects can be handled by the background subtraction method, while it is still possible to be tolerant towards illumination changes.

For these reasons, it is decided to use background subtraction for motion segmentation. This is an area of active research, and many interesting background modelling methods have been proposed.

5.2.1 Background Modelling Methods

Background subtraction is very dependent on the choice of the background modelling method since it determines the models ability to adapt to changes and model scene dynamics. Providing that a good background model can be obtained, the actual background subtraction is straightforward. The difficulties therefore lie within creating and maintaining a robust background model.

Several methods exist for background modelling. However, for modelling of challenging dynamic and changing scenes, two method are dominant, the Mixture of Gaussians (MoG) and the Codebook method. These two methods are described in the following. A description of an additional six prominent methods are available in Appendix A.1 on page 127.

Mixture of Gaussians The MoG method models the background of the scene using a number of Gaussians for each pixel. In [Stauffer and Grimson, 1999], a method is proposed where a number of Gaussians are determined beforehand. However, methods for dynamically adjusting the number of Gaussians exist e.g. [Christensen and Nikolajsen, 2005].

In [Wang and Suter, 2005], some practical issues relating to illumination changes and relocation of objects in the background are presented along with solution proposals.

Codebook The Codebook method is a multimodal method that describes the background colors of each pixel using a number of cylinders in RGB space, called codewords. The collection of codewords for a background pixel constitutes the codebook for that pixel. The Codebook method is described in [Kim et al., 2005] along with methods for dynamically and adaptively updating the background model. Extensions to the updating process are proposed in [Andersen and Corlin, 2005].

Discussion and Choice of Background Modelling Method

A comparison between MoG and Codebook is presented in [Chalidabhongse et al., 2003]. In [Chalidabhongse et al., 2003], a perturbation test is suggested and performed. By randomly perturbing the color vectors in the image increasingly, the detection rate of the method tested can be evaluated. A perturbation means a change in chromaticity or brightness. The pixel becomes increasingly different from the background the more perturbed it gets. The performance measure is the amount of perturbed pixels detected as foreground. The higher detection rate with small perturbation, the better the method. The results unanimously indicates the superiority of the Codebook method, as this has the highest sensitivity and detects the perturbed pixels at the lowest perturbation magnitude. This higher sensitivity for the Codebook method enables it to do better segmentation when foreground camouflage is present. Two of the graphs presented in [Chalidabhongse et al., 2003] are shown in Figure 5.6, indicating that the Codebook method performs best in both indoor and outdoor environments.

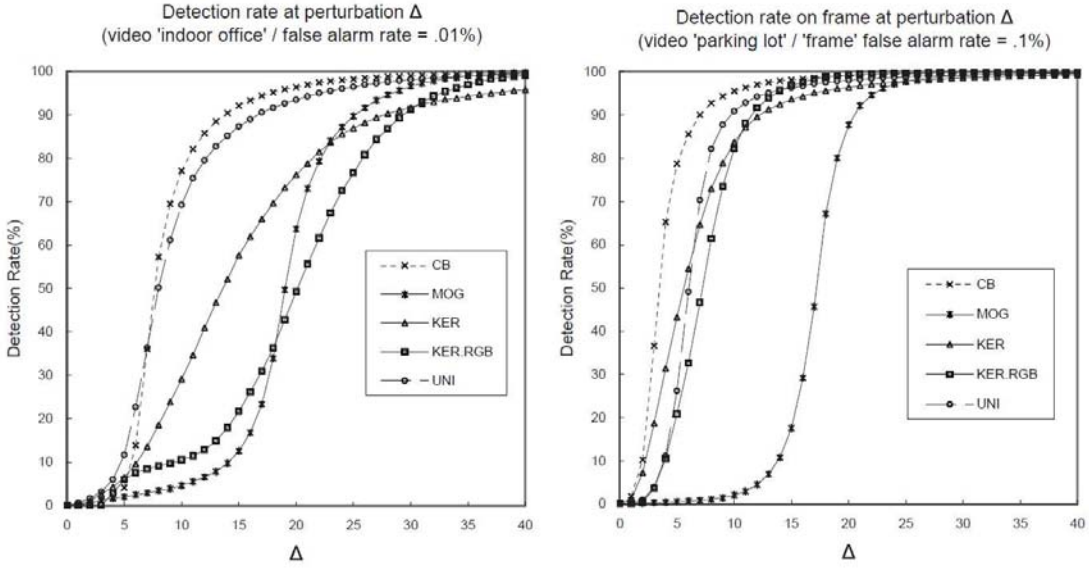


FIGURE 5.6: Result graphs of the perturbation test indicating the superiority of the Codebook method, both indoors and outdoors. The interesting curves are the Codebook (CB) and Mixture of Gaussian (MoG). Figure from [Chalidabhongse et al., 2003].

However, some improvements to the MoG method that was used for comparison in [Chalidabhongse et al., 2003] have been suggested, e.g. as those mentioned in

[Wang and Suter, 2005]. These could have changed the outcome of the tests as they address some of the issues with MoG pointed out in the literature. Recent work within this field, in [Christensen and Nikolajsen, 2005] and [Andersen and Corlin, 2005], have used MoG and Codebook, respectively. Both obtained reasonable results with improvements to the methods. No bias can really be drawn from their conclusions without further, more detailed investigation. However, the Codebook method has proven to perform well in our earlier work [Hansen et al., 2007]. Therefore, the Codebook method is chosen for foreground segmentation in this work.

5.2.2 Analysis Summary

In this analysis two choices are made. First, it is chosen to discard temporal differencing and optical flow in favor of background subtraction for motion segmentation. To be able to perform background subtraction robustly, a number of methods for building a background model are investigated as well as the possibilities for dynamically maintaining the model. The Codebook method is chosen in favor of MoG, as it has proven to perform well in earlier work and because it has great potential for handling dynamic scenes.

5.3 Conceptual Design

This section describes the conceptual design of the motion segmentation. The design is based on the choices made in the analysis. In order to make a model of the background using the Codebook method, a number of frames are used to train the model. After the training is completed the background model can be used to detect foreground objects. This phase is referred to as online classification. The conceptual design of the foreground segmentation is depicted in Figure 5.7, with the background model illustrated by a cylinder and the training and online classification illustrated by shaded boxes.

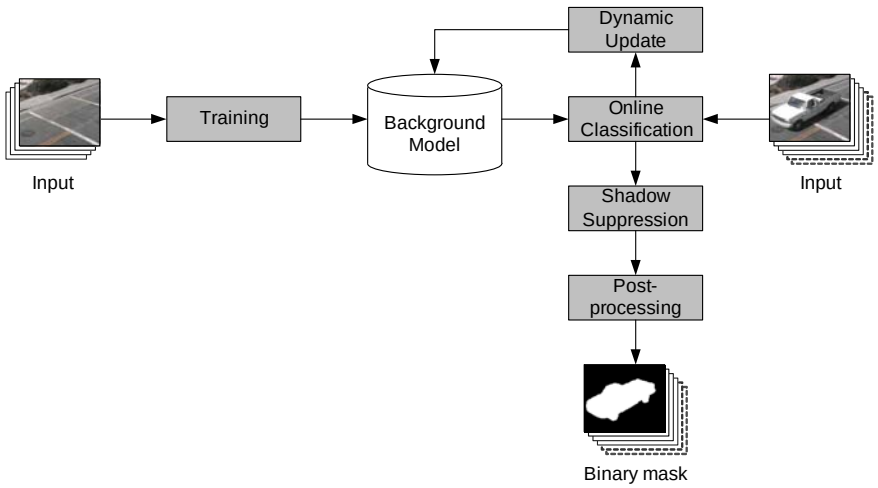


FIGURE 5.7: Conceptual design of the foreground segmentation module.

During online classification the model needs to be updated. The purpose of updating is twofold. Firstly, the illumination of the scene change over time, and the background model

must adapt accordingly. Secondly, static objects may be added to the scene after training, and these objects must be added to the background model.

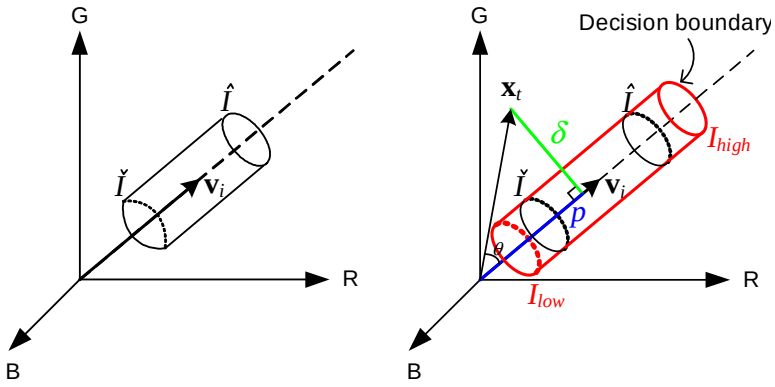
The output of the motion segmentation is a binary mask. After motion segmentation, shadow suppression is performed. Some isolated pixels may be falsely classified as foreground, and to avoid this post-processing is performed before outputting the binary mask.

In the following, it is explained how the Codebook method models the background using codewords. The training of the background model and construction of the codewords are explained afterwards. This is followed by how online classification is performed. Finally, dynamic updating of the background model is described.

A number of parameters are used during training and online classification of the codebook. The values of these parameters are based on experiments. A complete list of the parameters are given in Appendix E.1 on page 153.

5.4 Background Model

The background model consists of a codebook for each pixel in the frame. A codebook consists of a number of codewords. The codebooks does not necessarily contain the same number of codewords. The variation at a pixel determines the number of codewords. If the variation is small only a single codeword is created while a large variation results in more codewords.



(a) Illustration of a codeword. The codeword spans a volume in the RGB-space.

(b) A codeword and a pixel $\mathbf{x}_t$. If $\mathbf{x}_t$ is within the volume (the red cylinder) it activates the codeword.

FIGURE 5.8: A codeword in RGB-space.

A codeword is illustrated in Figure 5.8(a). Each codeword spans a volume in the RGB-space. The volume is shaped as a cylinder. A sampled pixel within the volume of a codeword is classified as background. When a pixel is in this volume, it is said to activate the codeword. Each codeword consists of the following elements:

$\mathbf{v}$: An RGB vector representing the mean color of the codeword.
It is defined as $\mathbf{v} = (v_R, v_G, v_B)^T$.

$\check{I}$: The minimum brightness or intensity of all pixels activating the codeword.

$\hat{I}$: The maximum brightness or intensity of all pixels activating the codeword.

f : The frequency with which the codeword has been activated.

λ : The maximum negative run-length (MNRL). It is the longest time interval a codeword has not been activated.

p : The first time the codeword was activated.

q : The last time the codeword was activated.

Using these elements, a codeword c is defined as

$$c = (\mathbf{v}, \check{I}, \hat{I}, f, \lambda, p, q) \tag{5.1}$$

The elements $\mathbf{v}$, $\check{I}$ and $\hat{I}$ are illustrated in Figure 5.8(a). The use of the remaining elements is elaborated in Section 5.5 describing the training of the background model.

As mentioned earlier, a pixel is classified as background if it is within the volume of a codeword. In Figure 5.8(b) it is shown how the pixel sampled at time t, called $\mathbf{x}_t$, is compared with a codeword. Figure 5.8(b) is an extension of Figure 5.8(a). The Codebook method separates the chromaticity and intensity when testing if $\mathbf{x}_t$ activates a codeword. According to [Kim et al., 2005] the variation of a background pixel is mainly caused by the change in illumination. Hence, background pixels are distributed as an elongated shape along the axis going toward the origin point of the RGB-space. This is the reason for doing a separation between the chromaticity and the brightness. The following section describes the color distortion metric used to determine the chromaticity difference. This is followed by how brightness changes are handled.

5.4.1 Color Distortion

The color distortion metric shown in Figure 5.8(b) is denoted δ and can be calculated as:

$$\delta = \sqrt{\|\mathbf{x}_t\|^2 - \left(\frac{\mathbf{x}_t \cdot \mathbf{v}_i}{\|\mathbf{v}_i\|}\right)^2} \tag{5.2}$$

In order to activate the codeword the color distortion must be smaller than some threshold ε:

$$\text{colordist}(\mathbf{x}_t, \mathbf{v}_i) = \delta \leq \varepsilon \tag{5.3}$$

ε is the radius of the cylinder-shaped codeword. Typical values for ε are between 8.0 and 15.0.

5.4.2 Brightness

Like the color distortion, the brightness of the pixel must also be within a given interval. The statistics $\hat{I}$ and $\check{I}$ are used to calculate the interval of allowed brightness. This interval is defined as $[I_{low}; I_{high}]$ and is shown in red in Figure 5.8(b). I_{low} and I_{high} is calculated using two parameters, α and β:

$$I_{low} = \alpha \cdot \hat{I} \quad , \quad I_{high} = \min\left(\beta \cdot \hat{I}, \frac{\check{I}}{\alpha}\right) \tag{5.4}$$

where $\alpha < 1$ and $\beta > 1$. According to [Kim et al., 2005], typical values for α is between 0.4 and 0.7 and β is between 1.1 and 1.5. The interval is able to limit the shadow level and highlight level. Hence, a human's shadow is classified as background whereas the human is classified as foreground. In order for the pixel $\mathbf{x}_t$ to activate the codeword, the following condition or function must return true:

$$\text{brightness}\left(I, (\hat{I}, \check{I})\right) = \begin{cases} \text{true} & \text{if } I_{low} \leq I \leq I_{high}, \\ \text{false} & \text{otherwise} \end{cases} \tag{5.5}$$

where $I = \|\mathbf{x}_t\| = \sqrt{{x_R}^2 + {x_G}^2 + {x_B}^2}$.

5.5 Training

In the following the construction of a single codebook is described. As mentioned earlier, each pixel in the frame has a corresponding codebook consisting of a number of codewords. The model is trained using a training sequence of N_{train} frames. The pixel sampled at time t in the training period, $\mathbf{x}_t$, is compared to the current codebook, $\mathcal{C}$, to determine which codeword c_m (if any) it matches. m is the matching index of the codeword in the codebook $\mathcal{C}$. The algorithm for constructing a codebook is shown in Listing 5.1.

LISTING 5.1: Algorithm for constructing a codebook

```
1  L = 0 , C = ∅
2  For t = 1 to N_train do
3      x_t = (x_R, x_G, x_B)^T , I = sqrt(x_R^2 + x_G^2 + x_B^2)
4      Seek matching codeword c_m in C based on the two conditions:
5          colordist(x_t, v_m) ≤ ε_1
6          brigthness(I, (Ǐ_m, Î_m)) = true
7      If C = ∅ or there is no match, then create a new codeword with index L:
8          L = L + 1
9          Create a new codeword c_L = (x_t, I, I, 1, t − 1, t, t) and add to C
10     Otherwise, update the matching codeword c_m:
11         v_m = ((f_m·v_{R,m} + x_R)/(f_m + 1) , (f_m·v_{G,m} + x_G)/(f_m + 1) , (f_m·v_{B,m} + x_B)/(f_m + 1))^T
12         c_m = (v_m , min(I, Ǐ_m) , max(I, Î_m) , f_m + 1 , max(t − q_m, λ_m) , p_m , t)
13 End for
```

The two conditions in line 5 and 6 in Listing 5.1 are Equation 5.3 and 5.5, respectively. The algorithm does not try to find the best matching codeword to $\mathbf{x}_t$. Instead it finds the first codeword that satisfies the two conditions. Based on experiments, [Kim et al., 2005] have shown that the order of the codewords in the codebook has no significant effect on the resulting detection sensitivity.

5.5.1 Temporal Filtering using Max Negative Run-Length (MNRL)

The resulting codebook from the algorithm in Listing 5.1 contains all the codewords that represent the training sequence. It may contain codewords modelling noise or moving foreground objects if some are present in the training sequence.

The purpose of the temporal filtering step is to remove all codewords not representing true background. This makes it possible to train with moving foreground objects and scene dynamics as described in Section 5.1.1. Dynamic background can be quasi-periodic, meaning that it appears almost periodically in the training sequence. A foreground object moving around in the scene during training is not quasi-periodic. For quasi-periodic appearance the MNRL (λ) is short while non-periodic appearance result in a long MNRL. The non-quasi-periodic codewords caused by moving foreground objects can be therefore be removed using the MNRL, while preserving dynamic background objects, like tree branches, that reappear quasi-periodically. The advantage of using the MNRL for temporal filtering contrast to other methods such as e.g. median filtering, is that it is independent of the frequency of codeword activations f.

To calculate the true MNRL, the training period for each codeword must be wrapped around:

$$\lambda_i = \max(N_{train} - q_i + p_i - 1 \ , \ \lambda_i) \tag{5.6}$$

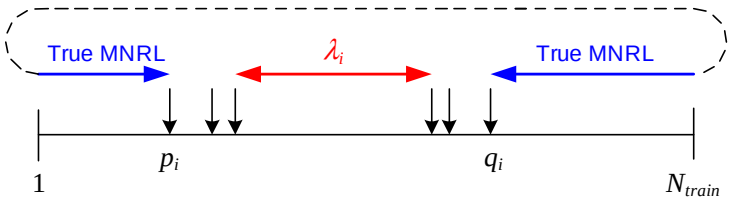


FIGURE 5.9: The true MNRL found using wrap around. A short vertical arrow illustrates each time the codeword has been activated. The MNRL obtained from the training is marked with a solid red line. The MNRL after wrap around is marked with a solid blue line.

The wrap around principle is shown in Figure 5.9. Codebooks are removed using a MNRL threshold, T_λ. A typical value for this threshold is $0.5 \cdot N_{train}$. The resulting codebook, $\mathcal{C}$, is given by:

$$\mathcal{C} = \{c_m | c_m \in \mathcal{C} \wedge \lambda_m \leq T_\lambda\} \tag{5.7}$$

5.6 Online Classification

After the training is completed the pixels in the frame are classified as either foreground or background. To classify the pixel sampled at time t, $\mathbf{x}_t$, the algorithm in Listing 5.2 is used.

LISTING 5.2: Algorithm for background subtraction

1 $\mathbf{x}_t = (x_R, x_G, x_B)^T$, $I = \sqrt{{x_R}^2 + {x_G}^2 + {x_B}^2}$
2 Seek matching codeword c_m in $\mathcal{C}$ based on the two conditions:
3 colordist$(\mathbf{x}_t, \mathbf{v}_m) \leq \varepsilon_2$
4 brigthness$(I, (\check{I}_m, \hat{I}_m))$ = true
5 If match found, then
6 Update c_m
7 Classify $\mathbf{x}_t$ as background
8 Otherwise
9 Classify $\mathbf{x}_t$ as foreground

A different threshold for the color distortion metric may be used when doing online classification. It is reasonable to only allow small variation in the color distortion during training, whereas the variation over longer periods of time would be greater due to large changes in illumination conditions. Hence, the value of ε_1 from Listing 5.1 may be smaller than that of ε_2 from Listing 5.2.

5.7 Dynamic Updating

In Listing 5.2 it is only stated that the codeword should be updated, but it is not shown how to do it. The activated codewords could be updated the same way they were during training (line 11 and 12 in Listing 5.1). However, this updating method is not able to handle gradual change in illumination. Instead an adaptive filter is used to update the codewords. Furthermore, the method described so far is not able to handle added static objects to the background after finished training. The background model can handle these added background objects by using layers. The adaptive updating and the use of layers are described next.

5.7.1 Adaptive Updating

According to [Kim et al., 2005] the variation of pixel values are different at different surfaces (shiny or muddy) and under different levels of illumination. This makes it reasonable to use a pixel-wise updating method as opposed to a frame-wise updating method. The following pixel-wise updating method is based on the update method from [Andersen and Corlin, 2005]. The mean color of the activated codeword can be updated using an adaptive filter. At time t the mean color vector is updated to be used at time $t+1$. This is achieved using the current mean color vector, $\mathbf{v}_{m,\,t}$, and the sampled pixel at time t, $\mathbf{x}_t$:

$$\mathbf{v}_{m,t+1} = (1-\gamma) \cdot \mathbf{v}_{m,t} + \gamma \cdot \mathbf{x}_t \quad , \quad 0 \leq \gamma \leq 1 \tag{5.8}$$

γ is the learning rate and is typically between 0.005 and 0.020.

The two intensity measures in the codeword are changed according to an estimation of the change in intensity. The change in intensity, ΔI, is estimated using half the intensity change in the mean color vector:

$$\Delta I = \frac{\|\mathbf{v}_{m,t+1}\| - \|\mathbf{v}_{m,t}\|}{2} \tag{5.9}$$

This estimate has in experiments shown to be better than $\Delta I = \|\mathbf{v}_{m,\,t+1}\| - \|\mathbf{v}_{m,\,t}\|$. The change in intensity from Equation 5.9 is added to $\check{I}_m$ and $\hat{I}_m$:

$$\check{I}_{m,t+1} = \check{I}_{m,t} + \Delta I \quad , \quad \hat{I}_{m,t+1} = \hat{I}_{m,t} + \Delta I \tag{5.10}$$

The distance between $\check{I}$ and $\hat{I}$ are not changed by adding $\Delta_{I,t}$. The remaining elements of the codeword are updated as done in the training.

A disadvantage of adaptive updating occurs when a pixel is falsely classified as background. In this situation the background model is updated to model a foreground object. When the foreground moves, the background model falsely classifies the corresponding pixel, which now represents true background, as foreground. To reduce the effect from this, only pixels that represent stable background is updated. When a pixel has been classified as background for the last N_{stable} frames it is said to be stable. N_{stable} is typically between 5 and 15 frames. A disadvantage of this approach is that a pixel may never be updated if it is wrongly classified as foreground every $N_{stable} - 1$ frames and background otherwise. However, keeping the value of N_{stable} low reduces this problem.

5.7.2 Updating using Layers

To allow the background model to adapt to addition and removal of objects, layers are introduced. When a foreground object remains stable for an on-line training period a new layer is added. Layers usually covers more than one pixel in the image, but each pixel is treated independently. A new layer is therefore merely addition of a new codeword in the codebook for a pixel. A new layer is only present in the pixels that the object covers. This meaning, that some pixels might contain one layer, while others contain another layer. Layers can also overlap, meaning that a codebook can contain parts of multiple layers at once. Lastly, the changes in the scene might not be permanent, and to delete a layer is merely a question of removing the codewords that describe the layer. If the inactive codewords are not deleted, the background model just keep on expanding and in the end, model something that is not true background.

An example of the concept of a layered background model updating is shown in Figure 5.10. A vehicle enters the scene and parks. At first, the vehicle is classified as foreground, but after holding still for a while it is absorbed into the background as a new layer.

To handle addition of new codewords, a type identifier is added to the codewords. By having different types of codewords, it is possible to discern whether a codeword describes the initial background obtained through training, a layer being trained online as background or a new layer of the background, obtained during online classification. The three applied denotations are listed and described in the following:

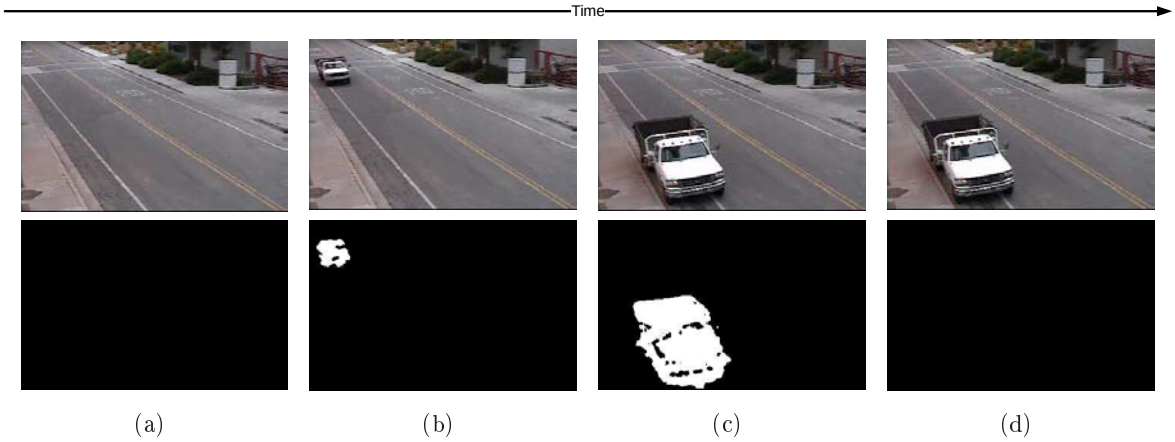


FIGURE 5.10: A series of pictures showing the concept of layered updating of the background model. The upper sequence is the input from the camera and the lower shows the output of background subtraction. (a) The background alone. (b) A vehicle enters the scene. (c) The car stops but is still detected. (d) The car is absorbed into the background model as a new layer and is thus not classified as foreground.

Permanent A codeword obtained during the initial training phase is denoted a permanent codeword. It is assumed that the background model obtained in training represents static background in the scene, and these codewords therefore always represent background. Permanent codewords are never deleted. It is chosen to never delete a permanent codeword in order to make sure that a codebook contains at least one codeword.

Training Training codewords are made whenever no permanent or non-permanent codeword is activated for a given pixel. The pixel is still classified as foreground, but the training codeword is in the codebook and undergoes online training, whenever it is activated. Online training is similar to the initial training.

Non-permanent When the online training period for a given codeword is over and if the codeword describes true background, it is changed to a non-permanent codeword and the pixel is classified as background. It has become a layer in the background model. Once a codeword has been deemed non-permanent, it is updated adaptively in the same way as the permanent codewords. A periodic cleanup removes inactive non-permanent codewords.

The methodology behind layered updating of the background model is shown as pseudocode in Listing 5.3. To clarify the contents, a few variables need defining:

$N_{train,\ online}$: The number of frames that a codeword must undergo online training before it can be considered for a change to a non-permanent codeword.

$T_{\lambda,\ online}$: The online MNRL threshold. The MNRL of the training codeword must be below this threshold before it can be changed to a non-permanent codeword.

$N_{expiration}$: The maximum allowed number of frames between activation of a non-permanent codeword. If more frames occur between activation, the codeword is removed from the codebook.

The first part of the procedure of background subtraction with dynamic update and layering is similar to the first part of the procedure listed in Listing 5.2. By applying layers to the already established adaptive update method, the background model is now capable of handling objects being added to and removed from the scene dynamically.

LISTING 5.3: Background subtraction with layered and dynamic background updating.

$\mathbf{x}_t = (x_R, x_G, x_B)^T$, $I = \sqrt{{x_R}^2 + {x_G}^2 + {x_B}^2}$
Seek matching codeword c_m in $\mathcal{C}$ based on the two conditions:
    colordist$(\mathbf{x}_t, \mathbf{v}_m) \leq \varepsilon_2$
    brigthness$(I, (\check{I}_m, \hat{I}_m)) =$ true
If match found, then
        If type is "permanent", then
            Update c_m adaptively
            Classify $\mathbf{x}_t$ as background
        If type is "non−permanent", then
            If c_m has expired $((t - q_m) > N_{expiration})$, then
                Remove c_m from $\mathcal{C}$
                Classify $\mathbf{x}_t$ as foreground
            Otherwise
                Update c_m adaptively
                Classify $\mathbf{x}_t$ as background
        If type is "training", then
            Update c_m as in training
            Classify $\mathbf{x}_t$ as foreground
            If c_m has completed online training time $((t - p_m) > N_{train,\ online})$, then
                If c_m represents background $(\lambda_m < T_{\lambda,\ online})$, then
                    Change type of c_m to "non−permanent"
                    Classify $\mathbf{x}_t$ as background
                Otherwise
                    Remove c_m from $\mathcal{C}$
                    Classify $\mathbf{x}_t$ as foreground
Otherwise
    Add new codeword with type "training" to $\mathcal{C}$
    Classify $\mathbf{x}_t$ as foreground

5.8 Shadow Suppression

A problem left unsolved by the motion segmentation described in the previous sections is moving shadows. The background subtraction based on the Codebook method is able to cope with shadows to some extend, but in order to handle stronger shadows like in Figure 5.11, a dedicated shadow suppression procedure must be applied.

The following presents an analysis of shadow suppression methods. This is followed by conceptual design and discusses the results obtained from experimental testing of the chosen method.

5.8.1 Analysis of Shadow Suppression

Shadow can appear as either cast shadow or self shadow [Javed and Shah, 2002]. A cast shadow is an object casting a shadow on the background, while self shadow is a part of the object and should not be removed. Results of cast shadows could be false shape, size and appearance of objects and merge of otherwise separate foreground objects. Furthermore, cast shadow can appear as an individual foreground object detached from the object causing it e.g. as depicted in Figure 5.11. Proper foreground segmentation should therefore exclude cast shadows. Though self shadow is not a problem by it self, it can make it very hard to separate the cast shadow from the object. A major challenge in suppressing shadow is also that moving cast shadow appears in connection with objects and that it often has many of the same characteristics as the objects, e.g. motion, size and shape as mentioned in Section 5.1.1.

FIGURE 5.11: Though the Codebook method is able to cope with shadows to some extend, it is not able to handle strong cast shadows from moving objects like in the example provided above. Because the allowed illumination variation in the codewords can not be lowered enough to include such shadows without corrupting the background model.

By reviewing a number of different shadow suppression approaches two are identified as the most interesting for solving the above mentioned challenges. Several other methods are presented in the review, which is available in Appendix A.2 on page 128. The two selected shadow suppression approaches are presented in the following:

Color segmentation The use of colors spaces that separate chromatic and intensity components, like YUV, HSV and normalized RGB, allows for detection of shadow pixels [Moeslund et al., 2006]. This is achievable by using a threshold scheme for the individual color space components, e.g. as in [Cucchiara et al., 2001] that apply thresholds in HSV color space based on knowledge of how shadow affects each color component.

Multi-view geometry By having multiple overlapping views of the same scene it is possible to exploit geometry and parallax, to separate objects from shadow. This is done in [Keck et al., 2006] which compares the color difference of pixels between views of an outdoor scene, using the homography of the ground plane. The method assumes that object pixels matched between views have a high color difference, while pixels that are background or shadow in both views have a low color difference.

The multi-view geometry approach [Keck et al., 2006] is interesting as it exploits the benefits of having multi-view information. However, since it is relatively new and untested, the information available about its performance is very limited. An implementation and test of this

approach is therefore made to evaluate its performance on the Matthews Lane dataset. The documentation of the multi-view geometry approach is available in Appendix B.1 on page 135. However, the experimental test shows that the method requires very precise homography mapping which can not be guaranteed due to the inaccuracies described in Section 4.4 on page 33. Furthermore, the test shows that the method has a significant chance of falsely removing object pixels as shadow. The multi-view geometry approach is therefore discarded. Therefore, it is chosen to use a color segmentation approach based on [Cucchiara et al., 2001], as it seems to be the most proven and tested method based on the shadow suppression survey in [Prati et al., 2003]. Since the method is based solely on color segmentation, it is not optimal for separating cast shadow from self shadow. Therefore, it is decided to utilize multi-view information by only considering the footage region as potential shadow, since the footage region in many cases only contains the cast shadow and not the object.

5.8.2 Conceptual Design

After online classification shadow suppression is performed to remove shadow pixels falsely classified as foreground. Figure 5.12 extend the conceptual design in Figure 5.7 to show the details of the integration of the shadow suppression in the foreground segmentation module. The color segmentation method utilizes a background image generated from the background model, and the footage region is obtained using multi-view information from another foreground segmentation process. This is depicted by the dotted lines in the figure.

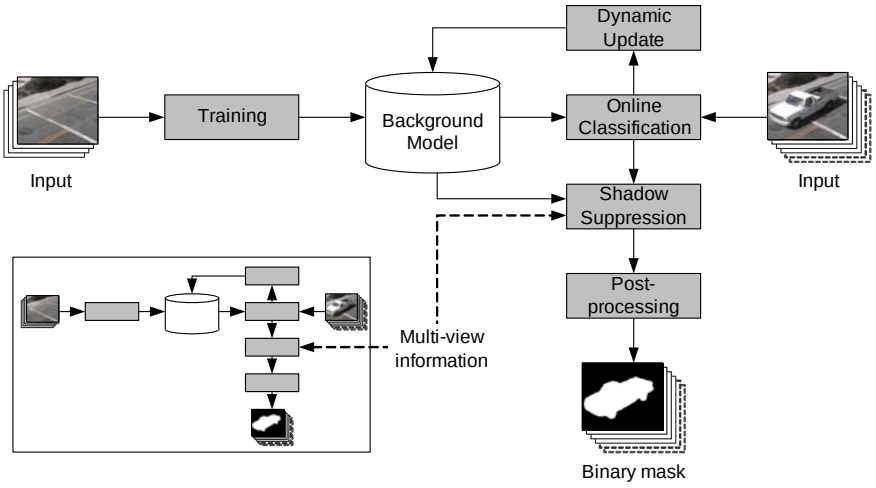


FIGURE 5.12: Conceptual design of the foreground segmentation module.

The HSV color segmentation method based on [Cucchiara et al., 2001] is explained in the following. Following this, the utilization of multi-view information to for improving the method is explained.

5.8.3 HSV Color Segmentation

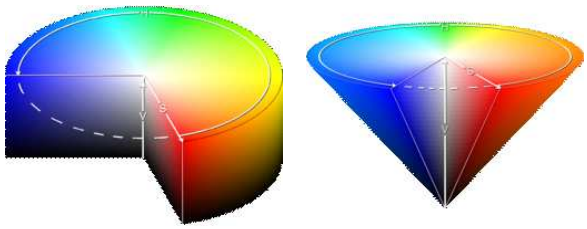
The color segmentation method compares the individual H, S and V components of the current frame $I(x, y)$ pixel-wise, with a background image $B(x, y)$. The background image is generated from the Codebook model by creating an image with the value of the most recently

activated codeword for each pixel. By thresholding each component individually it is then determined if the tested pixel is shadow or not. The three conditions for generating the shadow mask $S(x, y)$ are [Cucchiara et al., 2001]:

$$S(x,y) = \begin{cases} 1 & \text{if } \alpha_V \leq \frac{I^V(x,y)}{B^V(x,y)} \leq \beta_V \\ & \wedge (I^S(x,y) - B^S(x,y)) \leq \tau_S \\ & \wedge |I^H(x,y) - B^H(x,y)| \leq \tau_H \\ 0 & \text{Otherwise} \end{cases} \tag{5.11}$$

The first condition compares the intensity. The brightness thresholds α_V and β_V determines which interval is considered potential shadow. By choosing a β_V less than one, it is possible to avoid noise that causes the intensity to drop slightly. α_V is a lower limit for how much the intensity can drop and still be considered potential shadow. The lower limit is necessary since large drops in intensity should not be considered potential shadow due to dark foreground objects. The second condition is based on the assumption that the saturation component is either lowered or remains constant in shadows. The final condition is based on the assumption that hue remains relatively unchanged by shadow, since shadow cause little or no variation in hue. All three HSV components are normalized to be in the range 0 to 1.

From initial work with color conversion it was learned that the saturation component of the HSV color space is unstable at low intensities which causes high saturation values in shadow regions. This is a problem since it conflicts with the assumption that the saturation either remains constant or is lowered in shadow regions. The noise in saturation at low intensities is due to the cameras and compression which causes instable chromatic components [Blauensteiner et al., 2006]. The instability in the saturation component is so pronounced in dark regions due to normalization of the saturation by the brightness when converting from RGB to HSV [Blauensteiner et al., 2006]. This normalization causes the HSV space to take the shape of a cylinder, as depicted in Figure 5.13(a), while a non-normalized representation of the HSV space is cone shaped as in Figure 5.13(b).



(a) (b)

FIGURE 5.13: Two representations of the HSV color space. (a) Cylinder shaped HSV color space. (b) Cone shaped HSV space.

The normalization of the saturation is often an implicit part of the conversion from RGB to HSV [Blauensteiner et al., 2006], e.g. as in OpenCV's color conversion. A color conversion algorithm without normalization by the brightness is therefore used. As a result, a more stable saturation component is achieved.

5.8.4 Adding Multi-View Information

To reduce the number of pixels falsely removed, it is possible to limit the search for shadow pixels to foreground pixels that overlap with foreground from the other view as depicted by the blue footage regions in Figure 5.14(b). To ensure that imprecisions in the planar homography mapping does not cause a too small or missing overlap the foreground masks are dilated before warping. The full footage region can not be removed as shadow without the HSV color segmentation, since occlusion of the shadow would result in object parts being wrongly segmented as shadow.

(a) Test setup

(b) Footage region

FIGURE 5.14: Example of a moving object with a cast shadow. (a) Test setup. Left: first view. Right: second view. (b) Green: warped mask from the other view. Blue: overlap with foreground mask and warped mask.

Shadow outside the shared region of the two views is not shadow suppressed in order to avoid the risk of removing too much of the objects. Since objects outside the shared region often are very small, shadow can actually help in detecting small objects.

5.8.5 Results and Discussion of Shadow Suppression

In the following a discussion of the results obtained from experimental testing of the color segmentation method is presented. Experimental tests are carried out on sequences from the Matthews Lane dataset, since this is the only dataset with significant shadows. An example of a sequence is shown in Figure 5.14(a). An example of the output without multi-view information added is given in Figure 5.15.

To further reduce the number of pixels falsely removed by the shadow suppression, multi-view information is applied. This improves the result as depicted in Figure 5.16. A disadvantage of the method is if a shadow is only segmented as foreground in one view, no shadow is removed.

The results of the shadow suppression on short sequences with adjusted thresholds is very good, as depicted in Figure 5.16. However, for long sequences with illumination variation it is

FIGURE 5.15: Result of shadow suppression with non-normalized saturation. The yellow color is the foreground segmented by the Codebook method, and the red is the portion of the foreground classified as shadow. The frames match those in Figure 5.14(a). The thresholds for both views are set to: $\alpha_V = 0.25$ $\beta_V = 0.97$ $\tau_S = 0.20$ $\tau_H = 0.20$.

FIGURE 5.16: Result of shadow suppression by only considering the footage region potential shadow

very difficult to set appropriate thresholds that allows segmentation of shadows and preserves object foreground, e.g. as shown in Figure 5.17 where more object pixels than shadow pixels are removed. A reason why the thresholds are difficult to adjust is that roads and sidewalks have little color information and people are most often dressed in similar colors. Also, in scenes with strong shadows, people often appear dark (self shadow) regardless of their actual color. This is especially a problem at morning and evening time where the sun illuminates the scene from a low angle. All together, the results is that the shadow suppression thresholds becomes very sensitive to changes in illumination as it changes the color and brightness of both the scene and objects.

FIGURE 5.17: Result of shadow suppression in the narrow view configuration of Matthews Lane.

5.8.6 Summary of Shadow Suppression

The following lists positive and negative sides of using the color segmentation method for shadow suppression.

Pros:

- Represents chromatic and illumination components separately for better color segmentation.
- The threshold scheme accounts for the properties of shadow in each color component.
- The number of falsely removed pixels are reduced using multi-view information.

Cons:

- The shadow suppression thresholds are sensitive to illumination, and performance degrade over long sequences.
- Objects with large self shadow or resemblance with the background are likely to be misclassified.
- If a shadow is only segmented as foreground in one view no shadow is removed.

5.8.7 Partial Conclusion for Shadow Suppression

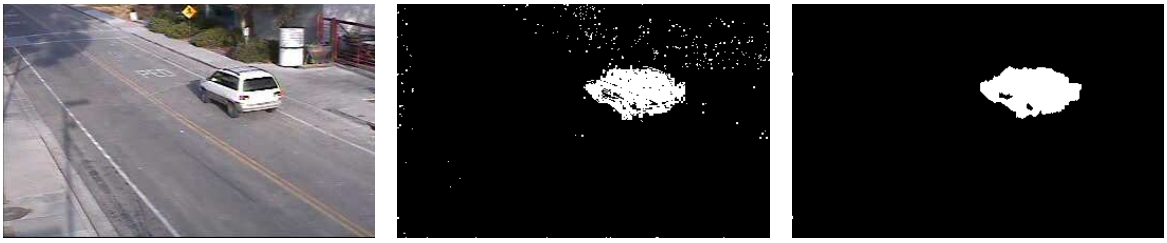
The performance of the shadow suppression is good for short sequences and for scenes with little variation. However, due to the length of the Matthews Lane dataset, the performance in this dataset is low and does not significantly improve the foreground segmentation result. Shadow suppression is therefore not used in any of the forthcoming tests. Though the performance is low for the Matthews Lane dataset, the method could still perform well for shorter sequences and with less illumination changes and background camouflage.

5.9 Post-processing

The output from the foreground segmentation module is a binary image. Some single pixels may be falsely classified as foreground due to noise. Because there is no surrounding foreground pixels these falsely classified pixels they can be removed using a $n_{median} \times n_{median}$ median filter. A 3×3 or a 5×5 median filter is sufficient to remove the falsely classified foreground pixels. An example of applying a 5×5 median filter on the output from the Codebook method is shown in Figure 5.18. The final output from the foreground segmentation module is the median filtered binary image.

5.10 Foreground Segmentation Test

As mentioned, this test is based on foreground segmentation without shadow suppression. The system is intended to run autonomously over long time periods. This means that the foreground segmentation must be able to handle a number of challenges as described under



(a) Input frame (b) Codebook output (c) Median filtered

FIGURE 5.18: The effect of using a median filter. The applied filter is a 5×5 median filter.

general considerations in Section 5.1.1. During night time the foreground segmentation is not expected to register objects since the cameras do not have night-vision capabilities and the street lights are weak. It is however expected to automatically recover to its normal performance at dawn.

The test is based on a three day long sequence from the Matthews Lane dataset. From this sequence 16 hours are selected and subsampled for ground truth labelling. The samples and ground truth can be found on the DVD (◎ */tests/foreground_segmentation_test/*). The following presents the setup used in the test. This is followed by a description of the applied metrics and ground truth labelling. Then, the results are presented and ended with a conclusion and summary.

5.10.1 Test Setup

The Matthews Lane dataset contains more than 160 hours of video recorded in an unconstrained environment covering a wide variety of conditions. Of the 160 hours, the longest continuous sequence of 78 hours is selected for the this test. Foreground segmentation is performed on all 78 hours by continuously updating the background model. Additionally, for ground truth labelling the last 16 hours of the 78 hour sequence are selected. In this way, the Codebook foreground segmentation is running for 62 hours before the ground truth testing, making the result more realistic in the context of doing long term surveillance. The 16 hour sequence runs from 5:00 to 21:00 and covers various lighting conditions from cloudy weather to sunshine. Also, the recordings are made on a weekday which contains a significant amount of traffic and activity, including a significant number of scene changes. The cameras are configured to adjust white balance and gain automatically, meaning that this can be a cause of artificial illumination changes. However, the automated settings are needed to enable recording over long periods of time, since this helps to compensate for the natural illumination changes during the day. Examples of variation in the test data are shown in Figure 5.19. For more detail about the dataset used in this test, see Appendix D.1 on page 147.

Some practical issues influence the video quality; the lamp posts on which the cameras are mounted can sway in the wind causing small camera movements, and vibration from near by construction work could cause similar effects. Furthermore, due to the large amount of data, the video sequences have been recorded using a lossy DivX6.5 format at highest quality setting and 780 kbps bitrate. This causes some compression noise in the images.

(a) Time: 5:32 (b) Time: 7:38 (c) Time: 11:24

(d) Time: 16:26 (e) Time: 17:41 (f) Time: 20:39

FIGURE 5.19: Test data scene examples. (a) Morning scene (Noise is often appears more significant in dark scenes like these). (b) Vehicle background camouflage (Parts of the truck body and the gray car behind it). (c) Heavy traffic and object movement. (d) Person background camouflage (Two persons on the left and parts of the two persons on the right). (e) Illumination changes causing cast shadow from trees and lamp post (f) Night scene and car headlights.

To prove the system's ability to produce consistent results in an unconstrained environment over long periods of time, the foreground segmentation is configured with parameters that are not fitted to a specific sequence. The parameters are therefore not selected with the goal of obtaining the best possible false rejection rate and false acceptance rate, and thus gives a more realistic impression of the system's ability to do foreground segmentation over long time.

5.10.2 Evaluation Metrics

For evaluation of the performance the following two metrics are used: false rejection rate (FRR) and false acceptance rate (FAR). A rejected pixel is a pixel not classified as foreground, and an accepted pixel is a pixel classified as foreground. The metrics are computed as:

$$FRR = \frac{\text{Falsely rejected pixels}}{\text{Number of foreground pixels}} \quad FAR = \frac{\text{Falsely accepted pixels}}{\text{Number of background pixels}} \tag{5.12}$$

Note that for empty frames with no ground truth foreground, FRR is undefined.

Humans and vehicles are marked separately during the ground truth labelling, meaning that it is possible to evaluate their FRR separately. Since noise is neither human or vehicle, it is not possible to evaluate the FAR separately for the two objects types.

5.10.3 Ground Truth Labelling

For ground truth labelling the 16 hours are sub-sampled every 5000 frames (roughly 5 minutes). This corresponds to 174 frames per view. An example of a sampled frame for both cameras is shown in Figure 5.20.

(a) View 1 (b) View 2

FIGURE 5.20: Example of a scene with both cars and humans. Time: 9:20.

For the labelling some assumptions are made:

- All shadows except self shadow are considered background.
- Vehicles holding still for more than 500 frames are background.
- Humans standing nearly still for more than 500 frames are foreground.
- Objects moved by humans are marked as human foreground in the ground truth (including car doors).

As a part of the ground truth labelling a "don't care" zone is defined around objects, in order to compensate for inaccuracies in the hand labelling. All ground truth images are mean filter with a 3×3 kernel. In this way, blurred edges are created around the objects. This edge boundary of blurred pixels defines the "don't care" zone around the foreground objects. Foreground segmentation by the system within this don't care zone is not considered in the calculation of the FRR and FAR. Figure 5.21 depicts the ground truth and corresponding don't care zones for the right view of Figure 5.20.

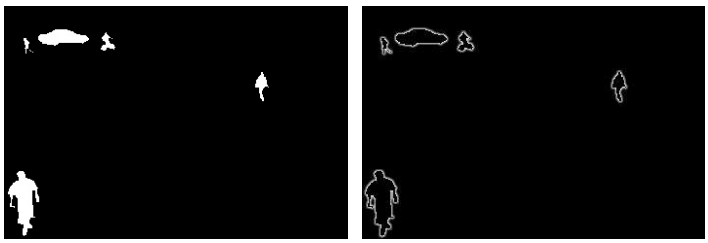
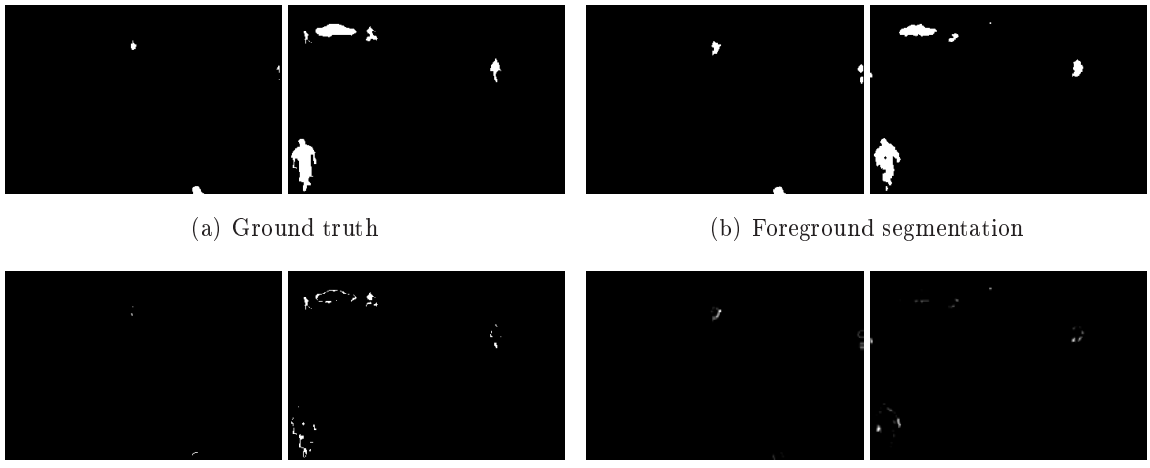


FIGURE 5.21: Left: ground truth for both humans and vehicles. Right: "don't care" zone surrounding the objects.

5.10.4 Comparison

Examples of the comparison for the scene in Figure 5.20 are shown in Figure 5.22. Figure 5.22(a) shows the ground truth for both cars and vehicles. Figure 5.22(b) shows the output

produced by the Codebook foreground segmentation. Figure 5.22(c) shows the falsely rejected pixels used to calculate the false rejection rate (FRR) and Figure 5.22(d) shows the falsely accepted pixels used to calculate the false acceptance rate (FAR).



(a) Ground truth (b) Foreground segmentation

(c) False rejections (d) False acceptances

FIGURE 5.22: (a) True foreground for both persons and vehicles. (b) Foreground segmented by the Codebook foreground segmentation. (c) Falsely rejected pixels for both views. FRR for view 1: 7.6% (Both), N/A (Cars), 7.6% (Persons). FRR for view 2: 20.7% (Both), 17.9% (Cars), 21.9% (Persons). (d) Falsely accepted pixels for both views. FAR view 1: 0.190%. FAR view 2: 0.308%.

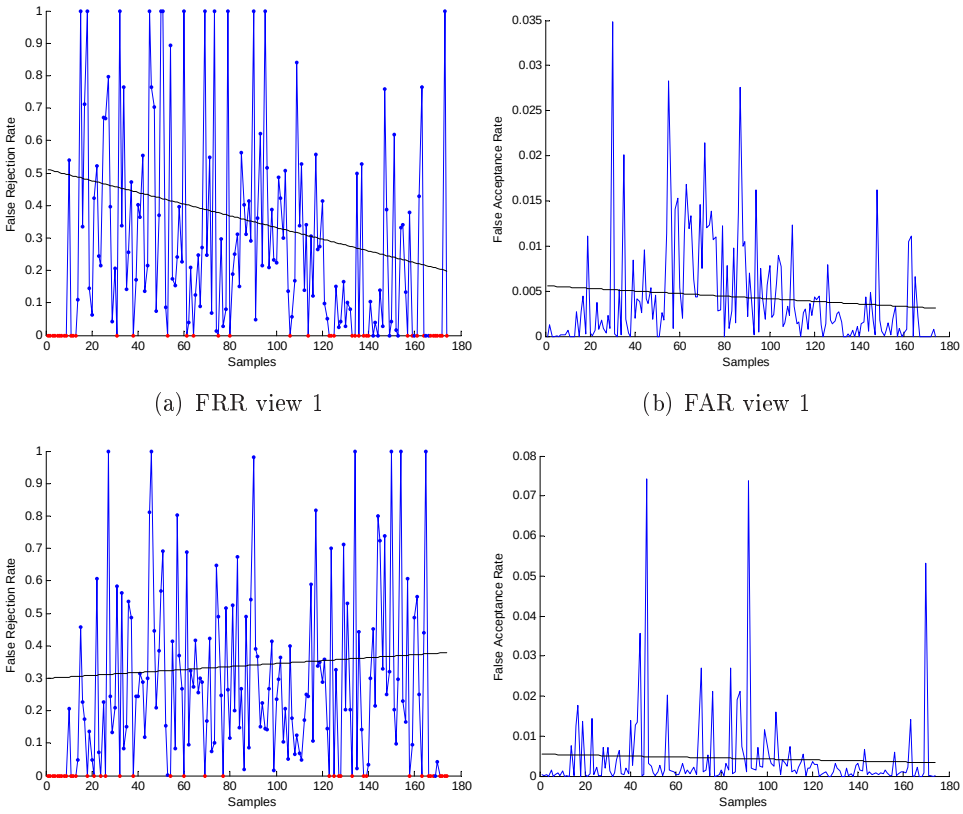
5.10.5 Results

The results of the 16 hours test for view 1 are given in Figure 5.23(a) and Figure 5.23(b). View 2 are given in Figure 5.23(c) and Figure 5.23(d).

The graphs show that the performance of the foreground segmentation is relatively consistent during the 16 hours test. As the fitted line indicate, no real tendency for an increasing of decreasing performance during period can be identified. However, from the graphs it is clear that the performance of the background subtraction is low. As can be seen from Figure 5.23(a) and Figure 5.23(c), there is a number of cases where the FRR is 100 percent. However, a significant reason for this is the metric used. Since the FRR is calculated by dividing with the number of ground truth foreground pixels, it is very sensitive in cases where there is a very limited number of foreground pixels. Scenes with few foreground pixels often occur when objects are far away from the camera.

Therefore, the total result is calculated for all frames of each view to avoid misleading results due to frames with very few foreground pixels. This is done by summarizing the number of falsely rejected pixels, falsely accepted pixels, true background pixels and true foreground pixels using the same formulas in Equation 5.12. The results of calculating the FRR and FAR for the total sequence are show in Table 5.1.

These results show very similar performance for both views, though with a relatively high FRR of 29.2 percent. The most frequent cause for this is objects with small size and background



(a) FRR view 1

(b) FAR view 1

(c) FRR view 2

(d) FAR view 2

FIGURE 5.23: The foreground segmentation results over time for each view. Results are shown for both views and include both humans and vehicles. The black line is fitted to the data using least sum of squares. Each sample corresponds to 5000 frames or roughly 5 minutes.

camouflage. An example of both is shown in Figure 5.24. By looking at the RGB values of the foreground objects it is seen that the color of their clothes is very similar to the background; only the intensity sets them apart which makes them very hard to segment correctly.

FIGURE 5.24: Example of objects that are difficult to segment. The green box shows a cyclist which is very hard to see due to his size. Two typical examples of background camouflage of persons with background colored clothes is also shown (yellow and red box).

By looking at the foreground segmentation results from an object-wise point-of-view rather

	View 1	View 2
Cars FRR	30.3%	29.6%
Persons FRR	25.3%	26.7%
Both FRR	**29.2%**	**29.2%**
Cars FAR	0.68%	0.64%
Persons FAR	1.26%	1.70%
Both FAR	**0.41%**	**0.42%**

TABLE 5.1: Total false rejection rate and false acceptance rate for the foreground segmentation test.

than pixel-wise, it is seen that the majority of the objects very often are segmented correct. This is shown in Figure 5.22; even though view 2 in Figure 5.22 has a FRR of 20.7 percent the four largest objects are clearly visible in the segmentation results, and the falsely rejected pixels are mainly caused by misclassification around the edges. It is therefore very likely that the forthcoming modules can perform well in spite of the poor FRR.

The following list some general observations made from the entire 78 hours of the recordings and corresponding segmented foreground mask:

- The performance of the Codebook method is consistent over time, meaning that there is no visible difference between the outputs produced over the three days.
- During night time the foreground segmentation also performs well with little or no noise, though the initial appearance of the scene changed completely. Also the foreground segmentation often detects objects correctly, though they are only lit by street lights. However, cars with headlights are not segmented correctly.
- Though the performance can decrease slightly in twilight there is no problems with recovering or adapting during sun rise and sun down.
- In narrow places where people walk very frequently at certain times of a day, e.g. at a pedestrian crossing, it is hard to avoid that foreground objects are being added as a layer into the background, since humans often wear similar colored clothes and appear in an almost quasi-periodical manner.
- Some static objects like parked vehicles with reflective surfaces could sometimes be very difficult to absorb fully into the background.

5.10.6 Partial Conclusion

The FRR of the foreground segmentation is high when compared to other published work. However it should be seen in the light that the test is performed over a significantly longer time period than what has previously been done and with a very dynamic and changing scene. Accordingly, the test gives a very realistic impression of how good foreground segmentation that can be expected in a 24/7 surveillance application given the current setup and scene complexity. Also, seen from an object segmentation point-of-view, the results are considered acceptable as basis for the later modules.

5.11 Summary

The general findings in the chapter regarding foreground segmentation are:

- In the analysis, it is decided to use background subtraction with added shadow suppression for producing a mask containing foreground objects.
- The background model is constructed and maintained by use of the Codebook method.
- A shadow suppression method based on an HSV threshold scheme with added multi-view information is selected.
- The shadow suppression method does not perform well over longer periods of time and with significant changes in scene illumination. It is therefore not applied in the foreground segmentation test.
- Post-processing is applied to lower the amount of false positives and false negatives in foreground classification.
- The foreground segmentation is capable of producing consistent segmentation results during a 78 hours long test sequence.
- The foreground segmentation test shows a false rejection rate of 29.2 percent and a false acceptance rate of 0.42 percent in a 16 hour test sequence.
- The false rejection rate (FRR) of the foreground segmentation is not as good as in other work, but the scenes are more challenging and the sequences are much longer than what is typically presented.
- Seen from an object segmentation point-of-view the results of the foreground segmentation is considered acceptable as basis for the later modules.

Chapter 6

Single View Tracking

The single view tracking module tracks the objects, which is used by the later modules to do object correspondence and assessing the danger level of the situation using view invariant analysis. This chapter analyzes existing object classification and tracking methods. This is followed by the design of the tracker including a description of probabilistic appearance models, which is used to resolve occlusion situations.

6.1 Analysis

The single view tracking module serves two purposes. The first is to classify moving objects as a specific type, e.g. "human" and "vehicle". Secondly, to track each single moving object through its movement in the scene. Occlusion of objects are natural in traffic scenes, and the module must be able to resolve this issue. Because the principal axis method [Hu et al., 2006] is used to do object matching in the correspondence module, single view tracking is required.

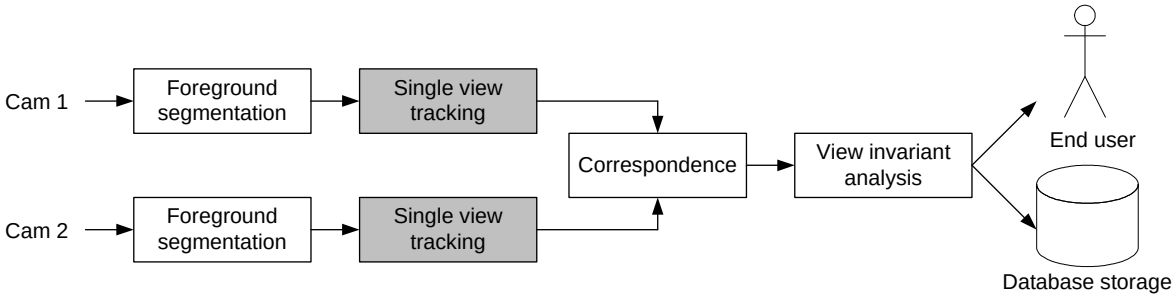


FIGURE 6.1: Overview of the system highlighting the single view tracking module.

The single view tracking module is highlighted in the system overview in Figure 6.1. The input to the module is the binary mask from the foreground segmentation module and the current color frame obtained from a single camera. The output is a track list where each tracked object is represented using a unique ID. This track list is given to the forthcoming module, which is the correspondence module.

The following analysis for the single view tracking module starts with general considerations for the module. This is followed by a review of work related to the field of object classification. Lastly, the tracking methods are analyzed at different levels.

6.1.1 General Considerations

In the problem analysis in Section 2.4 on page 17 a number of computer vision challenges are listed, some of which influence the single view tracking. The following emphasizes the main challenges related to the single view tracking module.

Occlusion The most important feature of a tracking module may be its ability to handle occlusion situations. Occlusion can be divided into three categories; inter-object occlusion is occlusion between moving objects, self occlusion is when an object occludes a part of its own object body and static occlusion is when a static background object occludes a moving object.

Foreground segmentation issues As shown in the foreground segmentation test in Section 5.10 on page 56, the segmentation is sometimes faulty. A relatively large area might wrongly be classified as foreground due to noise, a large portion of an object might be missing or an object might be divided into multiple disjoint blobs. An example of the latter is shown in Figure 6.2. These issues must be addressed by the module.

FIGURE 6.2: Example of foreground segmentation errors, where the human objects are divided into multiple disjoint blobs.

Track initialization Many tracking algorithms perform well during occlusion situations using some model. However, often the models require correct initialization which is difficult in an unconstrained environment where objects enter and leave at the same time. Human objects enter as a group and a vehicle object cover a large portion of the field of view even though the object has not fully entered the scene. Therefore, the module should utilize measures to reduce the problem of incorrect track initialization.

Data association Associating the tracks with the new measured positions of the objects is called data association. When using a low frame rate, the displacement of the moving objects might be quite large. Therefore, the problem of data association could be reduced using a high frame rate. 15 fps is the lowest frame rate in the available datasets (See Appendix D on page 147 for a description of the datasets), but this is a sufficient frame rate to ensure overlap of the objects (both human and vehicle) from frame to frame.

These general considerations are used throughout the analysis and design of the module.

6.1.2 Object Classification Methods

To have the best understanding of the object behavior, the correct classification of the moving object is needed. The classification must be able to discriminate between several classes depending on how fine a classification is wanted. A human object could e.g. be walking, cycling or skateboarding. Some classes could be hard to discriminate, e.g. a person walking appears quite similar to a person on skateboard. A more coarse classification is thus easier to achieve. Furthermore, since it is expected that a group might enter the scene, the classification should be able to classify this object as a group; or at least as a human object rather than a vehicle. In addition, background clutter should be rejected at this stage.

According to [Hu et al., 2004b] two main approaches for classifying moving objects exist:

Shape-based classification Descriptors of shape information are used. This could be based on points, boxes, silhouettes and blobs. Specific features could be image blob dispersedness, aspect ratio of bounding box and area of the blob.

Motion-based classification Humans are non-rigid articulated objects, and their motion shows a periodic property. This is utilized in motion-based classification by analysis of the presence of this periodicity to separate human motion from e.g. vehicle motion.

Discussion and Choice of Object Classification Method

Both approaches are able to solve the classification task. The motion-based classification requires some temporal amount of motion data before a classification is possible. This results in a delay when determining the type of an entering object. As an example, [Cutler and Davis, 2000] uses motion-based classification, and their method introduces a delay of one second. Furthermore, it is not clear how a motion-based classification approach performs when classifying a group of people. This is the main reason for choosing a shape-based classification approach.

6.1.3 Tracking Methods

To give an overview of the tracking methods, the structure used in [Yilmaz et al., 2006] is applied. [Yilmaz et al., 2006] is a survey focusing only on the object tracking issue. In the following, the main points of the survey are given. A more detailed description based on the survey is given in Appendix A.3 on page 130. A tracking method can be divided into three levels, and these are depicted along with the common choices in Figure 6.3. A bottom-up approach is followed in describing the issues related to tracking. The first issue is to find a suitable representation of the object. Afterwards the image features used to track the object is presented. At the top level, the general tracking approaches are described. Based on this analysis, a discussion and choice of tracking method follows.

Object Representation

The object representation is the lowest level in tracking and can be divided into shape and appearance of objects. A strong relationship exists between the object representation and the

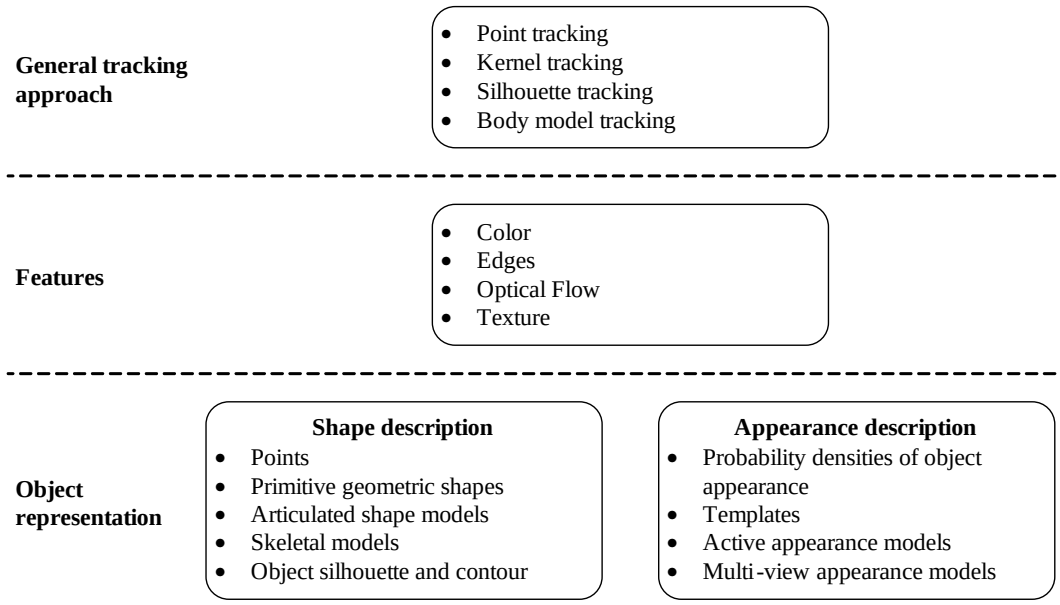


FIGURE 6.3: Overview of tracking methods. The three tracking levels are shown. For each level the common approaches are listed inside a box.

general tracking approach. The possible shape descriptions are listed in the bottom left box in Figure 6.3 and common shape representations are illustrated in Figure 6.4.

As with the shape of the objects a number of approaches exist to represent the appearance of the objects. Some representations combine both the shape and appearance of the objects. Common appearance representations are listed in the bottom right box in Figure 6.3 and for more details see Appendix A.3 on page 130.

Feature Selection

Selecting the features to use in the tracking of objects is closely related to the object representation; e.g. color features are used in probability densities of object appearance and object

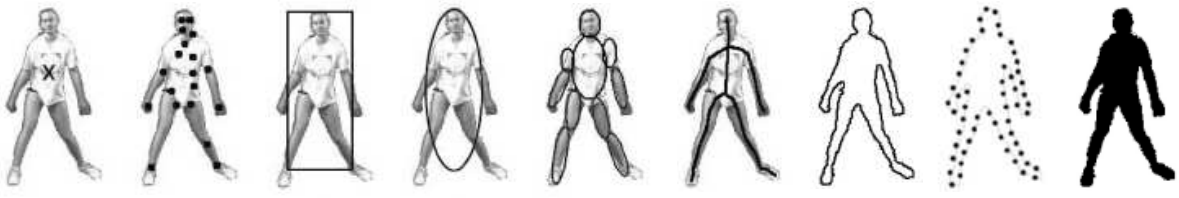
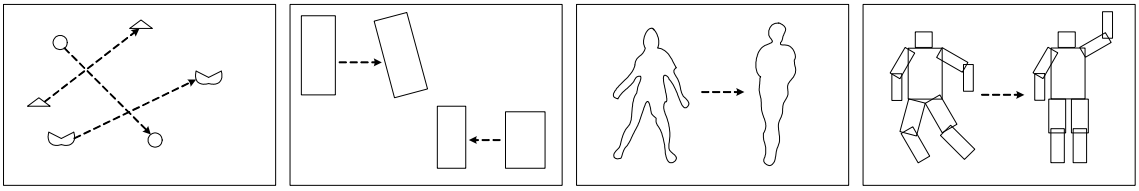


FIGURE 6.4: Object representations. (a) Points: Centroid. (b) Points: Multiple points. (c) Primitive geometric shapes: Bounding box. (d) Primitive geometric shape: Bounding ellipse. (e) Articulated shape models: Part-based multiple patches. (f) Skeletal models: Object skeleton. (g) Object silhouette and contour: Complete object contour. (h) Object silhouette and contour: Control points on object contour. (i) Object silhouette and contour: Object silhouette. Figure from [Yilmaz et al., 2006].



(a) (b) (c) (d)

FIGURE 6.5: General tracking approaches. (a) Point tracking with multiple point association. (b) Kernel tracking using a rectangular representation. (c) Silhouette tracking illustrated by a contour representation. (d) Body model tracking using rectangular body parts.

edges are used as features for contour-based representations. It is desired to use features that make objects easy to distinguish in the feature space. Often features are combined to track the objects. Some common visual features are listed in the middle box in Figure 6.3.

General Tracking Approach

In the following, four main tracking categories are mentioned. They are highly dependent on the chosen object representation and feature selection, and thus the applicability of the tracking method depends on which objects are to be tracked. It is a requirement that the object representation is valid for both non-rigid human objects and rigid vehicle objects. The four approaches are exemplified in Figure 6.5. The four general approaches are shortly described in the following. For more details see Appendix A.3 on page 130.

Point tracking The detected objects are represented using a single or multiple points. The association between points in consecutive frames is often based on the previous position and motion of the object. Point tracking is simple and fast and well suited for small objects. However, point tracking has problems with occlusion of the tracked points, misdetection, entries and exits of objects.

Kernel tracking In this context, kernel refers to the object shape and appearance; e.g. a rectangular shape and a color histogram describing the appearance within the rectangle. The most occurring appearance representations are in this context templates, probability densities and multi-view appearance models. The pros are the use of simple geometric shapes, and that kernel tracking is applicable for both rigid and non-rigid objects. However, pose estimation is not possible with this representation.

Silhouette tracking Silhouette tracking methods usually uses appearance densities and shape models for representing the information contained within the object boundary. Often edges are applied as the feature. Depending on the chosen object model, either shape matching or contour evolution is used for tracking the object. Trackers based on silhouettes provide an accurate shape description of the object. However, these methods often need a training phase and are highly sensitive to initialization.

Body model tracking In body model tracking the "body" of the object is modelled. The object is tracked by projecting the model into the image plane and compared with the image

data. The models are constructed offline and incorporate prior knowledge of the object body. Body model tracking natively holds the object pose and obtains good results during occlusion (including self-occlusion). However, this tracking method is computationally expensive, and rigid and non-rigid objects differs significantly [Hu et al., 2004b].

Discussion and Choice of Tracking Method

As mentioned in the analysis above, there is a strong relationship between the three tracking levels. In the following discussion, the focus is on the general tracking methods based on their pros and cons as listed above. Based on the choice of general tracking method the object representation and features follows.

The body model tracking methods are computationally expensive and two different models are required; one for vehicles and one for humans. The silhouette tracking suffers from initialization difficulties [Hu et al., 2004b], and in the given problem area it is expected that objects enter the scene occluding each other (e.g. moving in a group in the human case). Even though they offer good description of the object's pose, the aforementioned issues are the main reason for not choosing body model tracking or silhouette tracking.

Compared to point tracking, kernel tracking appears more robust, because point tracking has problems with occlusions, misdetection, entries and exits of objects. Therefore, it is chosen to use kernel tracking.

According to [Yilmaz et al., 2006] color is the most popular feature in kernel tracking. Many kernel tracking algorithms uses a primitive geometric shape combined with appearance probability densities or appearance templates based on color features. Examples are [Senior et al., 2006, Roth et al., 2005, Xu and Puig, 2005, Cucchiara et al., 2004, Hu et al., 2004a, Senior, 2002, Pérez et al., 2002, Khan and Shah, 2000, McKenna et al., 2000]. Special interest has been put on [Senior et al., 2006] because this presents tracking results on the PETS 2001 dataset, which is an outdoor scene containing vehicles, pedestrians and bicyclists. Given the reliable tracking results presented by this method, [Senior et al., 2006] is chosen as the basis for tracking the objects in a single view.

6.1.4 Analysis Summary

The single view tracking module consists of two tasks, which are object classification and object tracking. The object classification method is based on shape-based classification. Out of the many different tracking approaches it is decided to use a kernel tracking approach. Furthermore, it is chosen to base the tracker on [Senior et al., 2006].

6.2 Conceptual Design

The conceptual design of the single view tracking module is shown in Figure 6.6. The input to the module is the color input frame and the binary mask from the foreground segmentation module. The output is a list of the tracks, which is used as input to the following correspondence module.

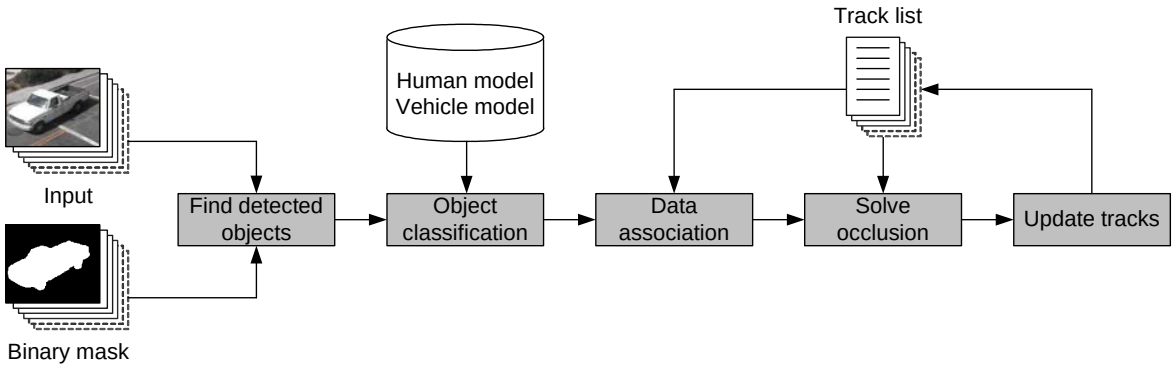


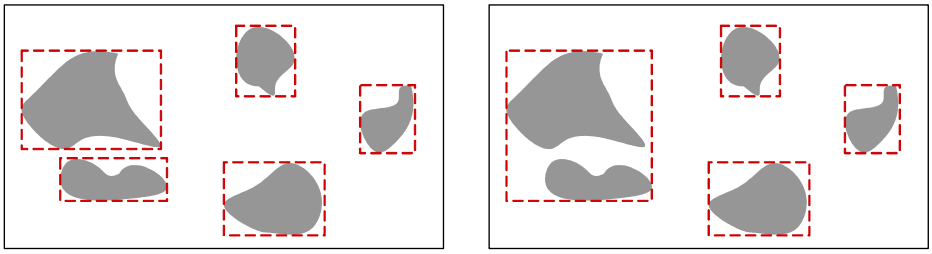
FIGURE 6.6: Conceptual design of the single view tracking module.

The remaining chapter is structured based on the conceptual design. As shown in Figure 6.6, the first step is to find detected objects by performing connected component analysis on the binary mask. The detected objects are classified as either a human object or a vehicle object. Detected objects are associated to the existing tracks. In case of inter-object occlusion this must be resolve. Each track contains a probabilistic appearance model [Senior et al., 2006], which is applied to resolve inter-object occlusions. The last step is to update the tracks. The chapter is ended by a summary. A number of parameters are used to perform single view tracking, which are explained in the following. A complete list of the parameters is given in Appendix E.2 on page 154.

6.3 From Blobs to Detected Objects

The first action to be taken in the module is to perform connected component analysis on the binary foreground mask. The analysis is performed using OpenCV's cvFindContours method, which produces a list of blobs. The extracted blobs are the outermost blobs, meaning that if an object is segmented with holes, the holes are closed. As mentioned in Section 6.1.1, it is quite likely that an object is splitted into several blobs. To avoid this situation, blobs are merged based on two rules inspired by [McKenna et al., 2000]. Two blobs are merged if their projection of the bounding box overlaps on the x axis. Furthermore, the vertical distance between two blobs' bounding boxes must be below a threshold. This principle is illustrated in Figure 6.7. After performing merging of blobs that satisfy the two rules a list of detected objects is now available, and all further processing is based on this list. Any detected objects, which are smaller than a threshold T_{noise}, are classified as noise and removed from the list.

The disadvantage of this merging approach is that it does not take the existing tracks into consideration. A person walking above another person in the image might be wrongly merged into one detected object instead of having two detected objects. The result is that the merging procedure creates more occlusion situations. However, this type of occlusion is not difficult because there actually is no occlusion and is hence easily solved.



(a) Before merging (b) After merging

FIGURE 6.7: Merging procedure. Each blob is marked by gray and its bounding box by a dashed red rectangle. Only two blobs are merged using the two merging rules.

6.4 Object Classification

In Section 6.1.2 it is chosen to perform shape-based classification. Some examples of work using shape-based classification are reviewed in the following.

The VSAM system [Collins et al., 2000] uses three shape-based features of the moving object and a camera zoom factor to discriminate between single human, vehicle, human group and background clutter. The three shape-based features are dispersedness, area and aspect ratio. These features are given as input to a neural network. VSAM is tested to classify objects correctly at a rate of 96.9 percent. [Lipton et al., 1998] uses dispersedness and area to decide whether a moving object is human, vehicle or background clutter. To achieve more precise results, temporal consistency constraints are added. [Lipton et al., 1998] reports a classification rate of 85 percent where the main problem is occlusion of single human objects. The W^4 system [Haritaoglu et al., 2000] analyzes the vertical projection of the silhouettes to determine whether the moving object consists of multiple people. The vertical projection is also used in [Hu et al., 2006] to distinguish people from vehicles.

The features used in the reviewed work are all view variant features. Hence, a different classifier must be trained for each individual dataset and also for each view in the dataset. This is the main reason for using a method that only applies a single metric. The object classification method used in [Hu et al., 2006] only uses the spread for distinguishing people from vehicles. The spread is calculated using the vertical projection histogram on the x axis of the foreground pixels assigned to the detected object. The vertical projection histogram for detected object i is denoted by $h_i(x)$. The spread for detected object i is given by:

$$s_i = \frac{\sum_{x=1}^{width-1} |h_i(x+1) - h_i(x)|}{\sum_{x=1}^{width} h_i(x)} \tag{6.1}$$

In the equation "width" refers to the width of the detected object. The vertical projection is steeper for an isolated human than for a vehicle, making the spread value higher for an isolated human than a vehicle. Furthermore, as the vertical projection of a vehicle is smoother than that of a group of humans, the spread value of a group of humans is also higher than that of a vehicle. By applying a single threshold, T_{spread}, it is possible to distinguish human objects from vehicle objects. This threshold is view variant, but is typically within the interval 0.08 - 0.10. Examples of spread values are shown in Figure 6.8.

(a) $s_i = 0.173$ (b) $s_i = 0.149$ (c) $s_i = 0.132$ (d) $s_i = 0.127$ (e) $s_i = 0.069$ (f) $s_i = 0.064$

FIGURE 6.8: Examples of vertical projections. In (c) and (d) the two persons are detected as a single object.

All objects are thus divided into two classes; either as a human object or as a vehicle object. Typically, cyclist and skateboarders are classified as a human object. The human class covers groups of humans, isolated humans, cyclists and skateboarders. The vehicle class covers both small cars and large cars. This classification is rather coarse, but has proven sufficient for this work.

The object classification is performed for each detected object in each single frame. As an object moves through the scene, the classifier might assign the wrong label for a single frame. To avoid this situation, temporal consistency as in [Lipton et al., 1998] is applied. For each track two histograms each consisting of two bins are defined. The two histograms are:

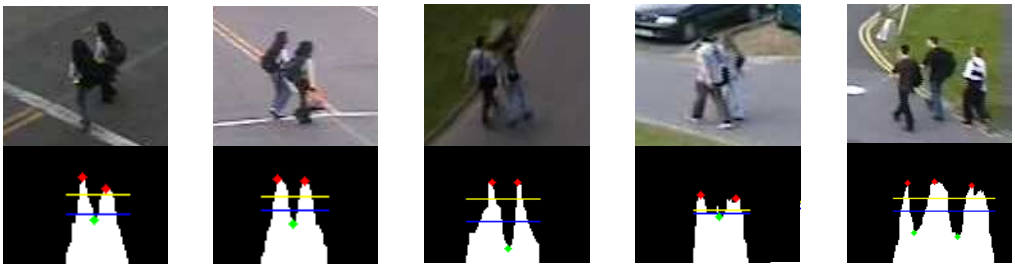
Entire history histogram The first bin is the number of times the detected object assigned to the track has been classified as a human object, which is denoted by $N_{1,human}$. The second bin is the number of times the classification was a vehicle object and is denoted by $N_{1,vehicle}$. The probability for e.g. human classification is given by $\frac{N_{1,human}}{N_{1,human}+N_{1,vehicle}}$.

Recent history histogram This histogram is basically the same as the entire history histogram, but only counts the recent N_{recent} object classifications. A N_{recent} value a little above one second of frames is found to be effective. The first bin is the number of human classifications in the latest N frames and is denoted by $N_{2,human}$. The second bin counts the vehicle case and is denoted by $N_{2,vehicle}$. The probability for e.g. human classification is given by $\frac{N_{2,human}}{N_{recent}}$.

The final probability for human classification and vehicle classification are given by the mean of the probability from each histogram:

$$P_{human} = \frac{\frac{N_{1,human}}{N_{1,human}+N_{1,vehicle}} + \frac{N_{2,human}}{N_{recent}}}{2} \qquad P_{vehicle} = \frac{\frac{N_{1,vehicle}}{N_{1,human}+N_{1,vehicle}} + \frac{N_{2,vehicle}}{N_{recent}}}{2} \quad (6.2)$$

Experiments in this work shows that only applying the entire history histogram caused needed change in the object classification to happen late. Only applying the recent history histogram would cause too many changes of the track's object classification. Combining the two has therefore proven to be effective.



(a) (b) (c) (d) (e)

FIGURE 6.9: Examples of splitting a group. Yellow line: peak limit. Blue line: valley limit. Red dot: peak point. Green dot: valley point. The detected object is splitted at each valley point.

6.4.1 Splitting a Group of People

A detected object classified as a human object is potentially a group of humans. [Haritaoglu et al., 2000] presents a method to split a group of people using the vertical projection. The approach finds the head locations in the vertical projection. [Hu et al., 2006] applies essentially the same approach, but splits the group using two thresholds. The latter approach is applied to split a group of people, because it is simpler. The idea is to find distinct peaks and valleys in the vertical projection. The two thresholds are a peak limit and a valley limit. Any peaks must be above the peak limit and valleys must be below the valley limit. This is illustrated in Figure 6.9. If a valley is located between two peaks, the detected object is splitted into two by a vertical line located at the valley point. The valley limit is given by the mean value of the entire histogram. The peak limit is given by 80 percent of the maximum value in the histogram. These values for the two thresholds have performed well for different datasets.

6.5 Data Association

After merging the blobs into detected objects and performing object classification, a list of detected objects is available. A list of existing tracks is also available, and the task is now to assign the detected objects to the tracks. This task is referred to as data association. For each track a Kalman filter estimating the bounding box location using a first-order motion model is available. The Kalman filter is used to predict the location of the bounding box and overlap with detected objects is found. The number of overlapping pixels is stored in a correspondence matrix, where the tracks are along the columns and the detected objects in the current frame are along the rows. Note that the object classification of the track or the detected object is not applied when performing data association.

The correspondence matrix is analyzed and the following situations can be identified: new object, object lost, object match, object splitting and object merging. The situations are illustrated in Figure 6.10. The new object situation results in a new track being created given that it fulfils some conditions, which is explained in Section 6.5.1. In case of a lost object, the track is kept alive. After T_{dead} frames where the track has not been assigned a detected

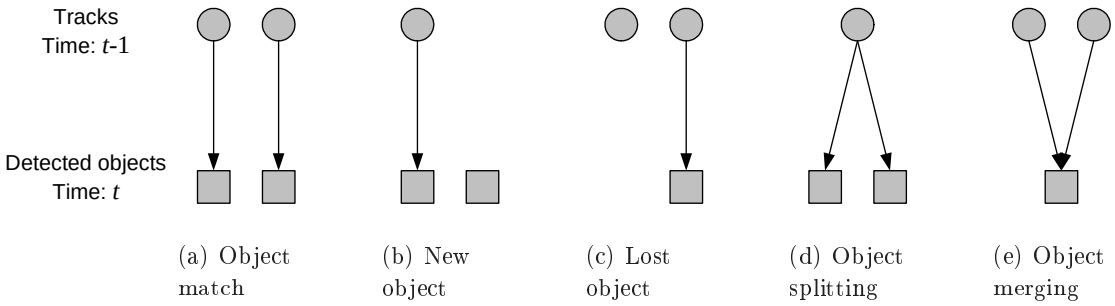


FIGURE 6.10: The five data association situations. A circle illustrates a track and a square illustrates a detected object. An arrow indicates association between a track and a detected object.

object, the track is destroyed. All tracks in the object match situation are updated using the assigned detected object. When a track has been updated for T_{stable} frames it is considered a stable track. T_{stable} is chosen to be around half a second, which is related to the use of probabilistic appearance models described in Section 6.6.

Object splitting and object merging are the most challenging situations. The two situations might happen at the same time, but object splitting situations are always resolved before object merging situations. Object splitting situations are resolved by selecting the detected object with the highest number of overlapping pixels. The associations with other detected objects are removed. This results in a track being created for the remaining unassociated detected objects. However, all detected objects that overlap with the track could be merged into one detected object, but because of the merging procedure described in Section 6.3 this is not used. Therefore, it is acceptable to only select one associated detected object during object splitting. Object merging situations are caused by two objects merging, and this is inter-object occlusion. This is resolved using probabilistic appearance models described in Section 6.6. Only stable tracks are considered when resolving occlusion situations.

As mentioned in Section 6.1.1, the data association problem is reduced with a high frame rate because of small object displacement between frames. From that point of view, it might appear unnecessary to apply a Kalman filter, but the Kalman filter is needed to apply track smoothness constraints presented in Section 6.5.2. A Kalman filter is also used to estimate the location of the centroid and the ground point. The ground point is used as the location of human objects. The estimation of the centroid is needed in order to apply the probabilistic appearance models, which is described in Section 6.6. Furthermore, the Kalman filter also performs low-pass filtering reducing the effect from noisy measurements. A description of the Kalman filter can be found in Appendix C.1 on page 141. To summarize, a Kalman filter is used to estimate the following for each track:

Bounding box The bounding box is used to perform data association by overlap with detected objects.

Centroid The centroid is used by probabilistic appearance models, which is applied to solve inter-object occlusions.

Ground point In the human case, the ground point is used as the view invariant localization

of the object. This is elaborated in Section 7.3.5 on page 90.

6.5.1 Creating New Tracks

As explained earlier, when a detected object is not associated to any existing track a new track might be created. It is desired to only create new tracks if they are fully visible. This is related to the probabilistic appearance models, where it is an advantage to initialize the model correctly (See Section 6.6 for a description of probabilistic appearance models). To address this issue, a border within the image is defined. This border is referred to as the sentry border. The bounding box of the unassociated detected object must be within the sentry border in order to create a new track. The sentry border is given by T_{sentry} and is located within 5 to 10 pixels from the image border.

The drawback of the condition above is that tracks for large objects like vehicles are created late. If a narrow view camera configuration is applied, a vehicle might always touch the image border as it drives through the scene. Hence, another condition based on the detected object's size is introduced. If the detected object is above a certain size, $T_{entrySize}$, it is created as a track even though it is not within the sentry border. The downside of this approach is that the probabilistic appearance model does not represent the total object, but only the part within the field of view. However, the alternative is that no track is created and occlusion can not be handled correctly.

6.5.2 Track Smoothness Constraints

When an object is about to exit the scene often another object enters the scene. The leaving object is tracked, but the entering object is not tracked due to the entry conditions explained above. The result is that the track might be "stolen" by the entering object. The probabilistic appearance model for the track adapts to the appearance of the entering object and is able to perform occlusion handling. However, the trajectory is wrong and the track actually contains two identities.

To avoid tracks being stolen, some constraints on the bounding box of the track are applied to ensure that the track motion is smooth. These constraints are executed after a detected object has been associated to the track. If the constraints are not met the track motion is deemed as not smooth, and the bounding box is not updated using the detected object's bounding box. Instead, the update is performed using the predicted location of the bounding box estimated by the Kalman filter. The result is that the leaving object is tracked until exit and the entering object is assigned a new track when the two objects split.

The first attempt to ensure smooth tracks was to apply constraints based on the individual motion models presented in [Veenman et al., 2001]. However, these motion models are to be used in a global data association scheme and are furthermore not developed to resolve issues related to untracked objects. Instead two other approaches are applied simultaneously:

Bounding box expansion This approach uses the bounding box and the area of the track. If the associated detected object causes e.g. a doubling of the bounding box area ($T_{bboxBox}$) and more than e.g. a 50 percent increase in the track's area ($T_{bboxArea}$), the track motion is deemed as not smooth.

Corner displacement This approach uses the corners of the track's bounding box. If the associated detected object causes a displacement of a corner more than T_{corner} pixels, the track motion is deemed as not smooth.

The corner displacement approach is only applied when the bounding box of the track is touching the image border, because the bounding box expansion approach does not perform well along the image border. The two approaches are only applied in the object match situation. They are not applied when the track is under occlusion.

6.6 Occlusion Handling using Probabilistic Appearance Models

Probabilistic appearance models are applied to resolve inter-object occlusion, which is illustrated in Figure 6.10(e). The following is based on [Senior et al., 2006], which covers the entire Section 6.6. Each track has its own probabilistic appearance model, which consists of an RGB color model with an associated probability mask. Two examples of a probabilistic appearance model are illustrated in Figure 6.11. The color model, which is denoted $M_{RGB}(\mathbf{x})$, shows the appearance of each pixel of an object. $P_c(\mathbf{x})$ denotes the probability mask and represents the probability of the object being observed at that pixel. Probabilistic appearance models might be viewed as weighted template matching, where the template is $M_{RGB}(\mathbf{x})$ and the weights are given by $P_c(\mathbf{x})$. The coordinates of $\mathbf{x}$ are expressed using the coordinate system of the model, which is normalized to the object centroid. E.g. the point $\mathbf{x} = (0, 0)$ is the object centroid in the model coordinate system. In the following, $\mathbf{x}$ expresses the model coordinate system.

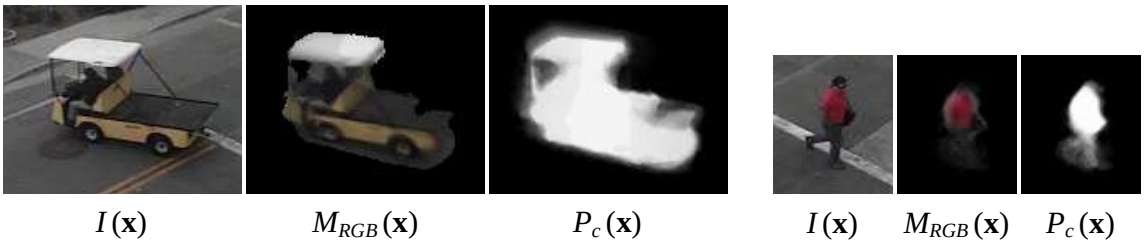


FIGURE 6.11: Two examples of probabilistic appearance models. For each example the input frame, the color model and probability mask at time t are shown.

The use of a track's probabilistic appearance model is illustrated in Figure 6.12. Depending on the given data association between the track and the detected objects, one of three possible scenarios is applied. When a new track is created, a new probabilistic appearance model is created. When in the object match situation, a track refinement step is applied before updating the model. The track refinement step refines the position of the centroid before updating. When two tracks are occluding each other only one object is detected, and this is the object merging situation (see Figure 6.10(e)). The probabilistic appearance model for the tracks are used to assign the pixels of the detected object between the two tracks using the flow in the bottom line of Figure 6.12.

In the following, the probabilistic appearance model is described starting at the foundation of the probabilistic framework. In Section 6.6.2 the track refinement principle is explained. This

is followed by a description of how the models are build, meaning the creation and updating of the models. The principle of disputed pixels and depth ordering is described in Section 6.6.4. Section 6.6.5 explains how occlusion is handled using the probabilistic appearance models, which corresponds to the bottom line of Figure 6.12.

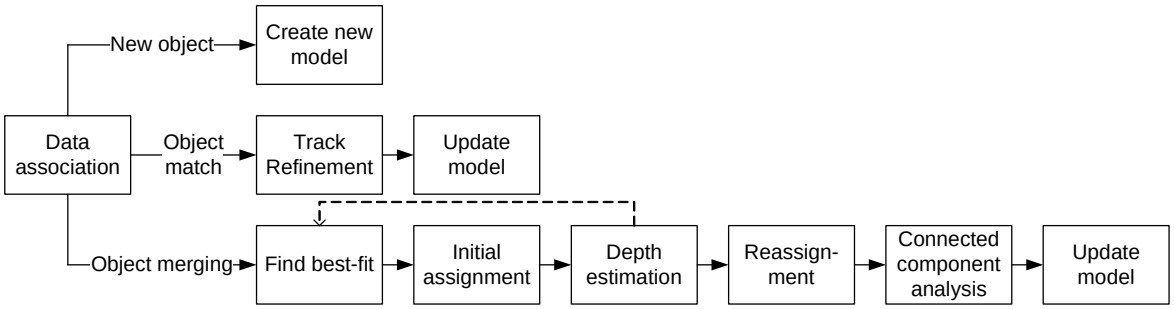


FIGURE 6.12: Flow related to the use of probabilistic appearance model.

6.6.1 The Probabilistic Foundation

The foundation of the probabilistic appearance model is the ability to estimate the probability that a given pixel $\mathbf{x}$ of a detected object belongs to the model $\mathcal{M}_j$ of track j. This is denoted by $P\left(\mathcal{M}_j|I\left(\mathbf{x}\right)\right)$. I is the color input image and is assumed to be normalized to the centroid of the detected object. The probability is calculated using Bayes' rule:

$$P\left(\mathcal{M}_j|I\left(\mathbf{x}\right)\right) \propto P_{RGB,j}\left(I\left(\mathbf{x}\right)|\mathcal{M}_j\right) \cdot P_{c,j}\left(\mathbf{x}\right) \tag{6.3}$$

The a priori probability is given by the probability mask of model $\mathcal{M}_j$, $P_{c,j}\left(\mathbf{x}\right)$. $P_{RGB,j}\left(I\left(\mathbf{x}\right)|\mathcal{M}_j\right)$ is the color appearance likelihood, and this is approximated using a Gaussian color distribution:

$$P_{RGB,j}\left(I\left(\mathbf{x}\right)|\mathcal{M}_j\right) = \frac{1}{\left(2\pi\right)^{3/2}\left|\mathbf{\Sigma}\right|^{1/2}} \cdot \exp\left(-\frac{1}{2}\left(I\left(\mathbf{x}\right) - M_{RGB,j}\left(\mathbf{x}\right)\right)^T \mathbf{\Sigma}^{-1}\left(I\left(\mathbf{x}\right) - M_{RGB,j}\left(\mathbf{x}\right)\right)\right) \tag{6.4}$$

The color model for track j, $M_{RGB,j}$, represents the mean color for each pixel. In [Senior et al., 2006] the covariance matrix $\mathbf{\Sigma}$ is assumed to be a diagonal matrix with identical variance in each color channel. Given these assumptions, Equation 6.4 reduces to:

$$P_{RGB,j}\left(I\left(\mathbf{x}\right)|\mathcal{M}_j\right) = \left(2\pi\sigma^2\right)^{-3/2} \cdot \exp\left(\frac{-\left\|I\left(\mathbf{x}\right) - M_{RGB}\left(\mathbf{x}\right)\right\|^2}{2\sigma^2}\right) \tag{6.5}$$

σ is selected empirically. Through experiments it is discovered that the assumption of the same variance in each color channel is valid. However, it is expected that the color variance is higher for a human object than for a vehicle object, which is rigid and has a larger surface with uniform color. In practice, applying different σ values for humans and vehicles did not improve the segmentation during occlusion, which is described in Section 6.6.5. Therefore, the same σ value is used for both humans and vehicles. It is further assumed that the covariance

is zero. Experiments showed that this is not actually the case, but applying this assumption causes a significant reduction in the computational demand.

6.6.2 Track Refinement

The probabilistic appearance model can be used to refine the tracking. Track refinement is equivalent to template matching and is performed before updating the track and the model in an object match situation (see Figure 6.12). The centroid is predicted using the first-order Kalman filter. Based on this predicted centroid location, the probability is calculated for each foreground pixel of the associated detected object and summed to a total probability. The probability for a single pixel belonging to the model is expressed in Equation 6.3. This is performed in an $N_{fit} \times N_{fit}$ neighborhood around the predicted centroid. The total probability for a particular object with a predicted centroid located at $(x, y) = (82, 105)$ in image coordinates using a 9×9 neighborhood is illustrated in Figure 6.13. The new centroid location for the detected object is selected at the point with the highest total probability (marked by a green dot in Figure 6.13).

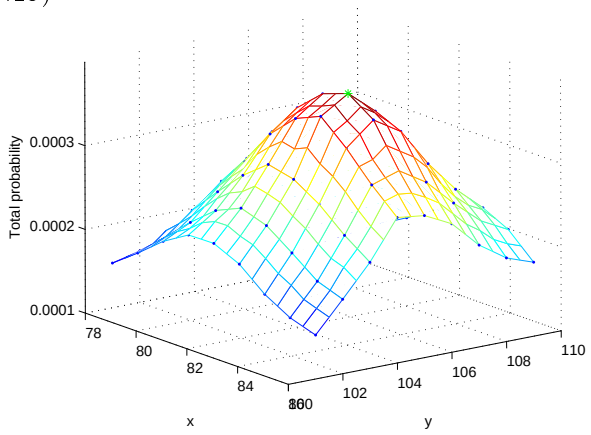


FIGURE 6.13: Total probability plotted along the z axis. The new centroid position is found at the maximum total probability marked by a green dot.

The new centroid location is used to normalize the image I before updating the model described in the following section. Without track refinement the accuracy of the model degrades; especially the color model becomes blurry. This means that using track refinement, more details are visible in the color model. Although computationally expensive, track refinement improves the model accuracy and the segmentation during occlusion. Using a neighborhood larger than 5×5 to fit the model makes processing in real-time inconceivable.

6.6.3 Building the Model

When a new track is created, a probabilistic appearance model is also created as shown in the top line of Figure 6.12. The image region containing the object is normalized to the object centroid and the object's foreground pixels are copied into the color model. The corresponding locations in the probability mask are initialized to a probability of 0.4 and the remaining pixels are set to zero probability. An initial probability value of 0.4 has proven to be acceptable; it is low enough to quickly fade away on the following update if the assigned foreground pixels

were due to noise.

When the object is isolated, meaning that the data association is the object match situation as in Figure 6.10(a), the probabilistic appearance model is updated by blending in the appearance of the associated detected object. This is the last box in the middle line of Figure 6.12. The color model is updated by blending in appearance of the current foreground pixels:

$$M_{RGB,j}(\mathbf{x},t) = \alpha \cdot M_{RGB,j}(\mathbf{x},t-1) + (1-\alpha) \cdot I(\mathbf{x}) \qquad \text{if } \mathbf{x} \in \mathcal{F} \tag{6.6}$$

$\mathcal{F}$ is the set of foreground pixels for the detected object. In the equation above it is assumed that I has been normalized to the centroid of the detected object located using track refinement as described in Section 6.6.2. The probability mask is also updated:

$$P_{c,j}(\mathbf{x},t) = \lambda \cdot P_{c,j}(\mathbf{x},t-1) \qquad \text{if } \mathbf{x} \notin \mathcal{F} \tag{6.7}$$

$$P_{c,j}(\mathbf{x},t) = \lambda \cdot P_{c,j}(\mathbf{x},t-1) + (1-\lambda) \qquad \text{if } \mathbf{x} \in \mathcal{F} \tag{6.8}$$

α and λ should be between 0.90 and 0.99. Otherwise, the model changes to fast or does not change at all. [Senior et al., 2006] suggests setting $\alpha = \lambda = 0.95$, which have proven effective.

In this way the probabilistic appearance models are continuously updated. A significant change to the object's appearance is clearly visible in the color model within half a second. This is the reason for setting the threshold for a stable track to around half a second. A significant change could be a person turning around. The probability mask has a "memory" of the object's shape. For non-rigid objects like humans the probability mask has lower probability for the limbs because these move over a large area, whereas the torso in comparison is stable. Examples of probabilistic appearance models are shown in Figure 6.11.

6.6.4 Disputed Pixels and Depth Ordering

Before describing the algorithm for segmenting occluding tracks, disputed pixels and depth ordering are explained. When two objects are merged into one detected object, the pixels of the detected object can be divided into three parts: pixels belonging to track 1, pixels belonging to track 2 and overlapping pixels. The overlapping pixels are called disputed pixels, and the principle is illustrated in Figure 6.14. A disputed pixel is identified by the fact that both tracks have a non-zero probability; or stated otherwise $P_{c,1}(\mathbf{x}) > 0 \wedge P_{c,2}(\mathbf{x}) > 0$. In the figure, red indicates non-zero probability for track 1 only, green indicates non-zero probability for track 2 only and yellow indicate non-zero probability for both track 1 and 2.

Each pixel of the detected object can be assigned to the track with the highest a posteriori probability calculated using Equation 6.3. All the red pixels are assigned to track 1 because track 2 has zero a priori probability. The green pixels are likewise assigned to track 2. However, which track is assigned the disputed pixels depends on the depth ordering of the two objects. The track that is assigned the fewest yellow pixels is given greater depth. This way, depth estimation of the objects can be estimated using the disputed pixels.

The depth estimation becomes a bit more complicated when more than two tracks are under occlusion. This is handled using a score. All tracks are compared one-by-one and it is

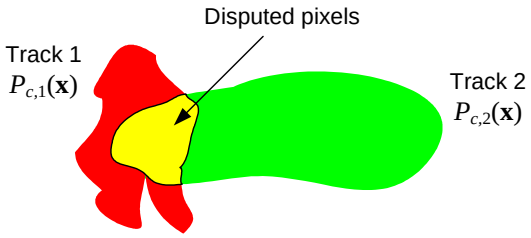


FIGURE 6.14: The principle of disputed pixels during occlusions. The disputed pixels are indicated by the yellow area, which is the overlap between track 1 and track 2.

determined which one is assigned most disputed pixels. The score is incremented for the track assigned most disputed pixels and the score is decremented for the "losing" track. This is done for all track combinations, and the track with the greatest depth has the lowest score.

6.6.5 Occlusion Handling

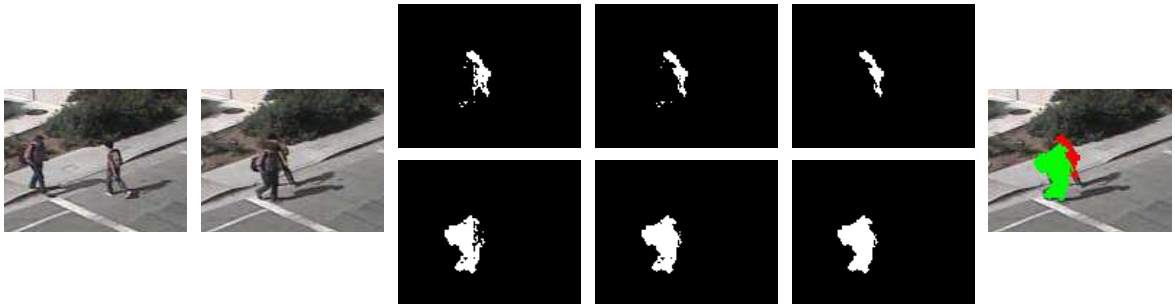
This section describes how occlusion is handled using probabilistic appearance models. The flow of the algorithm is illustrated in the bottom line of Figure 6.12. When an object occludes another object only a single object is detected. Before the merging, the objects have been updated and a probabilistic appearance model is available for each object. The objects are segmented using the procedure in Listing 6.1 and Figure 6.15 shows an example.

LISTING 6.1: Procedure for segmenting inter-object occlusions

1 The centroids are predicted for each track using a first−order Kalman filter.
2 Tracks are fitted to the foreground pixels as described in Section 6.6.2 in depth−order if this is available. The foremost track is fitted first, and the pixels where this track's probability mask has non−zero probability are not used when fitting the tracks with greater depth.
3 Given this best−fit location of the tracks, all disputed pixels are identified.
4 Each disputed pixel is assigned to the track with the highest probability calculated using Equation 6.3. This is the initial pixel assignment and is illustrated in Figure 6.15(c).
5 Tracks are ordered so tracks assigned fewer disputed pixels are given greater depth as explained in Section 6.6.4. This depth ordering is used in step 2.
6 All disputed pixels are reassigned to the foremost track which overlapped the pixels. This is the reassignment of pixels step and is illustrated in Figure 6.15(d).
7 Connected component analysis is performed to clean up the segmentation. See Figure 6.15(e).
8 All tracks with more than $T_{numPixels}$ pixels assigned are updated using the assigned pixels as explained in Section 6.6.3.

The connected component analysis, which is performed in step 7 in Listing 6.1, finds the largest blobs of the assigned pixels. Only blobs larger than e.g. 25 percent of the largest blob's area are included in the final segmentation. An example of the final segmentation result is shown in Figure 6.15(f).

In the first frame of merging of several objects, no depth estimation is available. In the fitting of the tracks, all pixels are used for all the tracks. When the amount of disputed pixels are



(a) Before occlusion (b) During occlusion (c) Initial assignment (d) Reassignment (e) Connected component analysis (f) Final result

FIGURE 6.15: Example of occlusion handling using probabilistic appearance models. (c), (d) and (e) show the intermediate steps for resolving the occlusion in (b).

large, $> T_{depth}$, the depth estimation is performed and used when fitting the tracks in step 2 in Listing 6.1. This is the reason for the dashed line connecting the "Depth estimation" block with the "Find best-fit" block in Figure 6.12. A value of $T_{depth} = 30$ has proven efficient in the available datasets.

In case of only human objects occluding each other, the depth estimation is performed differently. A high located ground point is far away from the camera, so the human object with the "highest" ground point location in the y direction is assigned the greatest depth. This assumption is only valid for human objects. Furthermore, the use of the principal axis method as described in Section 7.3 on page 85 has the benefit of making the ground point location more accurate and reliable.

6.7 Single View Tracking Test

This chapter does not contain a test of the module. The output of the module is given as input to the correspondence module. In the case of human tracks, the correspondence module improves the accuracy of the tracks. Therefore, the single view tracking module is indirectly tested in conjunction with the test of the correspondence module (Section 7.5 on page 95) and the overall system test (Chapter 8 on page 101).

6.8 Summary

The general findings in the single view tracking are:

- Blobs are merged into detected objects using a vertical and a horizontal constraint.
- Detected objects are classified using the spread. Track's object classification is made more robust using temporal consistency.
- Data association is performed using overlap between bounding boxes.
- Two smoothness constraints are applied to avoid tracks being "stolen" by entering untracked objects.
- Each track is assigned a probabilistic appearance model, which is applied to segment tracks under occlusion and refine the location of the centroid.

Chapter 7

Correspondence of Objects

The correspondence module performs object matching of the objects tracked in each view. This chapter documents the analysis and design of the correspondence module. Object matching of humans is performed using the principal axis method. Vehicles are matched using the footage region method. Having matched objects and known ground plane coordinates, a view invariant representation can be calculated. The accuracy of locating the objects in the view invariant domain is tested after describing the design.

7.1 Analysis

The correspondence module performs object matching between views for objects within the shared region. In the following, objects that match between views are also referred to as corresponding objects. Furthermore, because only the two camera setup is considered, an object in view 1 is said to be "paired" with an object in view 2.

Given that the world coordinates of the ground plane are known, the correspondence module makes a view invariant representation of the objects. As discussed in Section 2.4.5 on page 20, the view invariant representation is well suited for performing analysis of the objects' actions and interactions because the perspective effect in a view is eliminated. Furthermore, the same analysis approach can be applied to different camera setups. Hence, the output of the correspondence module is given as input to the view invariant analysis module as depicted in Figure 7.1. The figure shows the correspondence module's position in the system.

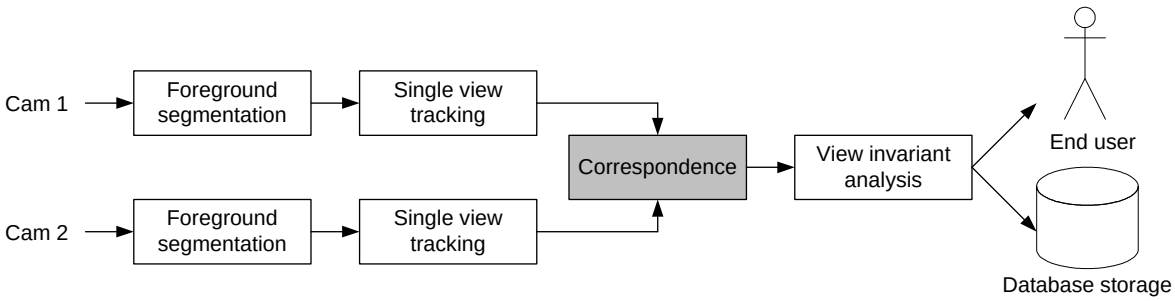


FIGURE 7.1: Overview of the system highlighting the correspondence module.

The analysis related to this module is carried out in the problem analysis and can therefore be found in Section 2.6 on page 21. The main findings in the analysis are:

- The principal axis method [Hu et al., 2006] is a robust object matching method for humans, but does not apply for vehicles.
- The footage region method [Park and Trivedi, 2006] is robust for matching vehicles, but is weaker in the human case.

Hence, the two methods are combined in order to have a more robust object matching method. From each view's tracking module a track list is available, and for each track the object classification is known, so either an object is human or vehicle. Given this object classification it is possible to find correspondence for humans between views using the principle axis method and for vehicles by the footage region method. The object classification of tracks is explained in detail in Section 6.4 on page 72. The following lists some general considerations.

7.1.1 General Considerations

In the problem analysis in Section 2.4 on page 17 a number of computer vision challenges are listed, some of which influence the correspondence of objects. The following describes some considerations related to the challenges. The considerations are incorporated into the design of the correspondence methods.

Occlusion Single view tracking might wrongly track several objects as a single object due to severe occlusion. This occurs frequently when e.g. humans move in groups. The correspondence module should take this into considerations and use information from multiple views to resolve the issue.

Inaccurate homography mapping As shown in Section 4.4 on page 33, perfect homography mapping can not be guarantied. Therefore, the methods for corresponding objects should be robust towards some inaccuracies in the homography mapping.

7.2 Conceptual Design

The conceptual design for the correspondence module is illustrated in Figure 7.2. The input is a track list from each single view tracking module. Depending on the object classification of the track, either the principal axis method or the footage region is applied. In the human case, after using the principal axis method, special attention is put on groups of humans. After this step all human objects are corresponded and their view invariant representation is calculated. In the vehicle case, the footage region is applied. Some ambiguity might occur when doing object matching using the footage region method and these are resolved next. This is followed by calculating the view invariant representation for the vehicles.

In Section 7.3 the correspondence for humans are explained. This is followed by a description of the approach for finding correspondence for vehicles. Afterwards in Section 7.5, the accuracy in locating both human and vehicle objects are tested. The chapter is ended by a summary. A number of parameters are explained in the following. A complete list of the parameters is given in Appendix E.3 on page 155.

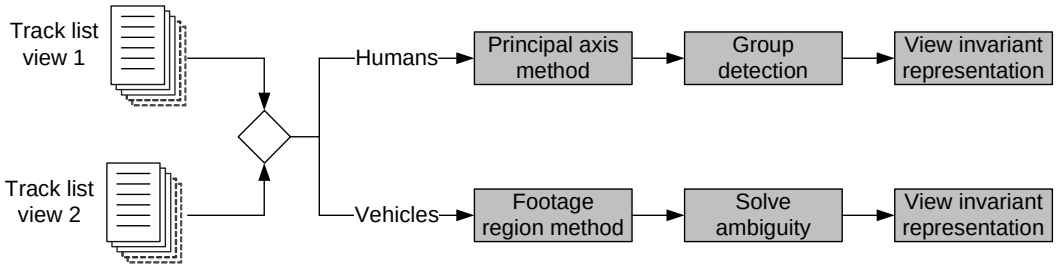
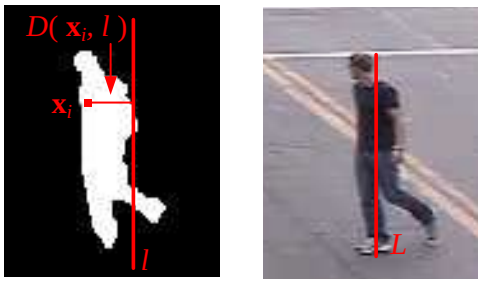


FIGURE 7.2: Conceptual design of the correspondence module.



(a) (b)

FIGURE 7.3: The principle axis is calculated by minimizing the perpendicular distance $D(\mathbf{x}_i, l)$ using least median of squares.

7.3 Correspondence of Human Objects

The correspondence of human objects between views is based on [Hu et al., 2006] and utilizes the principal axis. The principal axis is a vertical line in the image domain, which is calculated for each tracked object. The basic idea is that a human is symmetrical around the principal axis. The axis is recovered by minimizing the median of squared perpendicular distances between the foreground pixels of the object and a vertical line. The principle is shown in Figure 7.3(a), where $\mathbf{x}_i$ denotes a foreground pixel, l is a vertical line and $D(\mathbf{x}_i, l)$ is the perpendicular distance between a foreground pixel and a line. The principal axis L is determined by the least median of squares:

$$L = \arg\min_{l} median \left\{ D\left(\mathbf{x}_i, l\right)^2 \right\} \tag{7.1}$$

For a given vertical line l the squared perpendicular distance is calculated for all foreground pixels. These distances are sorted in an ordered list, and the median value of this list is found. This is done for all locations of l and the vertical line with the smallest median value in the ordered list is selected as L.

A brute force implementation of this principle is computationally expensive. Therefore, an algorithm using the vertical projection histogram is used to compute the least median of squares. The algorithm is found in [Yang and Levine, 1992], and an explanation of the algorithm is given in Appendix C.2 on page 144. An example of the principal axis L is given in Figure 7.3(b). The use of least median of squares, as opposed to e.g. least sum of squares, makes the method robust towards outliers in the foreground mask such as shadow pixels.

7.3.1 Geometrical Relationship

For each object the principal axis is extracted as explained above. The intersection of the object's bounding box and its principal axis is defined as the ground point of the object. The principle of finding correspondence is illustrated in Figure 7.4. Let $\mathbf{H}_{12}$ be the ground plane homography from view 1 to view 2 (the details of recovering this homography are described in Chapter 4 on page 29). Person S in view 1 is the projection of the 3D person S, and the figure shows the relationship with person K in view 2. L_1^S is the principal axis for person S in view 1 and $\mathbf{x}_1^S$ denotes the person's ground point. L^S is the principal axis for person S in 3D space. g_1^S is the line acquired by projecting L^S onto the ground plane in 3D space from the direction of the view of camera 1. L_1^S is the projection of g_1^S on the image plane of view 1. For person K in view 2, L_2^K and $\mathbf{x}_2^K$ denotes the person's principal axis and ground point, respectively. Let L_{12}^S be the line in view 2, which is obtained by transforming L_1^S from view 1 into view 2 using $\mathbf{H}_{12}$. L_{12}^S is also the projection of g_1^S on view 2. $\mathbf{q}_{12}^{SK}$ denotes the intersection of the warped principal axis from view 1 (L_{12}^S) and person K's principal axis (L_2^K). Given that person S in view 1 corresponds to person K in view 2, the intersection should ideally be the same as the location of the ground point. Hence, the distance between $\mathbf{q}_{12}^{SK}$ and $\mathbf{x}_2^K$ should be small and is used to evaluate the correspondence likelihood for the pair of principal axes L_1^S and L_2^K.

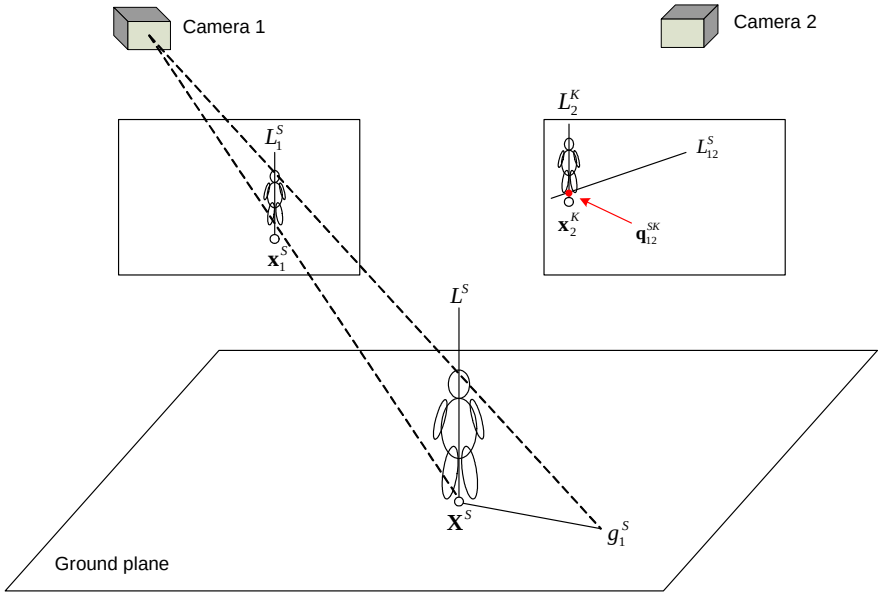


FIGURE 7.4: The geometrical relationship. Person S in camera 1 is compared with person K in camera 2.

The intersection point $\mathbf{q}_{21}^{KS}$ in view 1 can be obtained in a similar manner. The correspondence likelihood of pairing person S in view 1 with person K in view 2 is expressed by the correspondence distance [Hu et al., 2006]:

$$D_{12}^{SK} = \left(\mathbf{x}_1^S - \mathbf{q}_{21}^{KS}\right)(\mathbf{\Sigma}_1)^{-1}\left(\mathbf{x}_1^S - \mathbf{q}_{21}^{KS}\right)^T + \left(\mathbf{x}_2^K - \mathbf{q}_{12}^{SK}\right)(\mathbf{\Sigma}_2)^{-1}\left(\mathbf{x}_2^K - \mathbf{q}_{12}^{SK}\right)^T \quad (7.2)$$

A small D_{12}^{SK} value indicates a good match. $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ are 2×2 diagonal matrices. The two diagonal elements denotes the variance in the x and y direction for a given view. In

the general case, $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ would have non-zero covariance and be dependent on the image location (x, y). However, in [Hu et al., 2006] and this thesis it is assumed that coordinates x and y are independent and also the variance can be regarded as independent of the image coordinates, hence $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ are diagonal matrices. $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ can be determined by observing the distance between the intersection point and the corresponding ground point in a test sequence.

7.3.2 The Correspondence Algorithm

The correspondence distance is used in matching people by an algorithm described next. The algorithm of the principal axis method is presented in [Hu et al., 2006] and is in the following described for the two camera case, but the method extends to more cameras. Details on this can be found in [Hu et al., 2006].

At time t it is assumed, that M people with principal axes L_1^1, L_1^2, ..., L_1^M are observed in view 1, and N people with principal axes L_2^1, L_2^2, ..., L_2^N are observed in view 2. The correspondence algorithm is global, and it selects the pairs of axes by minimizing the total sum of the correspondence distance values of the pairs. The five steps of the algorithm are:

Step 1: The principal axes detected in the two views are combined pairwise. A list (θ) of all possible correspondence pairs of principal axes is created. Correspondence distances of these pairs are computed.

Step 2: For each pair (m, n) in the pair list θ, it is checked whether pair (m, n) satisfies the constraint $D_{12}^{mn} < D_T$, where D_T is a predefined threshold to classify true or false correspondence pairs. If not so, pair (m, n) is deleted from the pair list θ. Thus, the pair list θ only contains pairs satisfying the above constraint.

Step 3: From the pair list θ, all possible pairing modes are found. A pairing mode is a unique combination of the possible pairs. The pairing modes with maximum number (l) of pairs are selected and represented with $\mathbf{\Theta} = \{\Theta_1, \Theta_2, ..., \Theta_k, ...\}$, where $\Theta_k = \left\{\left(L_1^{k_1}, L_2^{k'_1}\right), \left(L_1^{k_2}, L_2^{k'_2}\right), ..., \left(L_1^{k_l}, L_2^{k'_l}\right)\right\}$. k is the index of a paring mode.

Step 4: The pairing mode from $\mathbf{\Theta}$ with the minimum sum of correspondence distance values is selected and its index is denoted by λ:

$$\lambda = \arg\min_k \left(\sum_{w=1}^{l} \left(D_{12}^{k_w, k'_w} \right) \right) \tag{7.3}$$

All principal axis pairs in pairing mode Θ_λ are the matched ones.

Step 5: The pairs in pairing mode Θ_λ are labelled.

After finding the correspondence of tracks, the intersection point is used as the measurement of the track's ground point. Furthermore, the bottom vertical line of the bounding box is changed to go through the intersection point.

(a) View 1 (b) View 2

FIGURE 7.5: The problem of groups being tracked as a single object in one view. The crosses in view 2 indicate intersection points obtained by warping the two principal axes in view 1.

7.3.3 Human Groups

It happens quite often that two or more human objects are tracked as one object, especially when people enter the scene as a group. Since multiple views are used it often happens that people are occluded in one view but separable in the other. Other work like [Senior et al., 2006] and [Hu et al., 2006] "wait" until the people in the group split and can use this information to detect the event, that a group entered the scene. However, based on observations, people that enter as a group are most likely to stay as a group while moving through the scene. The correspondence algorithm from [Hu et al., 2006] does not take this into considerations. So, in the situation where a group is tracked as individuals in view 1 and as a single track in view 2, there are tracks in view 1 that are not paired with any tracks in view 2. This situation is illustrated in Figure 7.5, where the persons in view 1 are splitted using the vertical projection as explained in Section 6.4.1 on page 74, while the occlusion is too severe in view 2 to split the objects using the vertical projection.

The basic idea of warping the principal axes is still valid in these situations. As can be seen in Figure 7.5(b) the intersection points (marked by colored crosses) are still located correctly. In the example, the location of the green intersection point is located highest. This is because the occluded person in view 2 is farther away from the camera. If more people are walking side by side, the intersection point would be located even higher.

To do correspondence for groups of humans a modified version of the correspondence algorithm is developed in this work, which is executed after the correspondence algorithm from [Hu et al., 2006]. In this version, the unpaired tracks in view 1 are located, and these are tested if they match an already paired track in view 2. If the matching conforms to two distance constraints the unpaired track in view 1 is paired with the paired track in view 2.

The algorithm is listed in the following, and Figure 7.6 is used as an illustration. Note that Figure 7.6 is the same situation as in Figure 7.5, but the person with yellow principal axis in view 1 is not drawn because he is the one being matched using the algorithm described in Section 7.3.2.

(a) View 1 (b) View 2

FIGURE 7.6: Resolving the problem of a group being tracked as a single object in view 2. See text for explanation.

Step 1: Find all unpaired principal axes in view 1. A list ($\theta_1^{unpaired}$) of all unpaired principal axes in view 1 is created and a list (θ_2^{paired}) of all paired principal axes in view 2 is created.

Step 2: A principal axis (L_1^m) in $\theta_1^{unpaired}$ is selected and is compared with a principal axis (L_2^n) in θ_2^{paired}. For each comparison two distances are calculated:

Distance 1: L_2^n is warped from view 2 into view 1 (L_{21}^n) and the intersection point is found ($\mathbf{q}_{21}^{mn}$). The distance from the intersection point to person m's ground point location is found as $D_1 = |\mathbf{q}_{21}^{mn} - \mathbf{x}_1^m|$. The distance is calculated this way for comparison with distance 2; the two distances are summed in step 3.

Distance 2: L_1^m is warped from view 1 into view 2 (L_{12}^m) and the intersection point is found ($\mathbf{q}_{12}^{nm}$). Furthermore, the ground point location, $\mathbf{x}_1^m$, is also warped from view 1 into view 2, $\mathbf{y}_2^m$. The distance from the intersection point to the warped ground point is found as $D_2 = |\mathbf{q}_{12}^{nm} - \mathbf{y}_2^m|$.

Step 3: If $D_1 < D_{T1}$ and $D_2 < D_{T2}$, then n along with the score $D_1 + D_2$ are stored in the list Λ_m. D_{T1} and D_{T2} are selected empirically.

Step 4: Select person n with the smallest score in Λ_m and label the pair (m, n).

Step 5: Remove L_1^m from $\theta_1^{unpaired}$. Go to step 2 and repeat the procedure with a different m until $\theta_1^{unpaired}$ is empty.

The algorithm is executed a second time with the order of view 1 and view 2 switched.

The main difference between this algorithm and the correspondence algorithm described in Section 7.3.2 is that this is essentially a greedy algorithm and not a global algorithm. The distance D_2 is applied, because it is expected that the warped ground point should be close to the principal axis in view 2. Furthermore, considering that the object in view 2 could be four or five people, then the distance from the intersection point to the ground point is expected to be large. This is illustrated in Figure 7.5(b) where the person farthest away from the camera has a higher located intersection point. Hence, the distance from the intersection point and warped ground point is used, and this distance must be smaller than D_{T2}.

(a) View 1 (b) View 2

FIGURE 7.7: Tracking of occluded person. Despite being occluded by the vehicle, the person's ground point is correctly located. View 1 is zoomed in on the person. A purple vertical line is a principal axis. A black box indicates the bounding box location before performing correspondence. A colored box is the bounding box location after correspondence and Kalman filtering.

Using the modified correspondence algorithm it is possible to add knowledge about the number of people in the group, even though a group is tracked as a single object.

7.3.4 Kalman Filtering

As mentioned, the intersection point found using the two algorithms described above is used as the new measurement of the ground point, and the bottom line of the bounding box is changed to go through the intersection point. The Kalman filter for the ground point and the bounding box is used to smooth these measurements. This improves the accuracy of the tracking when people are within the shared region of the two views. Even during misdetection of the lower body and occlusions, the ground point found using the intersection point from the principal axis method is not effected. An example of occlusion is shown in Figure 7.7, where the ground point and bounding box are correctly located in view 1 where the bottom portion is occluded by the vehicle. The improved tracking accuracy also improves the accuracy of the view invariant representations of humans, which is described next. Hence, view invariant analysis becomes more reliable.

7.3.5 Human View Invariant Representation

Human objects are paired using the algorithms described in Section 7.3.2 and 7.3.3. Given that the world ground coordinates are known, the view invariant representation of a human object is calculated as a single point in the virtual view. After using the Kalman filter the estimated ground point location is warped from each view into the virtual view. The final point location in the virtual view is the mean of the two warped ground points. The accuracy of this view invariant location is tested in Section 7.5. The view invariant representation for a group is shown in Figure 7.8. The displacement vectors between previous measured points in the virtual view are used for estimating the velocity vector of a human object.

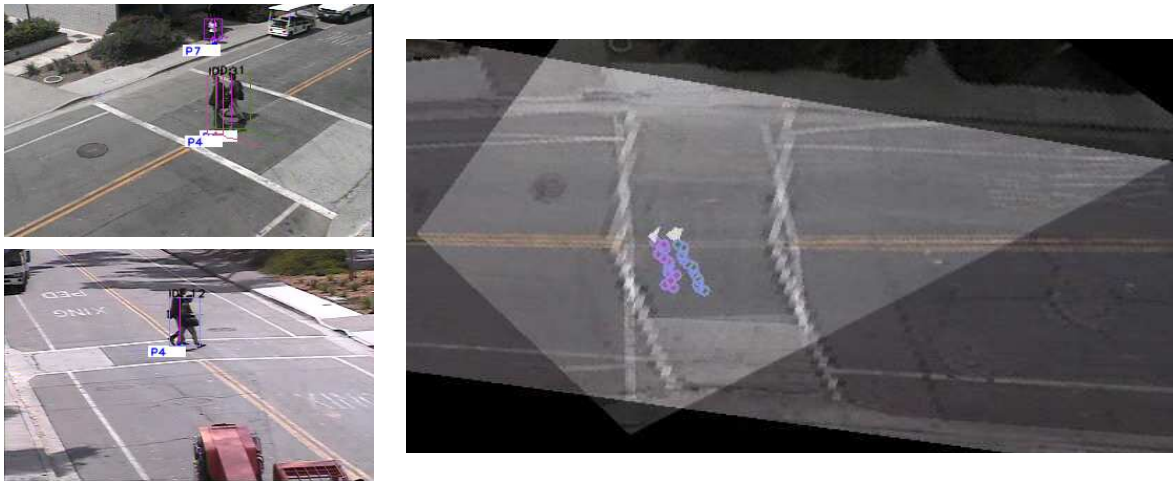


FIGURE 7.8: A group of two people are tracked and paired. To the right the view invariant representation is shown only for objects in the shared region. Even though the group is tracked as a single object in one view both persons have a view invariant representation. Previous points are shown for both humans.

7.4 Correspondence of Vehicle Objects

The correspondence of vehicles utilizes the footage region, which is also applied in the work by [Park and Trivedi, 2006] and [Khan and Shah, 2006]. However, [Park and Trivedi, 2006] and [Khan and Shah, 2006] does not apply single view tracking; both methods map all foreground pixels into a common domain and perform tracking in this domain. Hence, the applied vehicle correspondence method in this work differs on this point compared to previous work.

The approach works in two main steps. The first is to create an overlap matrix and then solve any ambiguity expressed in the overlap matrix. The overlap matrix is created by warping all foreground pixels belonging to a vehicle track from view 1 into view 2 and find overlap with any foreground pixels for a vehicle track in view 2. Obviously, the warping direction can be reversed and the result would be the same. The principle is illustrated in Figure 7.9, where the same vehicle is visible in the two views. Figure 7.9(b) shows the foreground pixels assigned to the vehicle track in each view. These pixels are warped into the other view using the planar homography and overlapping pixels are found. The overlapping pixels are the common footage region and are marked with yellow in Figure 7.9(c). In the figure, there is overlapping pixels and the vehicle tracks are a possible pair. This is marked in the overlap matrix, where the vehicle tracks in view 1 are listed along the rows and vehicle tracks in view 2 along the columns. However, overlapping pixels is only an indication of correspondence. When two vehicles drive by each other, they will at some point occlude each other. In this case, a vehicle in view 1 has overlapping pixels with multiple vehicles, and a vehicle in view 2 also has overlapping pixels with multiple vehicles. This is called a many-to-many overlap. The possible scenarios are: one-to-one overlap, many-to-many overlap and many-to-one (or one-to-many) overlap. The overlap matrix for these situations are exemplified in Figure 7.10.

(a) Input frame (b) Foreground pixels (c) Warped pixels (green or red) and overlapping pixels (yellow)

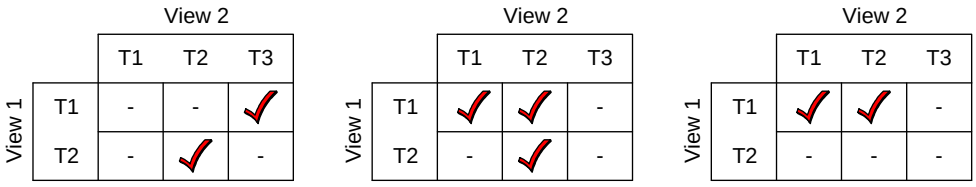
FIGURE 7.9: Finding overlap between vehicle tracks in view 1 and view 2. If there is overlapping pixels (marked by yellow), the tracks are a possible pair.

7.4.1 Solving Ambiguity

Using the overlap matrix the possible scenarios are identified. The many-to-one and many-to-many overlaps are considered as ambiguity in the overlap matrix. The algorithm developed in this work for solving the different scenarios is graphically depicted in Figure 7.11 and described in the following. When solving the ambiguity, the vehicle track with the most possible overlaps is solved first. In the end, only one-to-one overlaps are left.

One-to-one overlap: This is a straight forward situation because there is no ambiguity and is illustrated in Figure 7.10(a). Figure 7.9 is an example of an one-to-one overlap. The vehicle pairs are labelled, and the view invariant representation is calculated as explained in Section 7.4.3.

Many-to-many overlap: When two vehicles drive by each other they are at some point likely to occlude each other in one view. Because of the occlusion in one view the warped foreground mask of one the vehicles will wrongly overlap with both vehicles in



(a) One-to-one overlap (b) Many-to-many overlap (c) Many-to-one overlap

FIGURE 7.10: Possible scenarios illustrated using the overlap matrix. A check mark in the overlap matrix indicates that the warped foreground masks of the vehicles are overlapping.

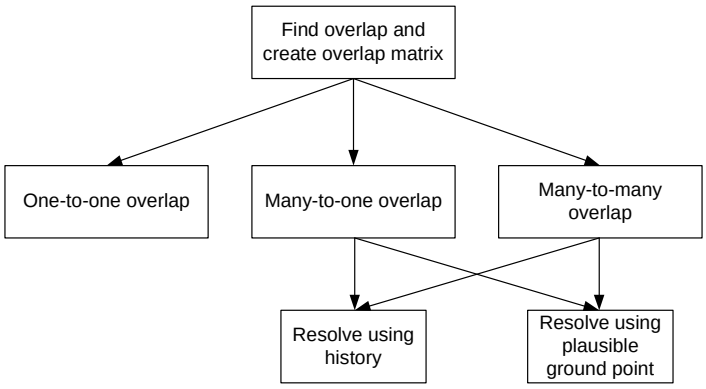


FIGURE 7.11: Algorithm applied for finding vehicle correspondence and solving ambiguity in the overlap matrix.

the other view. This yields an overlap matrix as illustrated Figure 7.10(b).

One of two methods is applied to solve this ambiguity. The first method is used to ensure that a historic "relationship" is preserved between vehicle tracks. Before the occlusion, the overlap matrix does not express any ambiguity and the vehicles are paired correctly. Using the historic correspondence the ambiguity is solved by maintaining the same correspondence. The historic relationship is only considered if the last $T_{history}$ frames have had the same pairing of vehicle tracks.

It is not guaranteed that the historic relationship is available to resolve the ambiguity, e.g. at track initialization. In these situations a different approach based on a "plausible ground point" is applied. Taking the vehicle's centroid and a vertical line through the centroid, the plausible ground point is located at the lowest foreground pixels on this vertical line. The plausible ground is illustrated in Figure 7.12(a) for two vehicles, where there is occlusion in view 1. This plausible ground point is located on the ground, but also beneath the vehicle itself. When warping this plausible ground point into the other view using the planar homography, the warped point should therefore also be located beneath the car in this view. The warped point is used to solve the ambiguity and is illustrated in Figure 7.12(b). The plausible ground point is only applied if the history is not reliable.

Many-to-one overlap: The many-to-one overlap situation is exemplified in Figure 7.10(c) and could be caused by two vehicles being tracked as one in a view. However, the view with many vehicle tracks could wrongly track noise classified as a vehicle. Furthermore, a cyclist can be misclassified as a vehicle due to the object classification metric used. In experiments it is observed, that even though vehicles are merged into one track they often split a little while later. Of course, this depends on the specific scene. Based on this experience, it is chosen to resolve this ambiguity as in the many-to-many overlap; first trying resolving using the history and, if not available, apply the plausible ground point.

(a) View 1 (b) View 2 (c) Legend

FIGURE 7.12: The plausible ground point can be used to solve ambiguities in the overlap matrix. The many-to-many overlap occurs in this situation because the vehicle with red centroid in view 1 occludes the lower part of the vehicle with the green centroid.

7.4.2 Handling Vehicle Occlusions

After resolving the ambiguities and pairing the vehicle tracks, it might happen that a pair of vehicle tracks does not overlap even though it is historically expected. The missing pair of vehicle tracks could be caused by the entire bottom portion of the vehicle being occluded in one view. An example is given in Figure 7.13(b), where the white van is occluding the bottom portion of the turquoise car in view 1. Before occlusion, the turquoise car has been paired correctly between views and a pairing is therefore expected. The foreground pixels assigned to the turquoise car in view 1 are shown in Figure 7.14(a) during occlusion. Since no foreground pixels are available on the bottom portion of the vehicle there is no overlap with the turquoise car in view 2.

The issue is resolved by utilized the probability mask from the probabilistic appearance model for the turquoise car. Probabilistic appearance models are explained in Section 6.6 on page 77. The probability mask represents the probability that a pixel belongs to the object. Stated otherwise, it contains a "memory" of the shape of the vehicle. This memory is utilized in reconstructing the shape of the vehicle. The probability mask is thresholded and projected into the image as shown in Figure 7.14(b). With the shape reconstructed it is possible to once again find overlap with the turquoise car in view 2. Given that there is overlap, the vehicle tracks are paired. Thus, vehicle tracks are correctly corresponded even during severe occlusion. Reconstruction of the vehicle shape is only applied, when there is a historically

(a) Before occlusion (b) Severe occlusion of turquoise car

FIGURE 7.13: Example of severe occlusion. The occlusion causes a missing historically expected pairing of the turquoise car in view 1 and view 2.

(a) Foreground pixels (b) Reconstructed shape

FIGURE 7.14: Solving severe occlusion by reconstructing the shape of the occluded vehicle.

missing pair and one of the vehicle tracks of this pair is under occlusion.

7.4.3 Vehicle View Invariant Representation

When ground coordinates are known, the view invariant representation for humans are found as a point in the virtual view. It is also possible to represent a vehicle as a single point in the virtual view, but since vehicles differ significantly in size this is not advantageous. Size in this context refers to the area on the ground below the vehicle. Because of this, it is chosen to represent the vehicle by its footage region in the virtual view. A large vehicle like a truck then has a larger area in the virtual view than a small vehicle like an electric cart. Because the foreground detection of the vehicle might be missing a few pixels, "holes" in the footage region of the vehicle might occur. This can be solved by fitting an ellipse to the points that make up the footage region in the virtual view. Another possibility is to find the convex hull for the points and fill in all the points within the convex hull. The latter approach is chosen since it has proven to be less sensitive towards outliers compared to fitting an ellipse.

Having the footage region in the virtual view (enclosed by convex hull), the centroid is located. This centroid simplifies calculation of the vehicle's velocity vector. The most recent displacement vectors are applied to calculate the velocity vector. In conclusion, the view invariant representation consists of the footage region in the virtual view and this region's centroid. The view invariant representation for a truck is illustrated in Figure 7.15.

The size of the footage region is sensitive to misdetection of the vehicle in one view. Especially if the pixels at the bottom of the vehicle are misdetected, the footage region is reduced in size. The footage region increases in size if the vehicle's cast shadow is detected in both views. Hence, the location of the centroid might not be perfectly accurate, but experiments showed that the centroid location is reliable for calculating the vehicle's velocity.

7.5 Correspondence Accuracy Test

The accuracy of locating humans and vehicles is documented first in the following. Afterwards, the effect of doing correspondence of objects is investigated. This is followed by a test of the

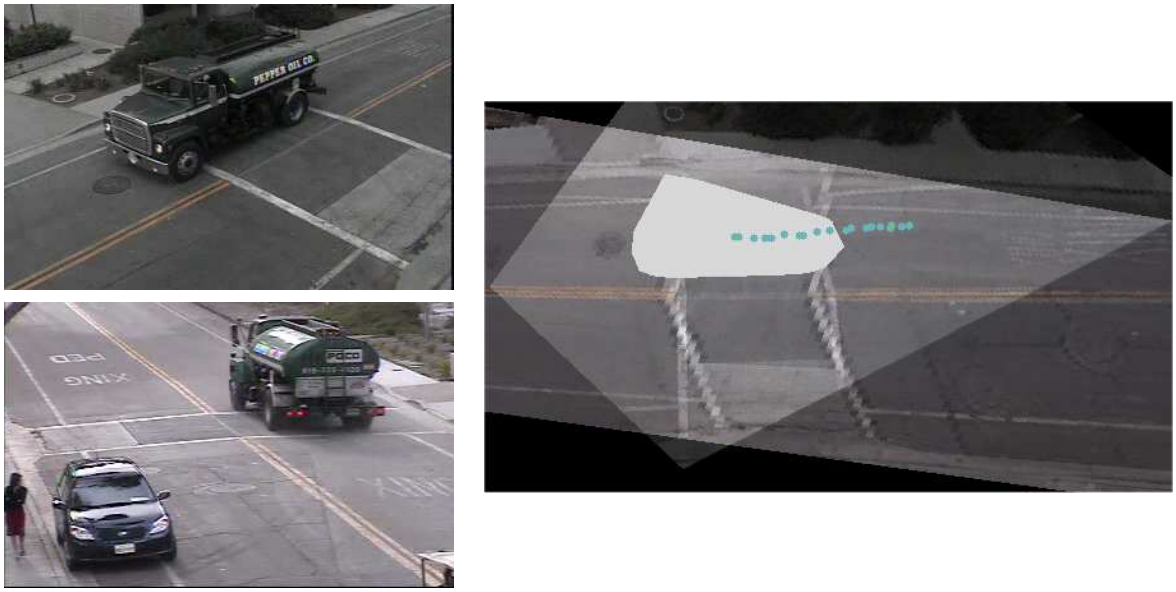


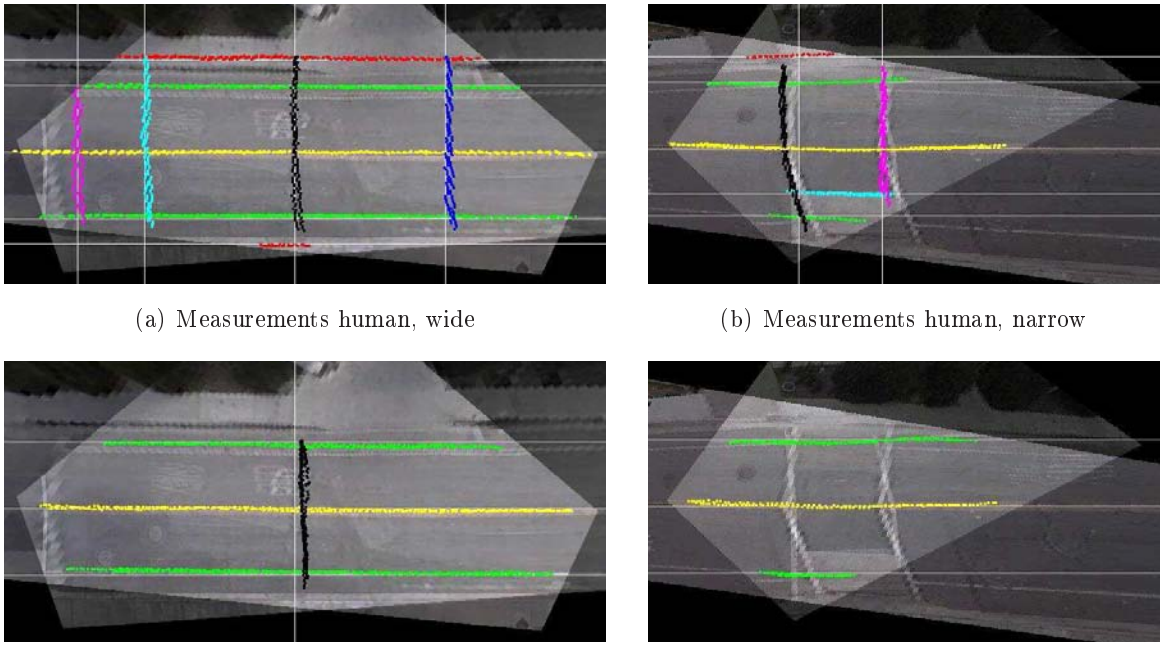
FIGURE 7.15: The greenish truck is tracked in both views and paired. To the right the view invariant representation is shown only for objects in the shared region. The big blob is the footage region enclosed by convex hull, and the blue points are the current and previous centroid locations of the footage region.

estimated size of a vehicle and a test of the estimated speed.

In order to measure the accuracy of locating humans and vehicles, ground truth is needed. For this test, choreographed sequences from the Matthews Lane dataset is used. The dataset is described in detail in Appendix D.1 on page 147. Matthews Lane has several line markings, which is used as ground truth lines. There is a narrow view and a wide view camera configuration in the Matthews Lane dataset. The narrow view covers a ground plane area of 10×20 meters, and the wide view covers 13×32 meters. As mentioned in Section 4.4 on page 33, the ground plane coordinates are found by having a person pacing out the scene. Because of this, the ground truth and all measurements are not 100 percent accurate. However, the result can be used to give an indication of the accuracy level.

The accuracy test for humans is conducted by having a person walking along the line markings. This is conducted in both the narrow view and wide view camera configuration. The same is conducted for a vehicle. The vehicle drives along the lane markings, where possible. The lane marking should be located directly under the middle of the vehicle.

The system performs foreground segmentation, single view tracking and correspondence of the objects on the four sequences (two for the human case and two for the vehicle case). The results can be found on the DVD (◎ */tests/correspondence_accuracy_test/*). The view invariant representation for a human is a single point, and this is used as measurement for comparison with the ground truth. Vehicles are represented with its footage region and this region's centroid. The centroid is used as measurement to be compared with the ground truth. The measurements are illustrated in Figure 7.16 overlaid on the virtual view of the scene. Color coding is applied to measurements belonging to the specific lane marking. Lane markings make up the ground truth and a thin white line is drawn on top of the lane markings.



(a) Measurements human, wide

(b) Measurements human, narrow

(c) Measurements vehicle, wide

(d) Measurements vehicle, narrow

FIGURE 7.16: Measurements overlaid on the road. The measurements are color coded to make it easier to distinguish measurements when these overlap. The ground truth is illustrated as a thin white line.

Each measurement is compared with the ground truth. The ground truth is either a vertical line or horizontal line in the virtual view (See Figure 7.16). The perpendicular distance from the measurement to the ground truth line is used as the estimation error. The mean absolute estimation error for all measurements is listed in Table 7.1. To perform view invariant analysis reliably, the measurements must be close to the ground truth, but they must also be stable. The stability of the measurements can be estimated by the standard deviation of the measurements, which is also listed in Table 7.1. If the standard deviation is high, estimation of the velocity might not be reliable.

Some of the inaccuracy of the result in Table 7.1 could be contributed to the inaccuracy of the ground truth and ground plane coordinates as already mentioned. Furthermore, driving in the vehicle and having the lane marking directly below the middle of the vehicle is not a simple task. This is the most probable reason for the high mean absolute estimation error for vehicles in the wide view camera configuration.

When looking at the black measurements in Figure 7.16(b) it is evident that the assumption of a planar ground plane is not valid for the given scene. This contributes to the inaccuracy of the result. However, considering the cameras are elevated to 6.5 meters looking down on the objects, the accuracy result is considered satisfactory for the later view invariant analysis.

7.5.1 Disabling Correspondence

The accuracy test can be used to illustrate the advantage of using multiple views compared to a single view. For vehicles it is not possible to calculate the view invariant footage region

	Camera conf.	Mean abs. estimation error	Measurement std. dev.
Human	Narrow	15.25 cm	13.47 cm
	Wide	9.13 cm	9.51 cm
Vehicle	Narrow	8.80 cm	9.09 cm
	Wide	21.20 cm	10.60 cm

TABLE 7.1: Accuracy results for humans and vehicles.

	Correspondence	Mean abs. estimation error	Measurement std. dev.
Narrow	Enabled	15.25 cm	13.47 cm
	Disabled	19.72 cm	21.97 cm
Wide	Enabled	9.13 cm	9.51 cm
	Disabled	38.33 cm	30.75 cm

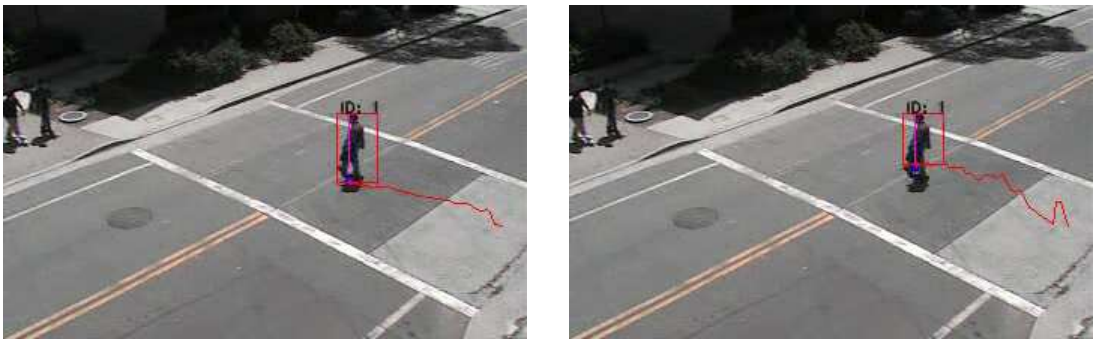
TABLE 7.2: Accuracy results when correspondence is enabled and disabled for humans.

when only single view information is available. However, the view invariant representation of humans can be obtained from a single view by warping of the ground point from the image domain into the virtual view. Therefore, the test is conducted once more using the two sequences for humans, but this time with the correspondence module disabled. The point in the virtual view is calculated by warping the ground point from each image domain into the virtual view. This results in twice as many measurements as before. The result with correspondence module enabled and disabled is listed in Table 7.2.

The mean absolute estimation error increases by disabling the correspondence module. Furthermore, the standard deviation of the measurements increases causing problems for the reliability of the following view invariant analysis. A slight improvement of the result with correspondence enabled is due to the fact that the virtual point is calculated as a mean of two points. However, the most significant reason for increased performance when doing correspondence is the improvement of the ground point location for tracks. This is exemplified in Figure 7.17, where the rather poor foreground segmentation causes the noisy location of the ground point shown in Figure 7.17(b). It should be noted that the foreground segmentation in the test sequences are good and no occlusions occur. If this is not the case, the estimation error is expected to increase further when disabling correspondence. Using the principal axis method the localization of humans is robust towards misdetection of the lower body (exemplified in Figure 7.17(a)) and occlusions.

7.5.2 Size Test

The vehicle used in the test is a small electric cart and measures 1.11×2.82 meters. The ground plane size of the vehicle is 3.13 m^2. The footage region should therefore approximately cover the same area. In the sequence with the narrow view camera configuration 409 measurements are available for calculating the vehicle's size. Based on the measurements the size is estimated to 4.06 m^2. 1060 measurements are available for the wide view camera configuration, and the size is estimated to 4.02 m^2. In both cases the standard deviation is around 0.35 m^2.



(a) Correspondence enabled (b) Correspondence disabled

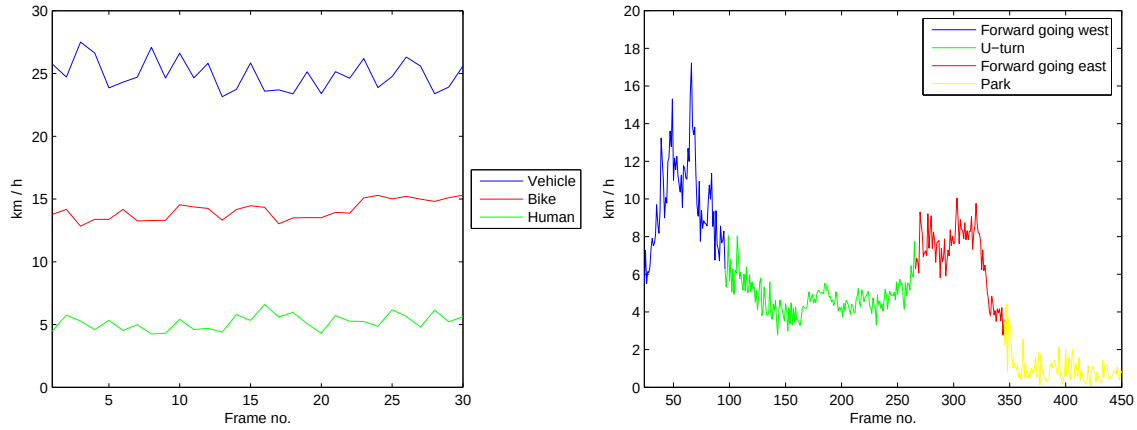
FIGURE 7.17: The effect of doing correspondence for humans. This is also one of the positive effects of using multiple views instead of a single view.

The size estimate is almost the same for the two views, but larger than the true ground plane size of the vehicle. This is probably due to two factors. The first is the problem of inaccuracies in the ground plane coordinates, which has been discussed before. Whether this increases or decreases the size of the vehicle depends on the type of inaccuracy. The second factor is that off-plane points might be mapped to the ground plane and become part of the footage region. This is very likely given the camera setup used since all sides of the vehicle are not visible. Adding a third camera would reduce the influence of off-plane points being part of the footage region.

7.5.3 Speed Test

Given that the location of humans and vehicles can be found as a point in the virtual view, it is possible to estimate a ground plane velocity vector. The velocity vector is calculated using the displacement vectors. The magnitude of this velocity vector is the object's speed. No ground truth data is available for objects' speed. However, to illustrate that it is possible to calculate the velocity vector (and thus the speed) a vehicle, a cyclist and a walking human are selected from the Matthews Lane dataset. Their velocity vectors are automatically estimated by the system and the speeds are calculated. A two second period (30 frames) of the estimated speed is illustrated in Figure 7.18(a). The standard deviation for this period is around 1.0 km/h for each object. As expected the vehicle moves at a higher speed than the cyclist, which is faster than the walking human.

The examples in Figure 7.18(a) show objects moving at a constant speed. It happens often that an object accelerates or decelerates. An example is shown in Figure 7.18(b), where a vehicle is driving forward, then performs a U-turn, drives forward in the opposite direction and finally parks. The graph shows how the speed of the vehicle changes when going through the four phases. The color coding is applied to illustrate the different phases of the vehicle's movement. As expected, the speed changes throughout the U-turn manoeuvre.



(a) Speed for different object

(b) Vehicle making a U-turn and parking

FIGURE 7.18: Examples of the estimated speed for moving objects.

7.6 Summary

The general findings in the correspondence module are:

- Humans are paired using the principal axis and correspondence algorithm presented in [Hu et al., 2006].
- When a group of people is tracked as a single object in one view and as individuals in another view, a new modified version of the correspondence algorithm is applied.
- Vehicles are corresponded by their footage region.
- An overlap matrix is introduced to identify ambiguities when doing vehicle correspondence. Historic constraints and plausible ground points are applied as methods to resolve ambiguities.
- Severe occlusions of vehicles are resolved by reconstructing the vehicles shape using the probabilistic appearance model.
- Humans are represented as a point in the view invariant virtual view. Vehicles are represented by their footage region and the centroid of this region.
- The accuracy of locating human and vehicle objects are tested using a narrow and wide view camera configuration. The mean absolute estimation error is 13.60 cm. An average standard deviation of 10.67 cm on the view invariant localization of the objects is reported.

Chapter 8

Results and Discussion

All the modules except the view invariant analysis module are tested as a complete system. The test is documented in this chapter. The test method is described first and the test sequences used in the test are introduced. This is followed by the execution and results of the test. Afterwards, the results are discussed keeping the issues related to humans and vehicles separate.

8.1 Overall System Test

An overall system test is performed and documented in this chapter. However, the view invariant analysis module is not part of the overall system test, because this module is application dependent. The view invariant analysis module is introduced later in Chapter 9 on page 117, where it is used to demonstrate the applicability of the system and evaluate the system. Figure 8.1 shows the overview of the system, where the modules tested in the overall system test are highlighted.

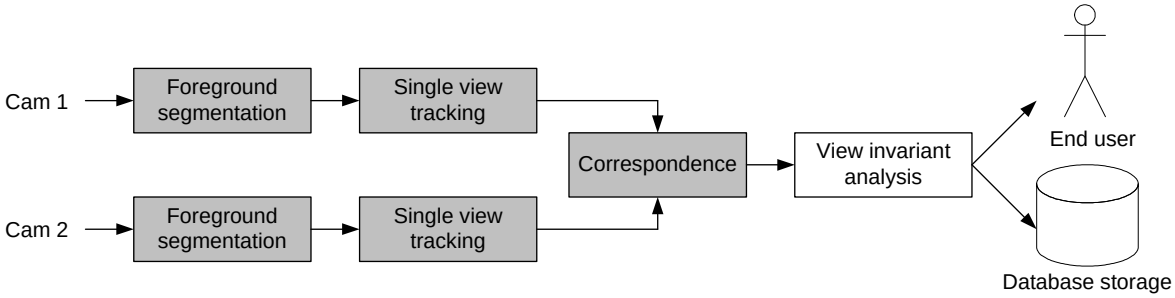
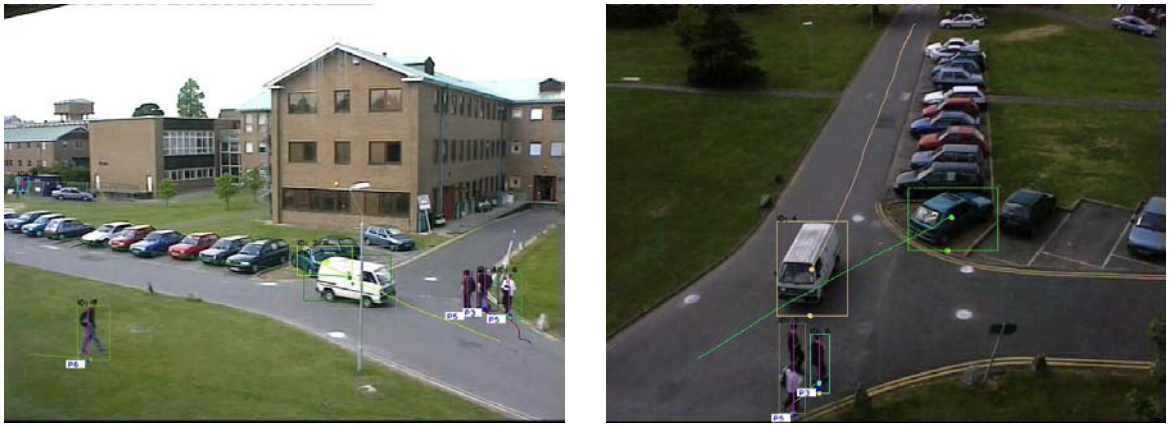


FIGURE 8.1: Overview of the system highlighting the modules tested in the overall system test.

The output of the correspondence module is the view invariant representation of the humans and vehicles. This output is used to evaluate the performance of the system. For instance, if a human is not tracked in a view, the human object is not detected in the view invariant virtual view. The point is that any error observed in the virtual view is due to an error of the underlying modules.

The test is described first. This is followed by the execution and the results of the test. Finally, the results are discussed.



(a) View 1 (b) View 2

FIGURE 8.2: Tracking result for the PETS 2001 dataset. Vehicles and humans are correctly classified and tracked.

8.2 Test Description

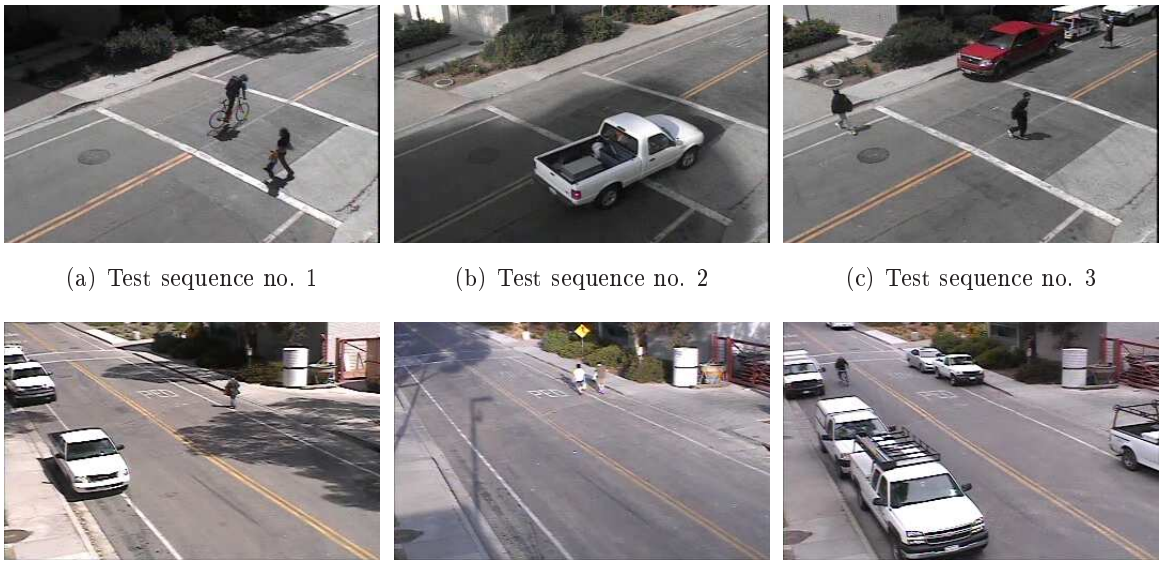
Three datasets are available for testing the system and they are described in Appendix D on page 147. The PETS 2001 and HERMES datasets are short in comparison with the Matthews Lane dataset, and this is the main reason for using the Matthews Lane dataset for the overall system test. However, the system is also tested on the PETS 2001 and HERMES datasets, and qualitative results can be seen on the DVD (◎ */tests/pets/* , */tests/hermes/*). An example of the tracking result for the PETS 2001 dataset is shown in Figure 8.2. Images of the results in the HERMES dataset are available in Section 9.2 on page 118, since the dataset is used to demonstrate the system's potential for detecting a near collision between a group of humans and a vehicle.

For the overall system test the Matthews Lane dataset is applied, which is described in detail in Appendix D.1 on page 147. This dataset uses two camera configurations and contains recordings lasting several days. Using this dataset it is possible to test the system in a dynamic traffic environment at different times of the day, under different illuminations and with different camera configurations. The system is tested using six test sequences, and an overview of the test sequences are given in Table 8.1. The table shows, that three of the six test sequences are recorded with the narrow view camera configuration, and the other three are recorded with the wide view camera configuration. The date and time period for the test sequence are listed next. The columns "# humans" and "# vehicles" refer to the actual number of humans and vehicles within the shared region of the two cameras. People on skateboards and on bikes are considered humans, whereas motorcyclists are considered vehicles. Only objects within the shared region can be matched and should be represented in the output of the system. Therefore, only objects within the shared region are considered.

The column "BG hours" in Table 8.1 indicates the time period the foreground segmentation module has been active before the single view tracking modules and correspondence module are activated to start the test. For the first test sequence in the table, the foreground has been segmented continuously for six hours when the test sequence starts. The point of testing

No.	Camera	Date	Time of day	# humans	# vehicles	BG hours
1	Narrow	03-05-2007	14:00-15:00	236	51	6
2	Narrow	04-05-2007	07:00-08:15	184	30	1
3	Narrow	04-05-2007	11:00-12:00	364	57	5
4	Wide	16-04-2007	14:30-15:00	92	29	1
5	Wide	14-05-2007	18:00-19:40	177	53	51
6	Wide	15-05-2007	10:00-11:00	298	47	67

TABLE 8.1: The six test sequences applied in the overall system test. See text for explanation.



(a) Test sequence no. 1 (b) Test sequence no. 2 (c) Test sequence no. 3

(d) Test sequence no. 4 (e) Test sequence no. 5 (f) Test sequence no. 6

FIGURE 8.3: Snapshots from the six test sequences. Only a single view is shown for each test sequence.

the system in this manner is that the foreground mask might not be the most optimal quality, but the quality is realistic if the system is to be deployed and perform 24 hours a day, seven days a week. Furthermore, note that the applied test sequences last from 30 minutes to 100 minutes because of varying traffic throughout the day.

In total, tracking output from 385 minutes of test data is used. This covers tracking of 1351 humans and 267 vehicles. Snapshots from each test sequence are shown in Figure 8.3.

To evaluate the system performance a definition of when an object is correctly tracked is needed. The definition of a correct tracked object is: From the time the object enters the shared region and leaves the shared region, the track representing the object in each view must remain the same and the tracks must be paired correctly between views. However, for instance in only a single frame it can happen that the tracks are not paired correctly or the object is not detected in one view. As long as these errors are below 0.5 second of continuous duration, the tracking of the object is considered as correct. The point is that an error occurring for a

single frame can easily be corrected a little while later using the history and is therefore not a problem.

When the tracking of an object is wrong for more than 0.5 second, the system is said to make an error. Five types of overall system errors are identified and explained in the following section. Because overall system errors are caused by errors made in the underlying modules, the types of module errors are listed afterwards. This is followed by an overview of the errors given in Figure 8.4.

8.2.1 Types of Overall System Errors

The five overall system errors are:

No object in virtual view (totally) An object is within the shared view and visible in both views, but is not detected in the virtual view at any time.

No object in virtual view (partially) An object is within the shared view and visible in both views, but only a part of the object's trajectory is detected in the virtual view. The missing parts of the object's trajectory must last longer than 0.5 second.

Wrong correspondence in virtual view Objects are wrongly paired between views for more than 0.5 second. E.g. a white van and blue car is within the shared region; if the white van in view 1 is paired with the blue car in view 2 there is a wrong correspondence in the virtual view.

Wrong object classification in virtual view An object is detected in the virtual view, but with the wrong object classification for more than 0.5 second. E.g. a vehicle detected as human in the virtual view.

Non-existing object tracked in virtual view An object is detected in the virtual view, but there is no object. This could be caused by noise being wrongly detected as humans in both views. The non-existing object must be detected for more than 0.5 second.

These errors are observed in the virtual view, but are caused by errors occurring at module level. These errors are addressed in the following section, and an overview of these errors and overall system errors is given in Figure 8.4.

8.2.2 Types of Module Errors

The three modules foreground segmentation, single view tracking and correspondence are responsible for the overall system errors. The foreground segmentation module is described in Chapter 5, and the errors related to this module are:

Noise detected Noise is detected as a human or vehicle object.

Misdetection in one view An object within the shared region is detected in one view, but not in the other.

Misdetection in two views An object within the shared region is not detected in both views.

The single view tracking module is described in Chapter 6, and the errors related to this module are:

Wrong object classification The object is classified wrongly, which causes it not to be paired by the correspondence module.

Track drifts The track smoothness constraints might cause a track to drift away from the actual object.

Track switch A permanent track switch happens. This is identity switch between two tracked objects.

Track lost Track is lost because the underlying motion model is not valid. The motion model is that objects move with a constant velocity.

Track lost during occlusion The track is lost during inter-object occlusion, which should be resolved by the use of probabilistic appearance models.

Track stolen by untracked object The track is "stolen" by an untracked object, which typically happens near the image border. For instance, if an untracked vehicle steals a track for a human, the object classification of the track is wrong.

Bad track initialization The track is not initialized properly. This can be due to redundant representation or late track initialization. Redundant representation is when several tracks represent the object. This could be caused by poor detection of the object so it is divided into several disjoint blobs, which the single view tracking module is not able to merge. Furthermore, this error also covers the case where no track is created. An object must be isolated before a track is created, which in some cases cause a late or no track initialization.

The correspondence module is described in Chapter 7, and a single error is identified for this module:

Bad homography The correspondence of objects is wrong. This could happen if the planar homography is not accurate enough, especially for visually very small objects.

8.2.3 Typical Error Relationship

Figure 8.4 presents an overview of the overall system errors and the module errors. Furthermore, the figure shows the typical relationship between the module errors and overall system errors. For example, a wrong object classification can cause that no object is detected in the virtual view (if the object classification is only wrong in a single view), and it can cause a wrong object classification in the virtual view (if the object classification is wrong in both views).

8.3 Execution

The system is executed with the test sequences listed in Table 8.1. The foreground segmentation module uses the same parameters for all the sequences. However, the parameters are not

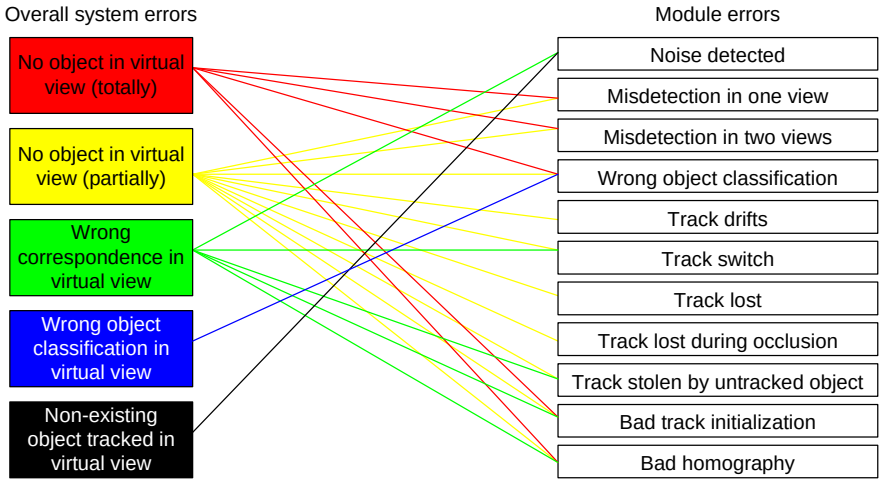


FIGURE 8.4: The typical relationship between overall system errors and module errors. The overall system errors are color coded for easier overview.

adjusted to give the optimal result, but instead adjusted so it is able to perform over several hours. The parameters for the single view tracking and correspondence module are the same for the three test sequences for the narrow view. The parameters are the same for the three wide view test sequences, but different from the narrow view parameters.

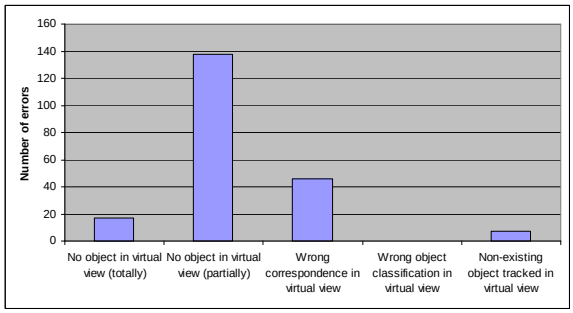
For each test sequence, the system produces an output video file containing the tracking results. Furthermore, a text file for each tracked object is generated listing the tracking results for the given track. The output video file and text files for each track are stored on the DVD (◎ */tests/overall_ system_ test/*). A file containing the parameters used for the given test sequence is stored in the same library.

The output video file is inspected and all overall system errors are noted. A single error corresponds to a single object. In cases tracking of an object causes several errors, the first occurring error is noted if this error is responsible for the following errors. For each overall system error, the module error that caused the overall system error is also noted. In this way, it is possible to identify the most occurring types of errors. Errors for vehicles and humans are kept separate so it is possible to analyze the performance related to the two object types.

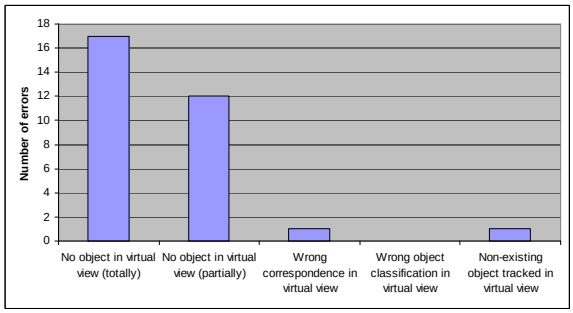
8.4 Results

The results for the test sequences are shown as graphs in Figure 8.5. The graphs show the overall system errors for each camera configuration and object type. It is clear from the graphs that the partial detection of the object is the most occurring error. As mentioned, the module error that caused the overall system error is also noted during the test. Figure 8.6 shows all the module errors related to human objects and vehicle objects. The figure indicates that the most occurring types of module errors are different for the two object types.

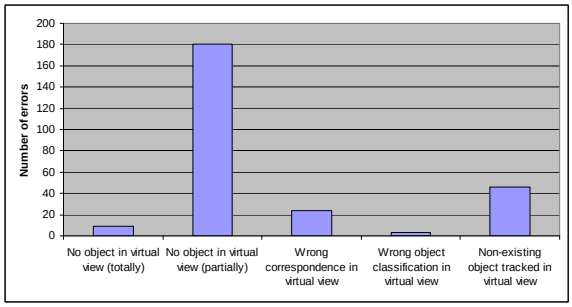
As shown in Figure 8.4 there is a certain relation between the overall system errors and the module errors, which is not expressed by the graphs in Figure 8.5 and 8.6. Therefore, the most and second most occurring module error for each overall system error are listed in Table 8.2. Only the overall system errors where several errors occur are listed in the table. It is



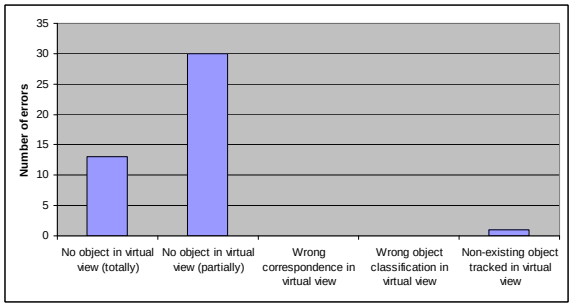
(a) Narrow view, human (Total: 784)



(b) Narrow view, vehicle (Total: 138)



(c) Wide view, human (Total: 567)



(d) Wide view, vehicle (Total: 129)

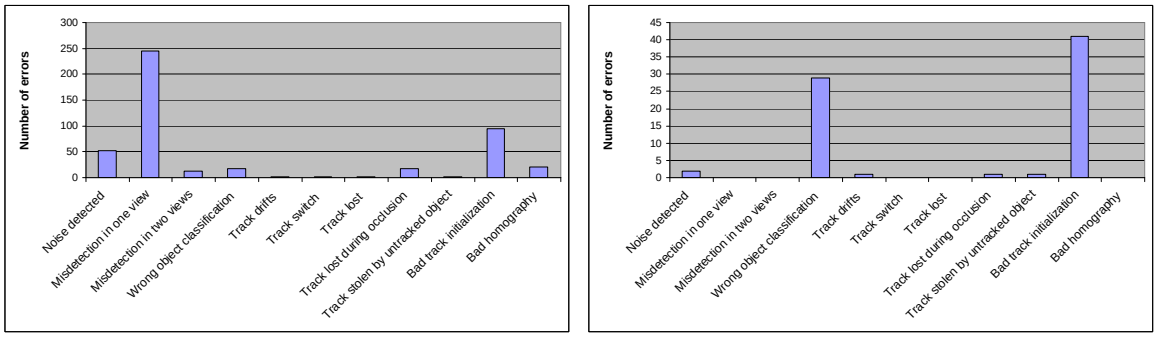
FIGURE 8.5: Overall system errors. The five error types are listed along the x axis and the number of observed errors along the y axis. "Total" refers to the total number of the given object type in the given camera configuration.

seen from the table that the two most occurring module errors explains almost all overall system errors. Table 8.2(a) shows the relation for tracking of humans. For humans, the most occurring module errors are misdetection in one view and bad track initialization. Special attention is put on these two errors during the discussion of the results related to humans in Section 8.5.2. Table 8.2(a) shows the relation for tracking of vehicles, and the most occurring module errors are bad track initialization and wrong object classification. Special attention is put on these two errors during the discussion of the results related to vehicles in Section 8.5.3.

8.4.1 Correctly Tracked Objects

Figure 8.5 and 8.6 only shows the errors, but does not illustrate how many objects are correctly tracked. The percentage of correctly tracked objects is shown in Table 8.3(a). The average in the table is 67.9 percent. However, from the table it is clear that the system performs better for the narrow view camera configuration. The main reason for this is that the system error of partial detection of objects happens more frequently in the wide view. In the wide view camera configuration, when an object is large in one view it is small in the other view. Small objects are more often misdetected than larger objects, and this causes partial detection of objects. In the narrow view camera configuration the objects are large in both views.

As mentioned, the partial detection of the object in the virtual view is the most occurring system error. Figure 8.7 shows a typical example of this error type, where the human object



(a) Human (b) Vehicle

FIGURE 8.6: Module errors. The module error types are listed along the x axis and the number of observed errors along the y axis.

Overall system error	No. 1 module error	No. 2 module error
No object (totally)	Misdetection in one view (46%)	Bad track initialization (38%)
No object (partially)	Misdetection in one view (72%)	Bad track initialization (14%)
Wrong correspondence	Bad track initialization (67%)	Bad homography (14%)
Non-existing object	Noise detected (96%)	Wrong object classification (4%)

(a) Human: Relation between overall system errors and module errors

Overall system error	No. 1 module error	No. 2 module error
No object (totally)	Bad track initialization (83%)	Wrong object classification (17%)
No object (partially)	Wrong object classification (55%)	Bad track initialization (36%)

(b) Vehicle: Relation between overall system errors and module errors

TABLE 8.2: The relation between overall system errors and module errors. Only the relevant overall system errors are shown along with the most and second most occurring module error. How often a module error occurs for an overall system error is shown as a percentage.

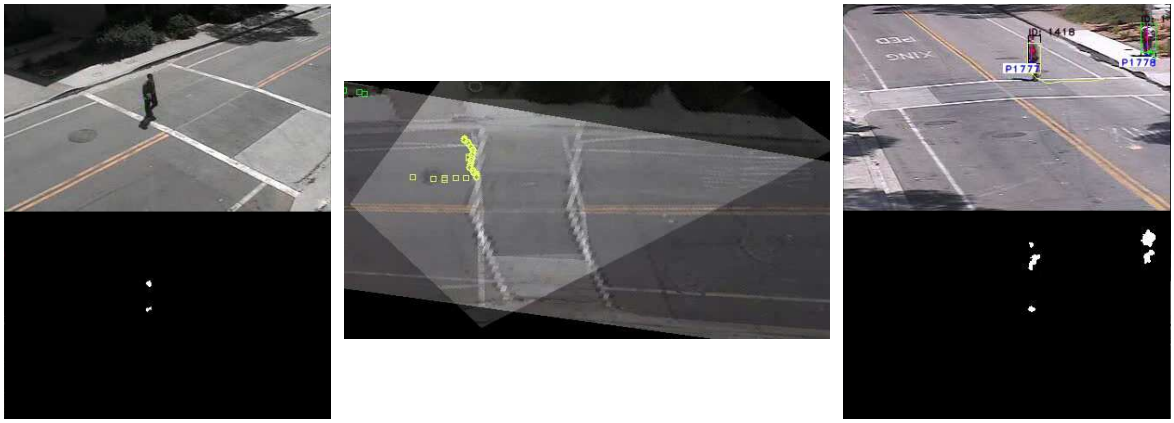
	Human	Vehicle
Narrow view	74.4%	77.5%
Wide view	53.8%	65.9%

(a) All system errors. Average: 67.9 %

	Human	Vehicle
Narrow view	92.0%	86.2%
Wide view	85.5%	89.1%

(b) Ignoring partial detections. Average: 88.2%

TABLE 8.3: Percentage of correctly tracked objects in the virtual view.



(a) View 1 (b) Virtual view (c) View 2

FIGURE 8.7: Typical example of partial detection of an object in the virtual view.

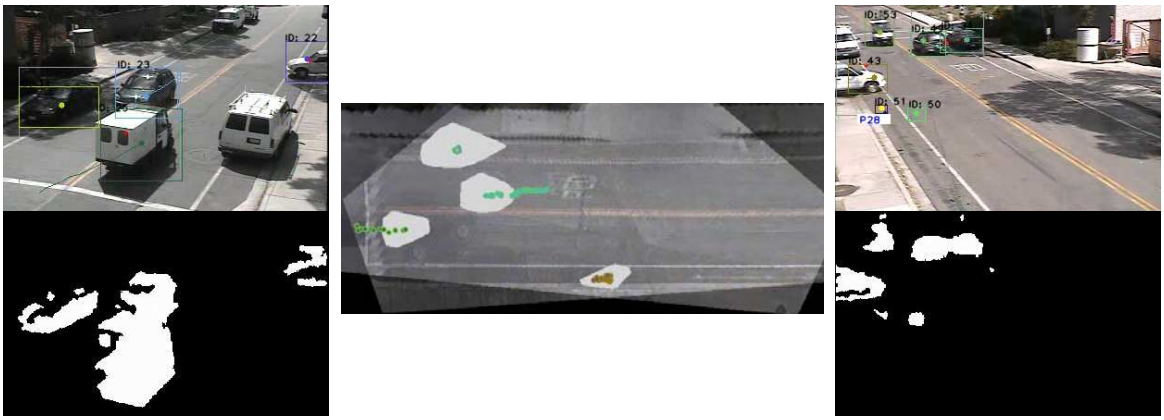
is misdetected in view 1. This causes the trajectory of the person's track to break in the virtual view. However, a few frames later the person is detected once more in view 1 and he is once more detected in the virtual view. Because the person is still tracked in view 2 this type of error is not as critical when compared with not detecting the object in the virtual view at all. Given this argument, it can be argued that it is justifiable to ignore the error type of partial detection of an object in the virtual view. Ignoring this error type results in an increase of the percentage of correctly tracked objects. This is listed in Table 8.3(b), and the average percentage of correctly tracked objects is increased to 88.2 percent. Furthermore, when comparing the system's performance in the narrow view and wide view using Table 8.3(b) it is almost the same.

8.5 Discussion

The following discusses the results and the errors presented in the previous section. The discussion is divided into four parts. The first part focuses on the system's overall performance and performance over time. Afterwards, results for tracking of human objects are discussed. This is followed by discussing the vehicle tracking results. The two objects are kept separate because different issues influence the tracking results. Finally, how the system manages detected noise is discussed.

8.5.1 Overall Performance

It is observed when inspecting the test data, given that the foreground segmentation is good and the tracks are created correctly, the tracking performs well. The correspondence of both vehicles and humans are rarely incorrect despite of the inaccuracies related to the homography mappings of the Matthews Lane dataset. This is also the case during occlusion of foreground objects. A tracking example is shown in Figure 8.8, where the four moving vehicles are correctly detected in the virtual view despite of occlusion. Note that noise is present in view 2, but this is not detected in the virtual view, which is an advantage. In general for the



(a) View 1 (b) Virtual view (c) View 2

FIGURE 8.8: Correct tracking of four moving vehicles during tracking. The noise present in view 2 are not detected in the virtual view.

given scene most occlusion is rather short term, because the objects are in transit. Long term occlusion occurs e.g. when people move in groups, but given that it is possible to split the group, the tracking still performs well. However, the few times a track is lost during occlusion, the occlusion has been long term.

Despite partial detection in the virtual view occurs often, a significant part of the objects' trajectories are detected. Due to the accuracy of the view invariant representation (see the test in Section 7.5 on page 95) the system provides a solid foundation for doing view invariant analysis and detect potentially dangerous situations. An evaluation of the overall system including view invariant analysis is carried out in Chapter 9 on page 117.

Compared to related work on tracking, 67.2 percent average correctly tracked objects is rather low. However, tracking results in related work are typically based solely on test sequences lasting a few minutes and not hour long sequences. Because the system is tested with long sequences, the quality of the foreground segmentation is not optimal, which do cause a significant drop in performance. However, testing with long sequences is needed to measure the system's performance during the day. Furthermore, unlike related work this system natively holds a view variant and view invariant representation of the objects. The representations can be used for behavior analysis and object identification.

Performance over Time

The system's performance over time can also be characterized using the overall system test. If ignoring the error of partial detections of objects in the virtual view, the performance for tracking of an object type is the same for the given camera configuration at different times of the day. However, when partial detections of objects are considered, the system's performance varies throughout the day. Dark and large cast shadows from background objects caused by bright sunny weather as shown in Figure 8.9 causes many partial tracks because of either poor detection or misdetection within the shadow region.

In test sequence no. 3, which is recorded from 11:00 to 12:00, the sun is very bright mak-

ing objects more similar to the background. The foreground segmentation is especially poor during the first 15 minutes of this test sequence and many overall system errors occur. Afterwards, the sun is not nearly as bright, and the segmentation quality increases significantly and likewise for the overall system performance. These issues caused by a very bright sun is not part of the foreground segmentation test in Section 5.10 on page 56, where the weather is changing between sunny and cloudy throughout the day. A second explanation of this issue is the use of layers in the foreground segmentation; a general observation from the foreground segmentation test is that layers are created for areas in the scene where many people move frequently. Due to the high traffic of objects they are absorbed into the background model and not segmented correctly until the layers are deleted. In test sequence no. 5, which is recorded from 18:00 to 19:40, at the end of the sequence it is twilight, and the objects start to appear more darker and similar to the background. This makes the objects harder to detect and causes a drop in performance.

8.5.2 Human Objects

As seen from Table 8.2(a), the main problem with tracking of human objects are misdetection and bad track initialization, which are discussed first in the following. This is followed by considerations regarding cast shadows and human groups, which typically are problems for a tracking system.

Misdetection

When the misdection of an object is similar to that illustrated in view 1 in Figure 8.7, the single view tracking module is not able to detect the human object. However, it is often the case that the human object is tracked in the other view, which is seen by the rather low errors for misdetection in two views in Figure 8.6(a). As mentioned, misdetection also occurs when the shadow from background objects is dark and large; it is hard to detect small human objects moving in the shadow even by manually inspecting the video. An example is given in Figure 8.9. Better performance of the foreground segmentation module is needed in order to resolve this issue. A possible solution could be to make a feedback loop from the tracking of objects to the foreground segmentation; e.g. codewords are made smaller in the region where the tracking predicts an object to be located.

Bad Track Initialization

Bad track initialization in the human case happens when e.g. a vehicle has just parked and a human object enters in a way so that it is merged with (or partly occluded by) the parked vehicle. The human object is considered as part of the vehicle object by the tracking module and is not created as a track before the human object separates from the vehicle. This causes a late track creation and hence yields a partial detection in the virtual view. If the human object never separates no track is created, and this would cause no detection in the virtual view.

FIGURE 8.9: Example of strong background shadow making detection of human objects hard. Two views are shown with four people. Arrows indicate how the people correspond between the two views. When the objects are visually large in a view they are easier to detect.

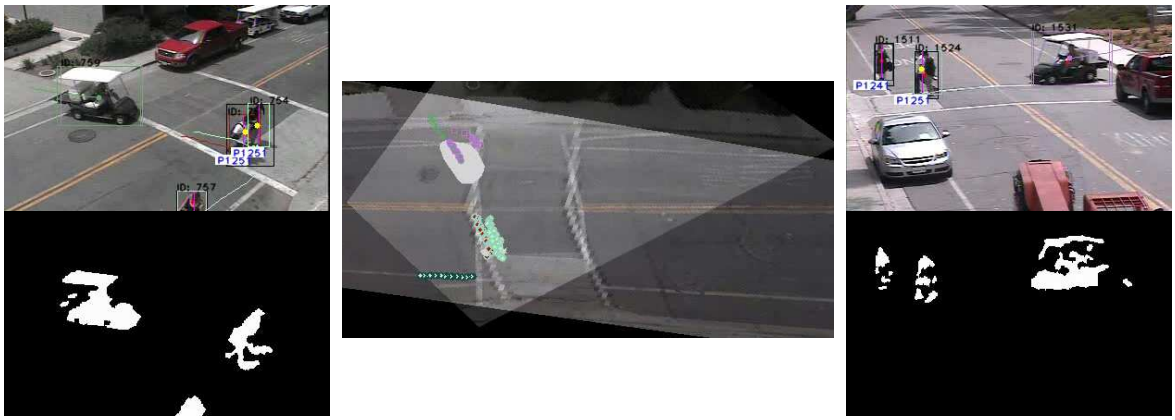
Moving Object Cast Shadows

The cast shadows from human objects are not always suppressed by the Codebook method. The shadow do not cause a problem if it is tracked along with the human object, which is typically the case as shown in the foreground mask in Figure 8.10(a). However, in some cases the cast shadow is detected as a disjoint blob. When the shadow is disjoint and of sufficient size, a track is created for the shadow. This is redundant representation of the human object. This is not an issue for vehicles due to their size and shape. The shadow issue for humans occurs most often during evening, where the cast shadows are long. When the shadow is tracked separately, it might be matched with the human object in the other view using the group correspondence algorithm (See Section 7.3.3 on page 88). Hence, two human objects instead of a single human object are detected in the virtual view, but the cast shadow does not result in a lost track for the true human object. If occlusion between moving objects (inter-object occlusion) does happen, the track for the shadow is the only track that might be lost. Different approaches for solving the shadow issue by shadow suppression are tested, but as described in Section 5.8 on page 50 none perform well over longer periods.

Human Groups

The correspondence of humans utilizes a modified version of the correspondence algorithm presented in [Hu et al., 2006]. The modified version makes it possible to detect groups and an example is shown in Figure 8.10. Despite two persons being tracked as one in view 2, two trajectories are available in the virtual view. The segmented cast shadows in view 1 and the holes in the segmentation in view 2 does not cause a problem. Furthermore, the person leaving view 1 and the small vehicle is correctly tracked. This makes it possible to perform view invariant analysis and e.g. detect if the vehicle is about to collide with the human group.

When many people enter the scene at the same time, the modified version of the correspondence algorithm is not able to correctly detect the individuals in the group. In the test data, this occurs for the largest groups, which varies from six to nine people. However, the location of the group is detected. A more feasible solution could be to use the footage region of the large group and then mark this region as a crowd instead of finding the individual ground points. The footage region should be reliable in this situation because the group occupies a



(a) View 1 (b) Virtual view (c) View 2

FIGURE 8.10: Correct tracking of a group of people. Furthermore, an isolated person and a vehicle is correctly tracked.

large area on the ground plane.

8.5.3 Vehicle Objects

As seen from Table 8.2(b), the main problem with tracking of vehicle objects are wrong object classification and bad track initialization, which are discussed first in the following. This is followed by considerations regarding long term occlusion of vehicles which are growing in size.

Wrong Object Classification

Unlike for human objects, vehicles are not completely misdetected by the foreground segmentation. This is probably due to the size of the vehicle. However, the foreground detection of the vehicle might be rather poor. In these cases, the vertical projection also becomes steeper and thus more similar to that of a human. The metric used for object classification is sensitive to this, and it occurs that vehicles are wrongly classified as a human object. Therefore, the vehicle is not detected in the virtual view.

Especially when the vehicle is visually small and poorly detected by the foreground segmentation, it is misclassified as a human as explained above. However, the use of the temporal constraints on the object classification means that the vehicle is often correctly classified as a vehicle when it becomes visually larger. Therefore, wrong object classification most likely only causes partial detection of the vehicle in the virtual view. However, in the narrow view camera configuration the time period the vehicle is within the shared region is rather short, and therefore any wrong object classification is rarely corrected. This is one of the reasons for having more total misdetections than partial in the narrow view (See Figure 8.5(b)).

Bad Track Initialization

The wrong object classification could also result in the vehicle track being splitted into several human tracks using the vertical projection method (See Section 6.4.1 on page 74). If this

(a) Correct tracking of vehicle (b) 2.5 seconds later: Redundant representation

FIGURE 8.11: Redundant representation due to poor detection of the vehicle.

happens for a longer time period, the created human tracks become stable and there is a wrong object classification and redundant representation of the vehicle. This is illustrated in Figure 8.11, where the vehicle is tracked as one and then splits into several human tracks. The vehicle might not be paired or the footage region for the vehicle in the virtual view is not estimated correctly. This is an example of bad track initialization, which is the second of the two main module errors for vehicles. As mentioned, the bad track initialization is partly caused by the wrong object classification, which is sensitive to poor foreground segmentation.

Bad track initialization often occurs when the vehicle enters the scene in the narrow view camera configuration. If it is poorly detected it might be splitted into several human tracks as explained above before the vehicle is completely within the scene. When the bad track initialization occurs in these situations, the vehicle is not detected at all in the virtual view. This further explains why the total misdetection of vehicles is larger than the partial misdetections in the virtual view for the narrow view camera configuration (See Figure 8.5(b)). In general for both camera configurations, if the vehicle is occluded by several human objects when it enters the scene, it might "steal" the tracks and is therefore represented by several human tracks and is not detected in the virtual view.

It is observed that when bad track initialization occurs, it is harder to recover from this situation compared to wrong object classification. Whereas wrong object classification is most likely to result in a partial detection in the virtual view, bad track initialization results in total misdetection in the virtual view. This point is confirmed by Table 8.2(b), where the order of the two module errors are switched for total and partial misdetection in the virtual view.

Long Term Occlusion Related Issue

Long term occlusion is in some cases an issue when tracking vehicles. The problem is that a vehicle grows quickly in visual size during occlusion compared to humans. The vehicle grows faster because it is moving at a higher speed and its shape can change significantly e.g. when making a turn. The "new" pixels that appear when the vehicle grows are not assigned to the vehicle track by the probabilistic appearance models. The result is many unassigned pixels. If a track overlaps with these pixels it simply takes the unassigned pixels and the vehicle is represented by several tracks. This issue is aggravated if the vehicle is not completely within the scene and is under occlusion. A possible solution is to add a step in the occlusion handling, which assigns all unassigned pixels to a track and maybe in favor of vehicle tracks.

8.5.4 Detected Noise

It occurs that noise is detected by the foreground segmentation module, which can not be removed by the noise threshold. The noise is generally classified as human instead of vehicle. This is because the vertical projection of noise typically is steep like the vertical projection of a human. Looking at the module error of noise detected in Figure 8.6(a) and 8.6(b), it is clear that noise is rarely detected as a vehicle. This is furthermore illustrated by the overall system error of non-existing objects tracked in the virtual view in Figure 8.5.

When the noise is detected in one view only, it is rarely paired with any object in the other view. Hence, no object is wrongly detected in the virtual view, which is also the case for the noise in view 2 in Figure 8.8. If noise should be detected in the virtual view, it must be located at the same area in the scene for the two views. This happens when a vehicle parks and is absorbed as a new layer into the background model as explained in Section 5.7.2 on page 48. The vehicle fades away over a period of time, and thus some noise is detected. The parked vehicle is located at the same area for the two views, and this could result in noise being wrongly detected as a human object in the virtual view. Similarly, when a parked vehicle starts moving it might leave behind a ghost, which could be detected in the virtual view. However, noise not detected with regards to a parked vehicle is unlikely to cause a detection of an object in the virtual view.

8.6 Summary

The main findings of the overall system test are:

- The system is tested using six test sequences with a total length of 385 minutes.
- Five overall system errors are identified during the test. An overall system error is caused by a module error and their relationship are noted.
- 67.9 percent is the average percentage of correctly tracked objects. Ignoring partial detections of objects, the average percentage increases to 88.2 percent.
- The main issues related to tracking of humans are misdetection by the foreground segmentation module and late track creation due to merged objects.
- The main issues related to tracking of vehicles are wrong object classification and redundant representations of the vehicle object.

Chapter 9

Evaluation by View Invariant Analysis

This chapter describes potential uses of the system for increasing traffic safety. Furthermore, a module for doing view invariant analysis is introduced to demonstrate the applicability of the system. From the view invariant representation of objects, events are defined that trigger alarms in potentially dangerous situations.

9.1 View Invariant Analysis

The purpose of the system is to provide a solid foundation for obtaining situational awareness. As mentioned in Section 2.1 on page 13, situational awareness can be described by three levels of understanding, where the highest level is the ability to project object states and events into future scenarios. The system covers the first level and some aspects of the second level by doing robust tracking of both humans and vehicles in temporal and spatial domain. Furthermore, the system provides the information needed to understand the significance of objects and events in a traffic environment. However, to reach the highest level of situational awareness a new module is needed on top of the existing system. The view invariant analysis module therefore adds a new level of interpretation to the system and provides an external interface for end users as depicted in Figure 9.1.

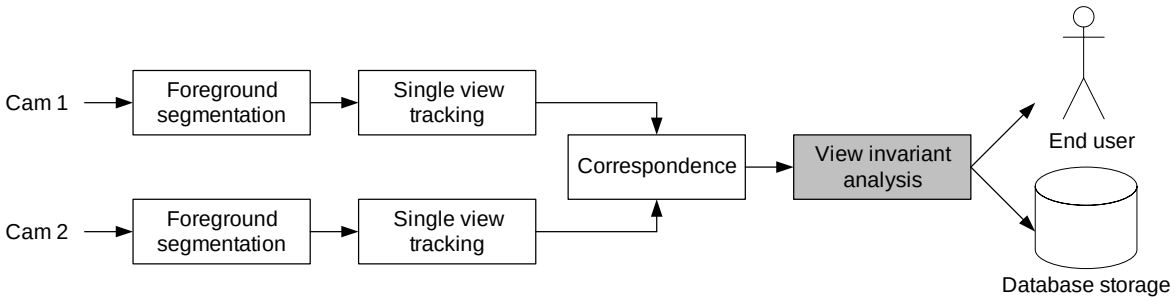


FIGURE 9.1: Overview of the system highlighting the view invariant analysis module.

A view invariant representation of objects is practical since measures such as distances and sizes of objects have meaningful relationships. Without it, it would be difficult to understand the significance of objects and events and to analyze the interaction of objects. A major advantage is also that the same analysis approach can be applied in any camera setup.

The needed complexity of the view invariant analysis depends on the application. The application defines the events that must be recognized and how accurate and detailed the recognition

should be. For instance, in some scenarios it might be sufficient to recognize coarse events to direct the attention of a human operator, while fully autonomous systems would require more detailed event recognition to allow the system to make decisions of its own. A coarse event could be a vehicle stopping on the road, while a detailed event could be prediction of collisions between objects. Several parameters can be considered in object event recognition such as position, size and velocity. These are all part of the object representations produced by the system.

The following demonstrates the applicability of the system using view invariant analysis. First, an example of collision detection is presented using the HERMES dataset, a description of the dataset can be found in Appendix D.2. Following this, a more end user oriented demonstration is presented using both recorded sequences and live camera input from Matthews Lane dataset, which is descriped in Appendix D.1.

9.2 Collision Detection

An example of prediction of a collision using view invariant analysis is shown in Figure 9.2. The example is based on choreographed data from the HERMES dataset. Two pedestrians are talking and crossing the road at the same time at a measured speed of 2.0-3.0 km/h. At the same time, a car is driving towards them with a measured speed of around 10.0 km/h. Potential collisions are detected as the intersections of the velocity vectors of the objects. Velocity vectors are depicted as red arrows, and intersections are marked by yellow crosses. By extending the velocity vectors, two intersection points are detected indicating a possible collision. Furthermore, from the view invariant analysis it can be shown that the distance between the front of the car and the nearest pedestrian is decreasing as an additional indication of a possible collision. At the given time in Figure 9.2, this distance is approximately 2.7 meters. In this sequence both pedestrians stop and take a step back as the car passes by. This example shows that it is possible to detect collisions from the object representations produced by the system, and the next step would be to prevent the accident.

9.3 Demonstration using View Invariant Analysis

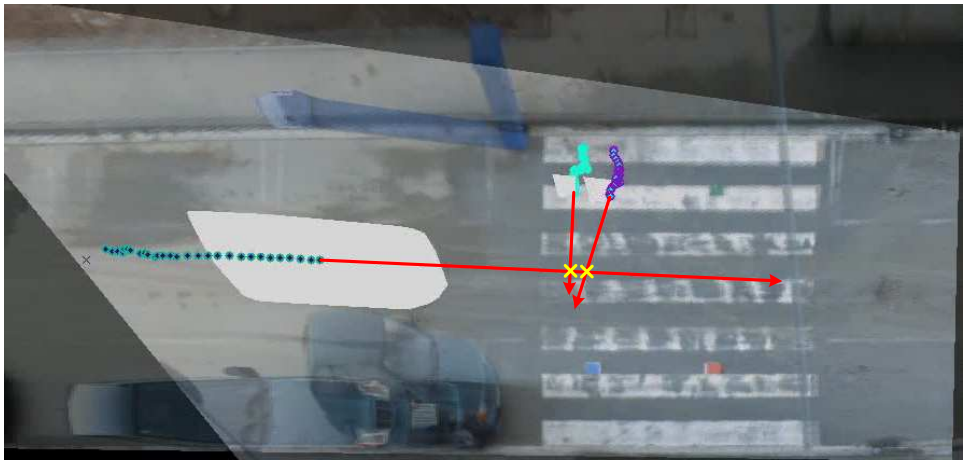
The following view invariant analysis module demonstrates the applicability of the system. The examples used for demonstration show potentially dangerous events. The examples are based on unchoreographed sequences from both the wide and narrow view configuration of the Matthews Lane dataset. Since unchoreographed sequences are used, no real dangerous events occur. The defined events are therefore very coarse and frequently occur in the sequences. Three events are defined to trigger an alarm:

Human and vehicle on road This event triggers an alarm if both a human and vehicle are on the road a the same time.

Vehicle stopped on road This alarm is triggered by a vehicle stopping on the road. Since vehicles should be allowed to park at the road side, the alarm is only triggered in a predefined area of the middle of the road.

(a) View 1

(b) View 2



(c) Virtual view

FIGURE 9.2: Collision detection in the HERMES dataset. A red arrow depicts the extended velocity vectors of the tracked objects, and the yellow crosses show intersections as an indication of a possible collision. Additionally, the distance between the front of the car and the nearest pedestrian can be measured as decreasing, which further indicates the possibility of a collision.

Speeding human To demonstrate the system's ability for speed detection, an alarm is triggered in the event of a speeding human. The speed limit is set to 15 km/h meaning that mainly bicyclists and skateboarders are detected. The speed is computed from the velocity vector. The speeding human event is chosen in the absence of speeding vehicles in the available datasets.

The purpose of these coarse alarms is to direct the attention of a human operator. Therefore, it is relevant to define an interface for a human operator that is intended to ease the task of monitoring traffic. The interface of the view invariant analysis module shows the camera inputs for the system and a top-down view of the given scene. If a dangerous event occurs an alarm is raised. When an alarm is raised all objects involved in the event is marked by a red bounding box in the camera inputs. This information is only shown if an alarm is active in order to avoid tiring or distracting the user unnecessarily. Furthermore, it is possible to hide the identity of objects by disabling the camera inputs and only present the top-down view representation for the user.

The same view invariant analysis is tested on several different sequences, and the outputs are available on the DVD (◎ */tests/evaluation/*). Figure 9.3 and Figure 9.4 show examples of alarms being triggered in the module interface by view invariant analysis. The figure text explains the depicted scenes and alarms.

In addition to testing with recorded data sequence, an online version of the system is made. This version receives live input from the cameras on Matthews Lane, and with few objects in the scene it runs near real-time on a 3Ghz computer with 2GB ram. For testing the system live track refinement (see Section 6.6.2 on page 79) is disabled since this is a computational expensive process. This affects the robustness of the system, but it is not essential for running the system. If several objects enters the scene, the performance drops which reduces the frame rate. The result of this is that tracks are more easily lost and that the Codebook background subtraction has difficulty adapting to rapid illumination changes. However, when the system runs near real-time the results of the online test is comparable to the overall test results.

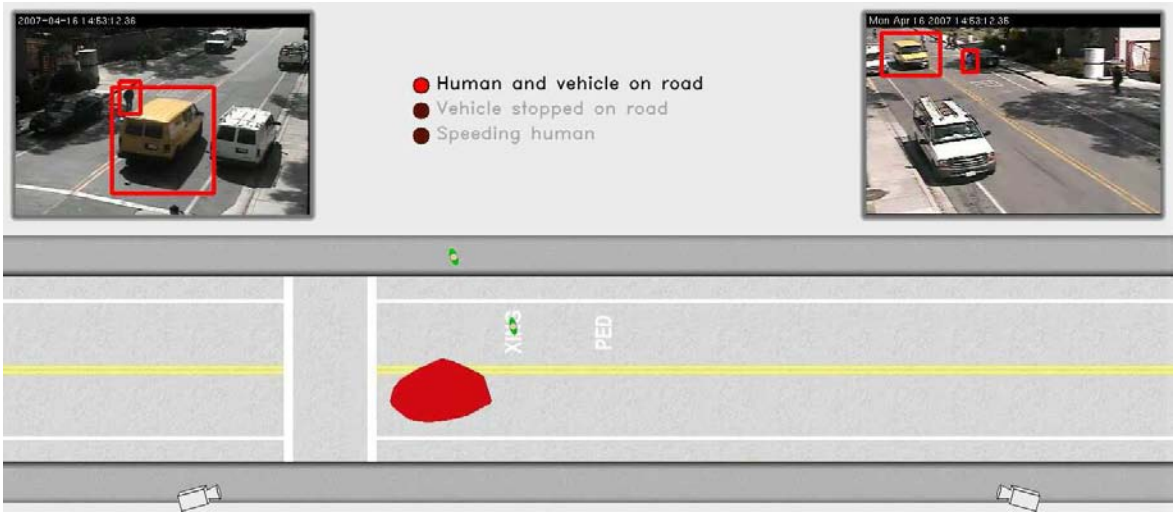


FIGURE 9.3: Recognition of events in the wide view configuration of the Matthews Lane dataset. A "human and vehicle on the road" alarm is raised by the yellow van passing a human crossing the road. The person is difficult to see with the human eye due to the shadow of the trees, but he is detected correctly in both views by the system. Note that, in the top-down view, a human is walking on the sidewalk, but he is not marked by a red bounding box in the alarm since he is not part of the event i.e. on the road.

9.4 Evaluation

For demonstrating the view invariant analysis, interesting sequences that contain the defined dangerous events have been selected. The view invariant analysis preforms well in these sequences, both by recognizing the defined events, but also by not raising false alarms when no events occur. The overall system errors defined in Section 8.2 on page 102 can make the view invariant analysis miss dangerous events or raise false alarms. However, an system error that cause an object only to be partially tracked would still in many cases result in correct event recognition. In general, if only a part of the object's trajectory is available, it is still

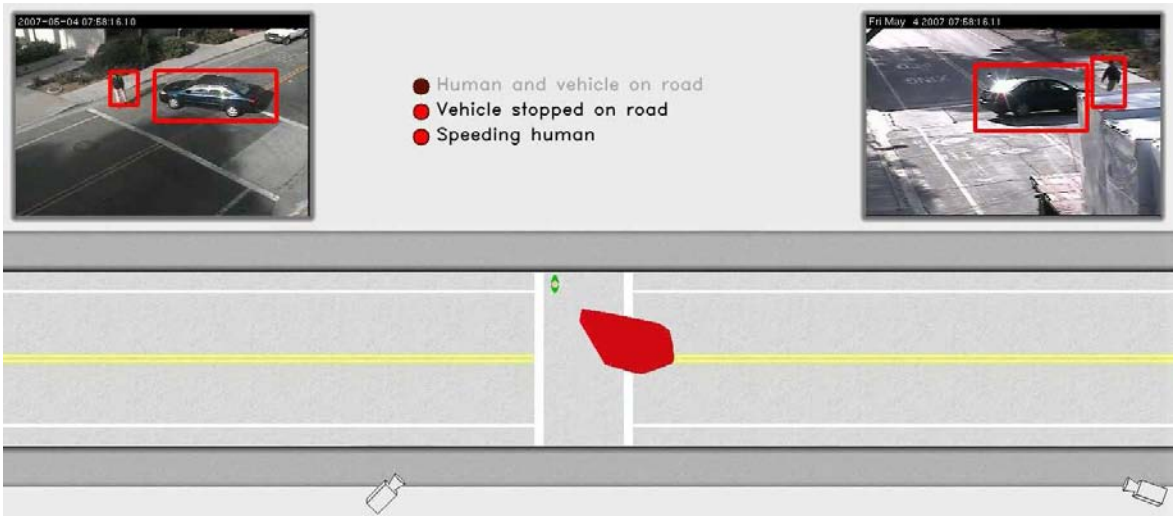


FIGURE 9.4: Recognition of events in the narrow view configuration of the Matthews Lane dataset. Two alarms are activated in the scene. A car is stopping on the middle of the road to perform a three-point turn and therefore triggers the "vehicle stopped on road" alarm. At the same time a skateboarder drives through the scene with an estimated speed above 15 km/h, which triggers the "speeding human" alarm. The "human and vehicle on road" alarm is not triggered since the skateboarder is outside the defined road area. Note that the bounding box of the skateboarder is correct even though he is occluded by a parked vehicle in the rightmost view.

possible to project the tracking information and establish a high level of situational awareness.

The events recognized by the view invariant analysis module is rather basic, since this work has focused more on the foundation for the view invariant analysis than the actual analysis. However, the demonstrations show the applicability of the system and more comprehensive analysis could be applied. For instance, from the position and velocity a movement pattern could be derived, e.g. if a pedestrian is walking straight on the sidewalk for a period and then starts turning towards the road, it could indicate that the pedestrian is about to cross the road. Also, in [Park and Trivedi, 2006] a temporal-spatial activity space is defined around objects using position, size, velocity and the context the objects are in. The context is considered since the personal space is larger in open areas than in crowded areas according to social psychology. From this activity space, schematic event grammar is used to describe spatial-temporal relationships between person and vehicle tracks. This is used to enhance situational awareness for application such as collision detection or people behavior control. These types of behavior analysis could be applied on top of the developed system.

Based on the results of the overall system test, the system developed in this work contributes with robust multi-view correspondence of both humans and vehicles. This provide a better basis for representing both object types in the view invariant domain compared to related work. The representations produced by the system greatly simplify the task of event recognition, since each object representation holds the parameters needed for view invariant analysis. In the end, this leads to a solid foundation for gaining situational awareness in traffic environments.

Additionally, the system natively holds a view variant and view invariant representation of objects. This is useful because the view variant domain allow extraction of meaningful object images which the view invariant does not. This ability is important for object identification e.g. reading the license plate of a car or saving a photo description of a person.

9.5 Summary

The main findings from the evaluation of the system using view invariant analysis are:

- The system provides a foundation for establishing situational awareness based on tracking of both humans and vehicles.
- The system natively holds both a view variant and view invariant representation of objects that eases the task of event recognition and object identification.
- The view invariant analysis module demonstrates the applicability of the system by recognizing potentially dangerous events and presenting alarms for an external user.

Chapter 10

Conclusion

The following presents the conclusion of the thesis based on the thesis definition. This includes presentation of the main contributions and developed methods of the thesis. The chapter is ended with an outlook and suggestions for future work.

10.1 Conclusion

In the introduction, the initial problem is formulated as "How can computer vision-based visual surveillance be applied in order to increase traffic safety?". As a first step in answering this question, the problem analysis states that a visual surveillance system should incorporate situational awareness in order to detect and predict accidents. This is complicated in problem areas such as busy urban locations with humans and vehicles as the two main object types. Situational awareness must be obtained from the problem area and include understanding of both object types and their interaction.

Very little work has been done on traffic monitoring of both humans and vehicles since most related work focus on either humans or vehicles only. Also, from the related work it is clear that the use of multiple cameras is a significant help for resolving challenges such as occlusion. The main goal in the thesis is therefore "to enhance automatic situational awareness by building a system capable of robustly tracking humans and vehicles through their activities and interaction in an unconstrained outdoor environment using multiple surveillance cameras".

The thesis presents a complete system for visual surveillance. The system is divided into modules where each module handles a significant stage in a computer vision framework, and each module receives input from a lower level module in the system.

A major problem of all surveillance systems that monitor unconstrained environments is occlusion, and solving this issue using a single camera setup is an almost impossible task. The use of multiple views is therefore essential within the problem area of this work. It does not only help to resolve occlusion, but it also improves the accuracy with which the position and size of objects can be determined.

When multiple views are used they must be registered in order to co-operate. The camera registration is based solely on homography mapping. This makes the camera registration process easier compared to systems using full camera calibration. This also makes the system independent of world coordinates, in case view invariant analysis is not required. As an additional property of the system, the methods for correspondence of objects between views are chosen and developed to be robust towards inaccuracies in the homography mapping since

perfect plane-to-plane mapping can not be guarantied in most realistic setups.

The foreground segmentation do not perform as well as in related work. However, the cause of this is mainly due to the length and complexity of the utilized dataset. A 78 hour sequence of the Matthews Lane dataset is used for testing. For this sequence, foreground segmentation is shown to perform consistently. A significant cause of segmentation errors is due to background camouflage of objects. A dedicated shadow suppression module is introduced to improve the foreground segmentation. Two shadow suppression methods are tested, but not found to perform well enough.

In reviewing multi-view surveillance systems a method based on the principal axis and a method based on the footage region stand out as prominent methods for doing correspondence between views of humans and vehicles, respectively. In this work, these two methods are combined by extending the existing methods. One of the main contributions of this work is therefore methods for doing robust multi-view correspondence of both humans and vehicles.

For vehicles, a scheme for handling several occlusion scenarios using multi-view correspondence is presented. The occlusion handling includes development of a new method for determining vehicle correspondence using a plausible ground point when no correspondence history is available e.g. at initialization. Additionally, a method for reconstructing vehicles under severe occlusion using probabilistic appearance models is presented.

For humans, the principal axis method is extended to handle groups, since this is not done by the existing principal axis method. The problem is that inter-object occlusion can cause a group to be tracked as a single human. This is solved by a modified correspondence algorithm, that do correspondence of humans which are left unpaired by the original algorithm. This allows accurate location of individuals in the groups, both in the view variant and view invariant domain.

Humans and vehicles are handled differently, but for both a view invariant representation of location, size and velocity is produced. The view invariant representation is essential for doing proper behavior analysis and understanding the interaction of objects, since it represents objects in a domain where the relations between object are meaningful. Furthermore, the view invariant representation means that the same analysis can be applied for any camera setup.

The system's potential for providing a base for obtaining situational awareness is shown using a view invariant analysis module. It is shown that the system is capable of predicting dangerous situations like collisions between human and vehicle objects. This module also demonstrates an interface for an external user e.g. a human operator monitoring traffic. The interface directs the attention of the user by raising alarms when dangerous events occur.

What sets this work apart from most of the related work is the extent and complexity of the data used for testing. The system is tested with several sequences of unchoreographed traffic environments. In total, the 385 minutes of test data used for the overall system test contains 1351 humans and 267 vehicles within the shared region of the two cameras. Furthermore, the system is shown to work in both wide and narrow view camera configurations.

From the overall system test five overall system errors with varying degree of seriousness are defined. The test shows that 67.9 percent of the object are tracked without any errors. Furthermore, by not considering the least serious error, but also most occurring error, the percentage of correctly tracked objects improves to 88.2 percent. The least serious error is

when an object is not tracked for the full duration of its stay within the shared region. At module level, most errors are caused by poor foreground segmentation. This leads to wrong object classification and track initialization.

By building a multi-view system capable of tracking humans and vehicles in an unconstrained outdoor environment the main objective of this thesis is met. The system is a significant step towards automatically achieving situational awareness through visual surveillance, but in order for the system to operate in a real world scenarios, further robustness should be built into the system.

10.2 Outlook

The following presents suggestions for future work. Though the higher level modules of the system are robust towards some errors on the lower levels, the most severe lower level errors propagate to the higher levels. This is the case with errors in the foreground segmentation, and it is therefore interesting to improve the foreground segmentation module. A suggestion for improving the segmentation could be by introducing a feedback from the tracking module, e.g. by increasing the sensitivity of the codebooks in areas where objects are predicted to be located. It would also be interesting to test the system using uncompressed inputs as experimental test shows that this can improve the foreground segmentation.

The tests show poor performance for groups of more than six humans and crowds are in general not handled properly by the system. Because of this, it is interesting to look at new ways for handling large groups and crowds. A solution could be to use the footage region for groups or crowds that are too large or dense to be tracked using the principal axis method.

The system does not run near real-time with several objects in the scene since it runs as a single-thread application. However, by running some modules separately while disabling others, it is determined that each module can run near real-time. In future work, the modular system structure would therefore allow the system to be distributed on several computers for near real-time processing.

The detection of an accident is a prerequisite in order to take the adequate measures in avoiding the accident. However, the actual accident prevention has not been addressed in this work. As a next step, it would be interesting to look at ways of alerting the involved parts. Road signs with build-in speed control that signal drivers exceeding the speed limit already exist [Hillerød Kommune, 2004], and solar powered markers on the road are already in use. Similarly, one could imagine that markers on the road or poles at the road side could flash a yellow light if an accident is about to happen. Furthermore, the increasing amount of technology in vehicles and recent work with in-vehicle warning systems [Aalborg University, 2007] indicate that it in the future will be possible to send warnings signals to vehicles to alert the driver about a potential collision ahead. These solutions only mention a way of alerting the involved parts, but a system that more actively intervenes could also be considered. For instance, if it is possible to communicate with the vehicle it could also be possible to control it and e.g. take control to bring the vehicle to a stop or steer around pedestrians in an emergency. Also, the system could control traffic lights and e.g. by changing the signal to red in all direction if a speeding or possible drunk driver is approaching the intersection. With the developed system as foundation, such solutions could be used for increasing traffic safety.

Appendix A

Review of Miscellaneous Methods

This appendix contains reviews of the relevant literature on background modelling methods, shadow suppression methods and tracking methods.

A.1 Background Modelling Methods

In Section 5.2.1 on page 40, the Mixture of Gaussians (MoG) and Codebook method is presented as solutions for background modelling. The following describes and analyzes six additional methods for background modelling. As pointed out in Section 5.2.1, the actual difficulties lie in creating and maintaining a robust background model. Therefore, the following lists some of the most prominent background modelling methods found in the literature.

Unimodal A single Gaussian is a simple, unimodal way of modelling the variation in color and intensity for a pixel. This method is employed in the Pfinder algorithm [Wren et al., 1997] and works acceptably within constrained areas. It is only capable of modelling a very static background such as an indoor office. Despite the use of the YUV color space making it more robust towards changes in light, this approach is not suitable for dynamic environments such as outdoor scenes. Also, a pertubation test in [Chalidabhongse et al., 2003] rules out this method in favor of the Codebook method.

Bimodal The W^4 system [Haritaoglu et al., 2000] uses what they call a bimodal background model for outdoor environments. The minimum and maximum intensities and the largest allowable intensity difference between two consecutive frames, determined over a training period, constitutes a computationally fast method. To model e.g. a waving tree branch in front of a light sky, variation around the minimum and maximum values is allowed. To update the background model, W^4 employs a "change map" keeping track of e.g. how often the pixels are classified as foreground and background. However, for modelling the dynamic environments used in this work, this method is insufficient.

Median filter The median filtering method is a very simple idea, based on the assumption that a pixel is background more than half the time. The current background model is then based on the N latest frames. An example of its use can be seen in [Cucchiara et al., 2003]. The size of N then uniquely determines the applicability of the model. If N is too large, adaptation is too slow, and if N is too small, it cannot be ensured that the majority of the pixels belong to the background. Instead of calculating the median every time, optimized methods exist to approximate the median recursively, making this method less computationally demanding. A survey in

[Sen-Ching et al., 2004] rules out the median filter, also with approximation, in favor of the MoG method.

Kernel estimation method In [Elgammal et al., 2000], a non-parametric way of building a background model that uses the very recent history information is presented. A kernel estimator is used to estimate the probability density function that a pixel has a distinct intensity value. The kernel estimator function is a multivariate Gaussian density function where independence between the different color channels is assumed. It is claimed that this generalization of the Gaussian mixture model allows for more accurate estimation of the actual density function by only concentrating on very recent information and "forgetting" about the past. A pertubation test in [Chalidabhongse et al., 2003] rules out this method in favor of the Codebook method.

Kalman filter A rather different approach can be taken by using a Kalman filter to predict the pixel value. The prediction is based on a number of previous frames. The idea is that the prediction is based on background pixels and if the real value is far from the prediction, a foreground object is likely to occupy that pixel. Reversely, if the prediction is not far from being correct, a background pixel has been observed. A possible issue could be that if a foreground object occupies a pixel for a long time, the prediction might be based on this instead of the background, resulting in a false positive recognition of foreground. In [Zhong and Sclaroff, 2003], where this method is proposed, the state prediction is only based on a single previous state. However, the survey in [Sen-Ching et al., 2004] rules out the Kalman filter in favor of the MoG method.

Eigen backgrounds The method of using eigen backgrounds for background subtraction is based on a series of images taken in different situations, spanning the scope of the application [Oliver et al., 2000]. These images are converted to an eigenspace image background model. The construction of this model can be done with foreground objects present, as these are not present for a long time and are typically small. The dimensionality of the model is reduced by the use of Principal Component Analysis to keep the most information expressed in variance. The resulting eigenspace is then easy to use to describe the static part of the scene as a weighted sum of the eigenspace basis vectors. It is contrarily hard to describe the portions of the image containing foreground objects by this eigenspace model and a threshold can be set to separate foreground and background pixels. As the eigen background method requires that representative pictures are taken of the scene, this can not be used for highly dynamic and changing scenes.

A.2 Shadow Suppression Methods

In Section 5.8.1 on page 51, two methods are presented as the most interesting. However, several other methods exist for shadow suppression, and a review of these are presented in the following.

A commonly used approach to shadow suppression is segmentation based on color and intensity information, which is emphasized by [Prati et al., 2003]. [Prati et al., 2003] presents a survey of moving shadow detection approaches along with a comparison of four selected methods that are representative for different overall approaches in the reviewed work. The selected

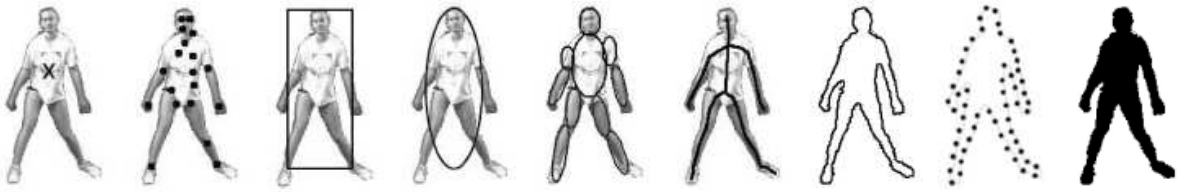
methods cover deterministic and statistic methods for determining class membership, where the deterministic methods covers two non-model based methods, and the statistical methods includes both a parametric and non-parametric method. Of the four compared methods, a deterministic non-model based method from [Cucchiara et al., 2001] is deemed the best general-purpose solution. Some of the other compared methods can out-perform the selected method in certain restricted scenes, like indoors or assuming low object speed, but in an unconstrained scene it is not possible to apply any of those assumptions. The selected method relies solely on comparing the current foreground with the background using an HSV threshold scheme. However, color and intensity are seldom by itself enough to do proper detection of cast shadow, since self shadow and dark objects in many cases are close to inseparable from cast shadow in the HSV color space. Therefore the following analysis reviews a number of approaches that rely on more than just color and intensity information. Also, the methods reviewed in the following are either newer than or not reviewed in [Prati et al., 2003].

Auxiliary scene information The method in [Zhao and Nevatia, 2004] uses knowledge about the sun's position to project an ellipsoid modelling a human onto the ground plane. The projection of the ellipsoid is an ellipse on the ground plane in which shadow is segmented using an intensity threshold. In this way, information about the time of day, date and geographical location is utilized for suppressing shadow. To utilize the sun's position, its exact position relative to the position of the cameras is needed, which cannot be derived without knowledge of 3D world coordinates. The method therefore conflicts with the goal of not being dependent on full camera calibration as stated in Section 2.7.1 on page 25.

Appearance models The multi-view geometry method in [Keck et al., 2006] is a simplified method of the work presented in [Jeong and Jaynes, 2005]. In addition to [Keck et al., 2006], [Jeong and Jaynes, 2005] applies an appearance model for the shadow pixels. The shadow appearance model is build online from a mixture of Gaussians when an object enters the scene. Though effective, the appearance model approach is only proven to work under restrictive conditions such as indoors with narrow views and uniform colored surfaces.

Texture and regions The work in [Javed and Shah, 2002] utilizes that shadow regions retain some of the texture and color information of the background. The method works stepwise with the goal of classifying pixels into cast shadow, self shadow and dark object regions by comparison with the background. Potential shadow is found as regions with low intensity, followed by K-means color segmentation that divides potential shadow into the three region types. Finally, the cast shadow is separated from the other regions by a gradient approach that compares the texture of the regions with the background. In [Javed and Shah, 2002] they report difficulty with separating cast shadow and self shadow, causing object segments to be removed in 5 percent of the frames. In [Javed and Shah, 2002] shadows are removed in 70 percent of the frames, and in 25 percent no shadow is removed.

Model based As mentioned earlier, [Zhao and Nevatia, 2004] exploit ellipsoid models of humans for shadow suppression. In [Koller et al., 1993] geometric 3D vehicle models are



(a) (b) (c) (d) (e) (f) (g) (h) (i)

FIGURE A.1: Object representations. (a) Centroid. (b) Multiple points. (c) Bounding box. (d) Bounding ellipse. (e) Part-based multiple patches. (f) Object skeleton. (g) Complete object contour. (h) Control points on object contour. (i) Object silhouette. Figure from [Yilmaz et al., 2006].

applied. The vehicle models are projected onto the ground plane assuming parallel incoming light which is set manually. In this way, edge segments of the shadow can be included in the vehicle model for a better match. According to [Prati et al., 2003] a model based approach for shadow suppression yields better results than a non-model based. However, the model based is often too complex and time consuming compared to the non-model based [Prati et al., 2003].

A.3 Tracking Methods

The overview of tracking approaches given in Section 6.1.3 on page 67 only contains the main points related to tracking method. The following describes the methods in greater detail.

A bottom-up approach is followed in describing the issues related to tracking based on the survey in [Yilmaz et al., 2006]. The first issue is selecting a suitable representation of the object. The next issue is to select the image features used to track the object. At the top level, general tracking approaches are reviewed.

A.3.1 Object Representation

The object representation is the lowest level in tracking and can be divided into shape and appearance of objects. A strong relationship exists between the object representation and the general tracking approach. The shape representations are described first, followed by a description of the appearance representations.

Points The object is represented by a single point or multiple points. This representation is suitable for tracking objects that occupy a small region in the image. Examples are shown in Figure A.1(a) and A.1(b).

Primitive geometric shapes The shape of the object is represented using e.g. a rectangle or an ellipse. Using this representation the motion of the object is modelled by translation, affine or projective transformation. Primitive geometric shapes have been applied in representing both rigid and non-rigid objects. Examples are shown in Figure A.1(c) and A.1(d).

Articulated shape models The human body consists of body parts held together with joints. To represent an articulated object, the body parts can be modelled using cylinders or ellipses as shown in Figure A.1(e), and the relationship between the body parts is governed by a kinematic model. This representation is not well suited for representing vehicles.

Skeletal models The object skeleton is often used to recognize objects, but is also capable of modelling both articulated objects and rigid objects. An example of the object skeleton extracted using the medial axis transform is shown in Figure A.1(f).

Object silhouette and contour The object boundary is defined using a contour representation, and two examples are shown in Figure A.1(g) and A.1(h). The area within this boundary is the object silhouette, and is shown in Figure A.1(i). Both the silhouette and contour representations are well suited for modelling complex non-rigid objects.

As with the shape of the objects a number of approaches exist to represent the appearance of the objects. Some representations combine both the shape and appearance of the objects. Common appearance representations are listed next.

Probability densities of object appearance Appearance features could be color or texture, and the probability densities can be derived from the image region given by the interior of the shape model, e.g. interior of a bounding box or a contour. The estimate of the probability density can be either parametric (e.g. Gaussian or mixture of Gaussians) or non-parametric (e.g. Parzen windows or histograms).

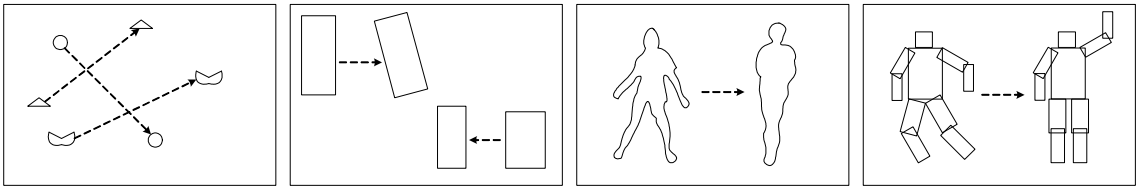
Templates The appearance template is formed using a simple geometric shape or the silhouette. Both the spatial and appearance information are encoded in the template. However, the object appearance is generated using only a single 2D view of the object. This may restrict the appearance template to only be used for objects whose poses do not vary considerably during movement.

Active appearance models In this representation both the shape and appearance are modelled simultaneously. In general, the object shape is defined by a number of landmarks, where each landmark contains an appearance vector representing color, texture or gradient magnitude. However, active appearance models need a training phase and are highly sensitive to the initialization of tracking, making it difficult to start tracking automatically [Hu et al., 2004b].

Multi-view appearance models As opposed to templates, these models are generated using different views of an object, e.g. views with different object poses or different illumination. To reduce the dimensionality of the different object views, a subspace is generated using e.g. Principal Component Analysis or Independent Component Analysis. This approach requires a training phase like the active appearance models.

A.3.2 Feature Selection

Selecting the features to use in the tracking of objects is closely related to the object representation; e.g. color features are used in probability densities of object appearance, and object



(a) (b) (c) (d)

FIGURE A.2: General tracking approaches. (a) Point tracking with multiple point association. (b) Kernel tracking using a rectangular representation. (c) Silhouette tracking illustrated by a contour representation. (d) Body model tracking using rectangular body parts.

edges are used as features for contour-based representations. It is desired to use features that make objects easy to distinguish in the feature space. Often features are combined to track the objects. Some common visual features are listed in the following.

Color Color is often a simple feature to distinguish objects, e.g. a red car versus a white car. The apparent object color depends mainly on the spectral power distribution of the illuminant and the object's surface reflectance properties. The apparent color is sensitive to illumination changes. However, some color spaces are able to reduce the sensitivity towards illumination changes.

Edges Boundary tracking algorithms often use edge features. The object boundary generally causes strong changes in image intensities. Edge detection methods (and especially the Canny Edge Detector) are often used to extract the edge features. Compared to color features, edge features are more insensitive to illumination changes.

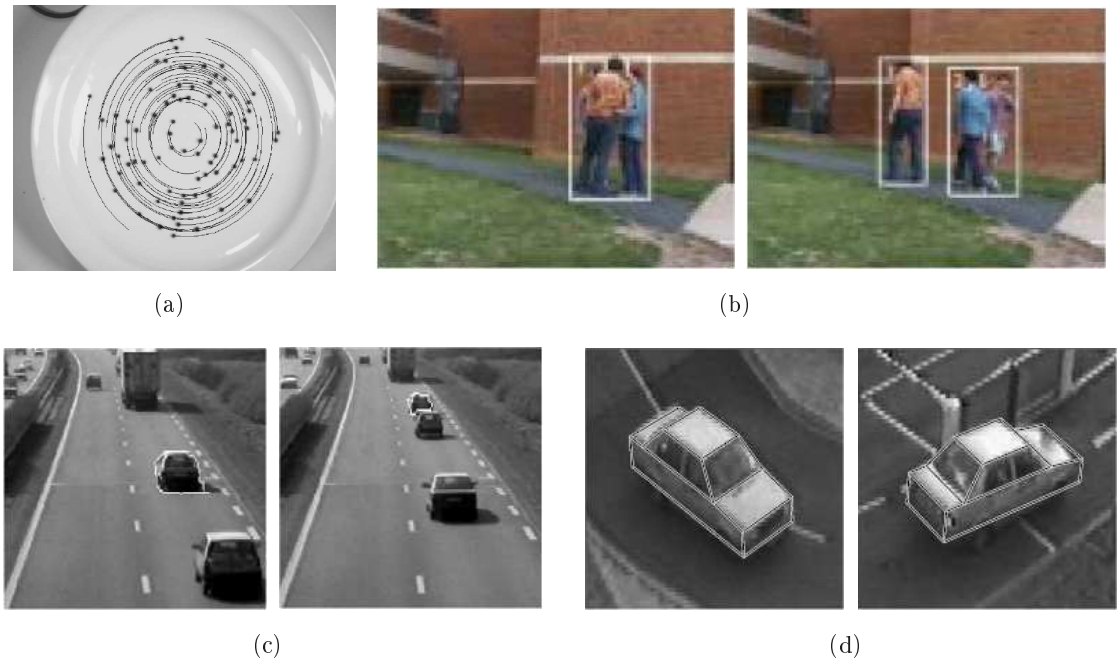
Optical flow Optical flow is often used as a motion segmentation method, but can also be applied in object tracking. In some cases the optical flow-based motion segmentation is combined with the tracking. Optical flow is a dense field of displacement vectors which defines the translation of each pixel in a region.

Texture Texture features are based on the intensity variation of the object surface. It can be seen as a measure of surface smoothness and regularity. To extract the texture features, a descriptor such as the Gray-Level Coocurrence Matrix is required. Texture features are less sensitive than color features towards illumination changes.

A.3.3 General Tracking Approach

In the following, four main tracking categories are identified. They are highly dependent on the chosen object representation and feature selection, and thus the applicability of the tracking method depends on the type of objects to be tracked. The four general approaches to tracking are shown in Figure A.2 and are described in the following. For each general tracking approach some examples of work using the approach are given.

Point tracking The detected objects are represented using a single or multiple points. The association between points in consecutive frames is based on the previous object state.



(a) (b)

(c) (d)

FIGURE A.3: Tracking results. (a) Tracking black dots on rotating dish [Veenman et al., 2001]. (b) Tracking a group of interacting people [McKenna et al., 2000]. (c) Tracking a vehicle on highway using the contour [Mansouri, 2002]. (d) Tracking a vehicle using a 3D wire-frame model [Koller et al., 1993].

The object state could include the position and motion of the object. Point tracking becomes complicated during occlusion of the tracked points, misdetection, entries and exits of objects. To overcome these issues two approaches for point association can be identified. The first is deterministic methods for point association where global motion constraints are applied. The second uses statistical methods for correspondence, where specific tools could be Kalman filters, particle filters and multiple hypothesis tracking. An example of association of multiple points is shown in Figure A.2(a). [Veenman et al., 2001] is a point tracking method, and a tracking result using this method is shown in Figure A.3(a).

Kernel tracking In this context, kernel refers to the object shape and appearance; e.g. a rectangular shape and a color histogram describing the appearance within the rectangle. Parametric motion or dense flow field are often used to model the object motion. The algorithms based on kernels differ mainly on the applied appearance representation. The most occurring appearance representations are in this context templates, probability densities and multi-view appearance models. Templates and probability density-based representations are popular due to their simplicity and computational efficiency. However, the appearance models are based only on the recent single view observations of the object. The appearance of the object may change dramatically during tracking, and this is build into the multi-view appearance representation. The multi-view appearance representations must be learned offline. Kernel tracking is illustrated in Figure A.2(b). [McKenna et al., 2000] is a kernel tracking method for humans, and a tracking result

using this method is shown in Figure A.3(b).

Silhouette tracking Silhouette tracking methods utilizes the information contained within the object boundary. Appearance density and shape models are usually used for representing this information. Often edge maps are applied as the feature. Depending on the chosen object model, either shape matching or contour evolution is used for tracking the object. Trackers based on silhouettes provide an accurate shape description of the object. This description can be utilized at levels at a higher abstraction level to e.g. understand the object behavior. Objects with complex shapes such as hands, head and shoulders are typical examples of objects where silhouette tracking is applied. As mentioned earlier, these methods often need a training phase and are highly sensitive to initialization. An example showing contour evolution is given in Figure A.2(c). [Mansouri, 2002] is a silhouette tracking method that only requires little a priori knowledge about the object being tracked. A tracking result for vehicles using this method is shown in Figure A.3(c).

Body model tracking In body model tracking the "body" of the object is modelled. The object is tracked by projecting the model into the image plane and compared with the image data. This is referred to as a predict-match-update scheme. The models are constructed offline and incorporate prior knowledge of the object body. According to [Hu et al., 2004b], a significant difference exists between human body tracking (non-rigid object tracking) and vehicle tracking (rigid object tracking). In the vehicle case, a 3D wire-frame vehicle model is often used. When modelling the human body the geometric structure can be represented using e.g. stick figure, 2D contour, volumetric model or hierarchical model. The main challenges are to construct the body model, find a representation of prior knowledge of motion models and motion constraints and finally to select a prediction and search strategy. The model based tracking algorithms obtain good results during occlusion (including self-occlusion for human objects). An example of a human body model using rectangular patches for body parts is illustrated in Figure A.2(d). [Koller et al., 1993] is a body model tracking method where vehicles are tracked using 3D wire-frame as shown in Figure A.3(d).

Appendix B

Shadow Suppression

The following documents the design and test of a shadow suppression method based on multi-view geometry.

B.1 Multi-View Geometry Approach

In Section 5.8.1 on page 51 it was decided to implement and test a multi-view geometry approach based on [Keck et al., 2006] since the method is new and relatively untested. The method is interesting because it exploits the benefit of having multiple views. The work in [Keck et al., 2006] presents a method that exploits the benefit of having multiple views and is based on work presented in [Jeong and Jaynes, 2005].

B.1.1 Conceptual Design

Like many other shadow suppression algorithms this method relies on comparing the color of a pixel. But rather than comparing pixels to a background model or image, it compares pixels between two views. The method does not specifically detect shadow, but separates object pixels from non-object pixels. The method relies on the following assumptions to remove shadow from the existing foreground mask:

- The color difference of pixels warped between the two views is close to zero when neither of the two pixels are covered by a foreground object.
- Objects move on a planer surface.
- The field of view of the cameras overlap.

The difference between a pixel and its mating pixel in the other view is calculated by a color similarity measure. Pixels of a shadow and background pixels are assumed to have low difference while objects have a high difference due to the parallax of the views. An example of three mapped points is shown in Figure B.1. By comparing pixels within the foreground mask generated by the Codebook method, it is possible to separate shadow and object pixels based on the calculated similarity measure. The mapping between views is handled by planar homography mapping, as explained in Section 4.2 on page 31.

FIGURE B.1: Three points warped using the homography. The multi-view shadow suppresion assumes that there is a low color difference between points that are on the ground plane (the red and green points) and a high difference for pixels mapping to an off-plane object (the yellow point).

B.1.2 Illumination Balancing

In order to make the method more invariant to illumination differences between views due to e.g. auto gain of the cameras, the illumination is balanced using the ratio between the average illuminations of the overlapping field of views in the two images. The effect of the illumination balancing can be seen in Figure B.2, and the overlapping field of view between the cameras is depicted in Figure B.3.

It should be noted that the use of field of view for illumination balancing is not always valid due to the perspective of the cameras, e.g. in Figure B.2 the shadow of the trees are much bigger in the right view than the left due to the perspective. This results in slight over compensation, which can be seen in the upper right corner of the adjusted image in Figure B.2. A better solution could be to select a region on the ground plane which has a similar size in the two views. However, the current illumination balancing improves the results, so no further effort is made to improve it.

B.1.3 Similarity Measure

The similarity measure used by [Keck et al., 2006], compares the RGB color components by the L2 Norm:

$$\text{Color difference} = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \tag{B.1}$$

Where R_1, G_1 and B_1 are the color components of a pixel in the first view. R_2, G_2 and B_2 are the color components of the matching pixel in the second view.

Only pixels classified as foreground by the Codebook method are tested for similarity. After computing the similarity measure a threshold is applied to determine if the pixel is in shadow or not; values below the threshold is classified as shadow. In the current work, the threshold is set to 90 (The theoretical maximal difference is 441). This is a deterministic method, and an attempt to improve the method by unimodal modelling of the difference was applied, but this did not yield an improvement.

(a) Original camera input

(b) Balanced illumination

FIGURE B.2: Example of illumination balancing. First row: original input images. Second row: the illumination in the right image has been adjusted to match the illumination level of the left image.

FIGURE B.3: The bright area shows the overlapping field of view between the cameras.

B.1.4 Results and Discussion

In the following, a discussion of the results obtained from experimental testing of the multi-view geometry method is presented. The observations is based on more than 90 minutes of recordings.

An example of the shadow segmentation by the method is given in Figure B.4. In the left view too much shadow are removed. This is due to two issues; first, he (right person) is wearing a grayish shirt and grayish backpack which is compared to the gray road in the other view; secondly, her dark hair (left person) is overlapping with the shadow in the other view because she is occluding his shadow. Both issues causes foreground objects to be wrongly removed as shadow. In the right view, the shadow segmentation performs well. Almost the complete shadow is removed without remove parts of the foreground objects. In many cases the method does not only remove shadow but also noise in general. Furthermore, since the method is not based on temporal information like the Codebook method, it reacts instantly to sudden noise e.g. caused by lighting changes.

In general, the method is good for removing strong shadow, since it relies more on parallax rather than color comparison. However, from the output videos of the experimental tests it is

FIGURE B.4: Result of the shadow segmentation. Foreground is marked as yellow and foreground removed by the shadow suppression is marked as red.

evident that the number of removed actual shadow pixels compared to the number of removed foreground object pixels is too low.

The work in both [Keck et al., 2006] and [Jeong and Jaynes, 2005] uses datasets with considerably better camera angels and planar surfaces for homography mapping. From their published data, both papers also appear to have more uniform colored ground plane surfaces than are available in the datasets of this work. This together with the results presented above indicates that the method does not perform well enough to be used.

B.1.5 Summary

The following lists positive and negative sides of using the multi-view geometry method for shadow suppression.

Pros:

- Removes strong shadow.
- Functions as multi-view object detection in addition to shadow suppression.
- Removes noise pixels on the edge of objects.
- Does not rely on temporal information and improves segmentation of scenes with rapid changes.

Cons:

- Requires a very precise mapping between views. This problem is enhanced by perspective distortion; the difference between the sizes of the shadow in the two views is a problem.

- It does not work if the shadow is occluded in one view.
- There is a significant chance of false positive (a part of the foreground object is removed), since warped object points are compared to what the other view contains e.g. other foreground objects or noisy background.
- The parallax assumption does not hold for cars. Due to their size, a point on the car might be warped to another point on the same car.

Appendix C

General Methods

This appendix describes the Kalman filter and least median of squares. The Kalman filter is used for tracking in single view, and the least median of squares is used for fitting the principal axis of humans.

C.1 Kalman Filter

The Kalman filter is a recursive filter that estimates the state of a dynamic system. The estimation is optimal in the sense that it minimizes the estimated error covariance given some conditions, which are listed in the following. This section gives a brief introduction to the discrete Kalman filter. The section is based on [Welch and Bishop, 2001], and the mathematical notation is also based on the same source. For a more detailed description of the Kalman filter refer to [Welch and Bishop, 2001].

The purpose of the Kalman filter is to estimate the state x of a given discrete-time process. The process is assumed to be governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + w_{k-1} \tag{C.1}$$

A represents the model of the process and is referred to as the state transition matrix. The process model is also referred to as the motion model. The noise associated with the process model is expressed by w. Furthermore, there is a measurement model that describes the relationship between the process state and the measurements. The measurement model is given by:

$$z_k = Hx_k + v_k \tag{C.2}$$

The observed measurement is expressed by z_k. H is referred to as the measurement matrix and relates the state x_k with the measurement z_k. The noise associated with the measurement model is expressed by v.

Both the model noise w_k and the measurement noise v_k are assumed to be white, Gaussian distributed and independent of each other. This is expressed by:

$$p(w) \sim N(0, Q) \tag{C.3}$$
$$p(v) \sim N(0, R) \tag{C.4}$$

Q is referred to as the process noise covariance and R as the measurement noise covariance. Both are in the following assumed to be constant over time. This assumption is also used in the developed system.

C.1.1 Computing the Kalman Filter

Before giving the equations for the Kalman filter, some definitions are introduced. In the following, $\hat{x}_k^-$ is defined as the a priori state estimate at time step k given knowledge of the process prior to time step k. $\hat{x}_k$ is defined as the a posteriori state estimate at time step k given measurement z_k. The a priori and a posteriori estimate errors are defined respectively as:

$$e_k^- = x_k - \hat{x}_k^- \quad \text{and} \tag{C.5}$$
$$e_k = x_k - \hat{x}_k \tag{C.6}$$

The a priori estimate error covariance and the a posteriori estimate error covariance then follows respectively as:

$$P_k^- = E\left[e_k^- e_k^{-T}\right] \quad \text{and} \tag{C.7}$$
$$P_k = E\left[e_k e_k^T\right] \tag{C.8}$$

The Kalman filter computes the a posteriori state estimate $\hat{x}_k$ using this equation:

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right) \tag{C.9}$$

$\hat{x}_k$ is computed as a linear combination of the a priori estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$. This difference is also referred to as the residual. A residual of zero means that the predicted measurement and the actual measurement are in agreement. K in Equation C.9 is referred to as the gain and is given by:

$$K = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{C.10}$$

K performs the weighting and can be understood as what is "trusted" more. If the measurement noise covariance R approaches zero, the measurement z_k is trusted more and more, while the predicted measurement $H\hat{x}^-$ is trusted less and less. However, as the a priori estimate error covariance P_k^- approaches zero the actual measurement z_k is trusted less and less, while the predicted measurement $H\hat{x}^-$ is trusted more and more.

C.1.2 The Kalman Filter Algorithm

The Kalman filter works by a predict-correct scheme. In the prediction step, the state is predicted ahead of time. The predicted state is (somehow) matched with actual noisy measurements. The matched measurement is used in the correction step to correct the filter. The

matching of the predicted state with the measurements is not considered here. The cycle is illustrated in Figure C.1.

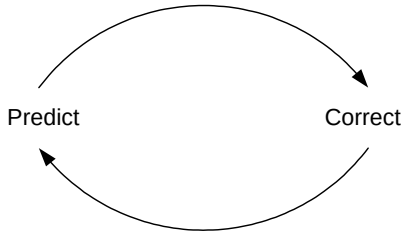


FIGURE C.1: The predict-correct scheme used by the Kalman filter.

In the prediction step, the following is computed:

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{C.11}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{C.12}$$

The purpose is to project the state and covariance estimates forward from time step $k-1$ to time step k.

In the correction step, the following is computed:

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{C.13}$$
$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H\hat{x}_k^-\right) \tag{C.14}$$
$$P_k = \left(I - K_k H\right) P_k^- \tag{C.15}$$

Equation C.13 and C.14 are Equation C.9 and C.10 applied at time step k, respectively. The a posteriori estimate error covariance P_k is also calculated. $\hat{x}_k$ and P_k are then used in the prediction step in the next time step. This is the recursive nature of the filter, which is advantageous with relations to the computational complexity. The complete algorithm is shown in Figure C.2.

The noise for both the process model and the measurement model must be specified. The measurement noise covariance R can usually be measured. On the other hand, the process noise covariance Q needs to be estimated, which generally is more difficult as the estimation from the model can not be observed directly. Furthermore, the state transition matrix A and measurement matrix H must be specified along with an initial estimate of the a posteriori state $\hat{x}_0$ and a posteriori estimate error covariance P_0.

In this work, the process noise covariance and measurement noise covariance are set through a trial and error process. A first-order motion model is used, which dictates the state transition matrix and measurement matrix. All Kalman filters are in this work initialized using the measurement (centroid, ground point or bounding box measurement) and with zero velocity.

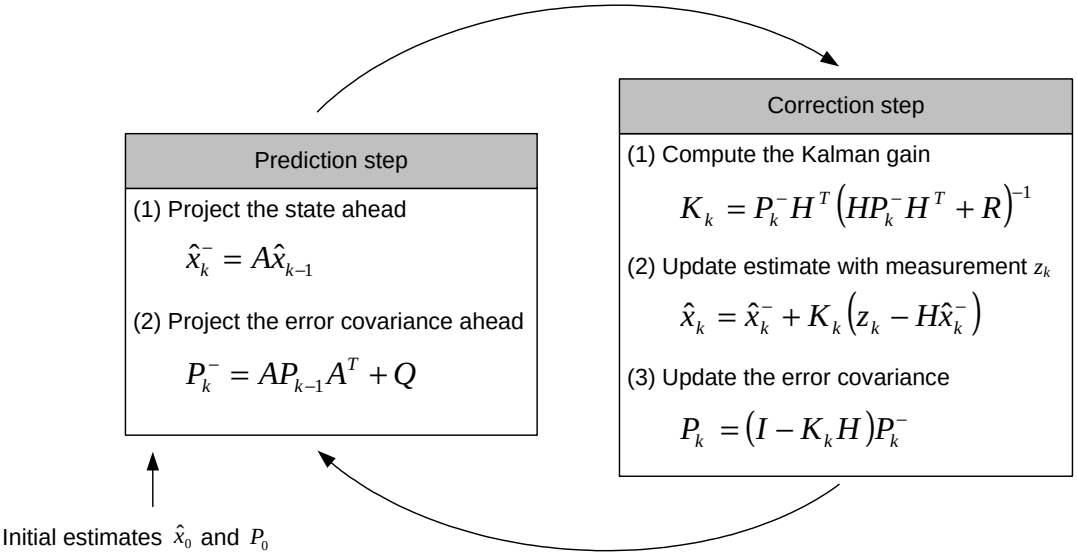


FIGURE C.2: A complete picture of the Kalman filter algorithm.

C.2 Least Median of Squares

The following explains the Least Median of Squares (LMedS) algorithm used for fitting a vertical line for humans. The vertical line is determined from the vertical projection histogram of the foreground mask.

Methods used for fitting a line can be evaluated by their *breakdown point*, which is defined as the smallest percentage of outliers that can cause the fitted line to explode [Barreto, 2001]. This is what separates the LMedS from the Least Mean of Squares (LMS), since LMS has a breakdown point of 0 percent compared to 50 percent for the LMedS [Rousseeuw and Leroy, 1987]. Unlike LMS, LMedS has no close form solution or formula, since the median is an order or rank statistic [Barreto, 2001]. This makes computation of the fitted line complex and a brute force solution is computationally expensive, see Section 7.3 on page 85. LMedS could be implemented using a sorting algorithm, however, since all values are integers, a better and less computationally expensive solution can be found using the histogram [Yang and Levine, 1992].

Let n be the total number of pixels in the histogram and the *range* be a number of bins in the histogram. From [Rousseeuw and Leroy, 1987], it can be shown that solving the LMedS minimization problem is equivalent to finding a sequence of length $\lfloor n/2 \rfloor + 1$ with the minimum range R, where $\lfloor \cdot \rfloor$ denote a floor operation [Yang and Levine, 1992]. More specifically, if $h = \lfloor n/2 \rfloor$ and the sequence with the shortest range is described as $\{x_k \cdots x_{h+k}\}$ where k is pixels in the histogram, then the solution to the minimization problem is given by:

$$L = \frac{x_{h+k} + x_k}{2} \tag{C.16}$$

The minimum range is then given by:

$$R = x_{h+k} - x_k \tag{C.17}$$

An example of a histogram is shown in Figure C.3, where L determines the position of the vertical line fitted by LMedS.

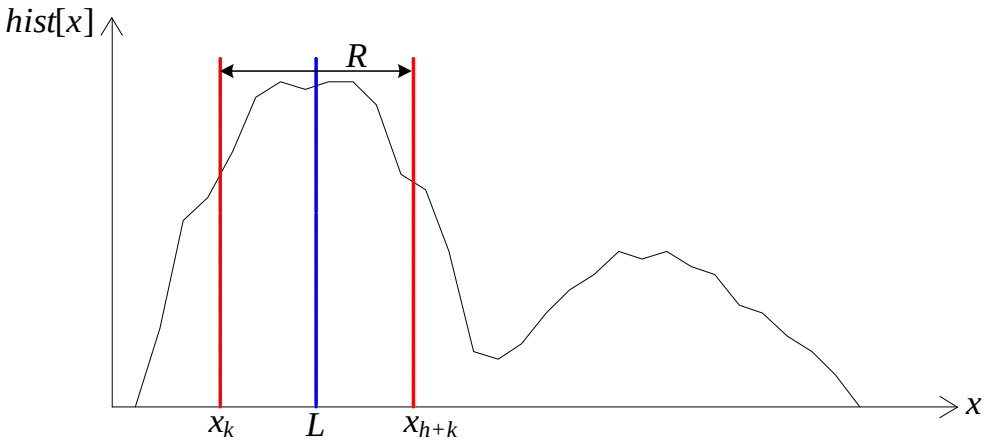


FIGURE C.3: LMedS solution for a histogram. L is the solution. x_k and x_{h+k} defines the shortest range R.

The algorithm used for calculating the LMedS is presented in Listing C.1. The algorithm is derived from pseudo code presented in [Yang and Levine, 1992], but since this code did not seem to provide the correct result, some modifications were made. The listed algorithm outputs two histogram pointers g_1 and g_2 (x_k and x_{h+k}) that minimizes the shortest range R, hence $R = g_1 - g_2$. If there is more than one occurrence of the minimum range R the average solution is used.

LISTING C.1: Least Median of Squares Algorithm

Step 1: Divide the histogram in two
$hist$ = vertical projection of the foreground mask
n = the total sum of pixels in the $hist$
m = the bin that contains $n/2$
N_H = the number of pixels above $n/2$ in m
N_L = the number of pixels below $n/2$ in m

Step 2: Initialize pointers
Two pointers p_1 and p_2 are initialized as follows:
    p_1 points to the first bin in $hist$ i.e $p_1 = 0$
    $p_2 = m + 1$ and $R = p_2 - p_1$

Step 3: Increment p_1 until N_H is zero
$g_1 = p_1$ and $g_2 = m$
while $N_H > 0$
    if $m - p_1 < R$ **then** $R = m - p_1$, $g_1 = p_1$ and $g_2 = m$
    if $hist[p_1] < N_H$ **then** $N_H = N_H - hist[p_1]$ and $hist[p_1] = 0$
    else $hist[p_1] = hist[p_1] - N_H$ and $N_H = 0$
    if $hist[p_1] = 0$ **then** p_1 = the first non−empty bin in $hist$
end

Step 4: Increment p_2
$p_2 = j$ where j is the smallest number larger than p_2 and $hist[j] > 0$
$hist[m] = N_L$

Step 5: Alternately increment p_1 or p_2 while ensuring that R is minimal.
while $p_2 < length(hist)$
    if $p_1 - p_2 < R$ **then** $R = p_2 - p_1, g_1 = p_1$ and $g_2 = p_2$
    if $hist[p_2] > hist[p_1]$ **then** $hist[p_2] = hist[p_2] - hist[p_1]$ and $hist[p_1] = 0$
    else $hist[p_1] = hist[p_1] - hist[p_2]$ **and** $hist[p_2] = 0$
    if $hist[p_1] = 0$ **then** $p_1 =$ the first non−empty bin in hist
    if $hist[p_2] = 0$ **then** $p_2 = j$ where j is the smallest number larger than p_2 \textbf{and} $hist[j] > 0$
end

Appendix D

Datasets

This appendix describes the datasets used to develop and test the system. The data recorded in this work is referred to as the Matthews Lane dataset and is explained first. This is followed by a description of the HERMES dataset. Finally, the PETS 2001 dataset is described.

D.1 Matthews Lane Dataset

The Matthews Lane dataset is a collection of sequences recorded in this work. Matthews Lane is the name of a road located in the campus area of University of California, San Diego. The dataset is recorded using two PTZ cameras mounted on lamp posts at 6.5 meters height. The PTZ cameras have two settings. All sequences are recorded using either a narrow view or a wide view camera configuration of the scene. An aerial view of the scene is shown in Figure D.1(a), and the two camera's locations are indicated by a red dot. Figure D.1(b) shows a street level view of the scene where both cameras are visible. A snapshot from the narrow view camera configuration is shown in Figure D.2. The narrow view camera configuration covers a ground plane area of 10×20 meters. A snapshot from the wide view camera configuration is shown in Figure D.3. The wide view camera configuration covers a ground plane area of 13×32 meters.

A total of 11 sequences are contained in this dataset, and they are listed in Table D.1. The table lists the date and time of the sequence. Furthermore, it lists if the narrow view or wide view camera configuration is used and if the camera's auto setting is enabled or disabled. Initially, it was believed that the auto setting could cause too drastic changes in the illumination, but test proved that it is not the case. Adjusting the cameras manually also proved difficult, and with natural changes in the illumination it is only possible to record sequences of a few hours length. Table D.1 also lists the number of hours in the sequence and the number of frames for each view. The 11 sequences have a total length of 151 hours and 40 minutes. A red star indicates that a part of the sequence is used as a test sequence in the overall system test. Six test sequences are used in the overall system test, which is reported in Chapter 8 on page 101.

The 11 sequences in Table D.1 are "real" surveillance recordings, meaning that no constraints have been put on the objects moving in the scene. However, five sequences not listed in the table are recorded with choreographed data. Four of the sequences are for testing the accuracy of the correspondence module, which is documented in Section 7.5 on page 95. The last sequence is used during development of shadow suppression methods, and examples of

(a) Aerial view

(b) Street view

FIGURE D.1: Matthews Lane: Aerial view and street level view of the scene. The two PTZ cameras are highlighted in the views.

(a) Camera 1

(b) Camera 2

FIGURE D.2: Matthews Lane: The narrow view camera configuration.

(a) Camera 1

(b) Camera 2

FIGURE D.3: Matthews Lane: The wide view camera configuration.

Date	Time	Weekday	Camera conf.	Auto	Hours	Frames
14-04-2007	10:00 - 14:00	Sat	Wide	No	4:10	112,065
15-04-2007	10:00 - 12:00	Sun	Wide	No	2:10	110,573
16-04-2007⋆	13:30-15:30	Mon	Wide	No	2:10	122,989
27-04-2007	09:30-14:00	Fri	Narrow	No	4:40	264,732
29-04-2007	06:30-19:00	Sun	Narrow	Yes	12:40	628,219
02-05-2007	14:50-17:50	Wed	Narrow	Yes	3:10	164,648
03-05-2007⋆	06:00-20:00	Thu	Narrow	Yes	14:10	652,011
04-05-2007⋆⋆	06:00-13:30	Fri	Narrow	Yes	7:40	396,688
05-05-2007	08:00-20:00	Sat	Narrow	Yes	12:10	362,163
06-05-2007	10:00-20:00	Sun	Wide	Yes	10:10	627,671
12-05-2007 - 15-05-2007⋆⋆	14:40 - 21:00	Sat, sun, mon, tue	Wide	Yes	78:30	4,308,191

TABLE D.1: The sequences in the Matthews Lane dataset ordered by date. A red star indicates that a test sequences has been extracted from that sequence to be used in the overall system test. See text for explanation.

the shadow suppression results are given in Section 5.8.5 on page 54.

The camera type used is Pelco Spectra III Series Dome System, and the two cameras are shown in Figure D.4. The cameras are connected to an AXIS 241SA Video Server, and the videos are captured over Ethernet using this video server. Due to this setup and the developed capture software, the achieved frame rate is 15 fps. The resolution is 352×240. The videos have been recorded using a lossy DivX6.5 format at highest quality setting and 780 kbps bitrate.

FIGURE D.4: Matthews Lane: Closeup of the two Pelco PTZ cameras. On top of each Pelco PTZ camera there is mounted an omnidirectional camera, which is not used in this work.

(a) Camera 1

(b) Camera 3

(c) Camera 4

(d) Camera 5

FIGURE D.5: HERMES: Snapshot. The four views are synchronized. Camera 2 is a moving PTZ camera and out of the scope of this work.

D.2 HERMES Dataset

The HERMES dataset is a choreographed outdoor dataset recorded in Barcelona, Spain. The dataset is part of the project named Human Expressive Representations of Motion and their Evaluation in Sequences [HERMES, 2007]. HERMES is an EU-project which concentrates on how to extract descriptions of people behavior from videos in a restricted discourse domain, such as pedestrians crossing inner-city roads, approaching or waiting at stops of buses and humans in indoor worlds like an airport hall, a train station or a lobby. The HERMES dataset used in this thesis is a single sequence of one minute and 37 seconds. Five synchronized cameras are used to record the sequence, but one of the cameras is a moving PTZ camera and is out of the scope of this work. A snapshot of the four stationary views of the scene is shown in Figure D.5. The dataset is used to evaluate the system, but only camera 1 and 5 are used (see Section 9.2 on page 118).

As mentioned, the sequence is choreographed and several events happen. The most interesting event with regards to this thesis is the near collision of a group of people and a vehicle. Furthermore, a thief steals a bag and the thief is then chased by one of the persons.

Each view has a resolution of 1392×1040, but for testing and evaluating the system the view is resized to 696×520. The frame rate is 15 fps.

D.3 PETS 2001 Dataset

PETS is an abbreviation for Performance Evaluation of Tracking and Surveillance [Computational Vision Group, 2007]. The purpose of PETS is to automate the performance evaluation process by providing several datasets, metrics and results. PETS workshops have been held annually since 2000. The dataset from 2001 is for outdoor tracking of people and vehicles using two synchronized views recorded at Reading University, England. The PETS 2001 dataset contains five sequences where some include use of an omnidirectional camera and a moving camera. For this work, only a single sequence is extensively applied during development of the system, and a snapshot from this sequence is shown in Figure D.6.

FIGURE D.6: PETS 2001: Snapshot from the test sequence used during development. The two synchronized views are illustrated.

Each view is recorded with a resolution of 768×576 and with a frame rate of 25 fps. The length of the used sequence is 1 minute and 47 seconds.

Appendix E

List of Parameters

This appendix lists the parameters mentioned in the thesis. Parameters are listed for the foreground segmentation module, single view tracking module and the correspondence module.

E.1 Foreground Segmentation

In the following, the parameters for the Codebook method are listed. This is followed by the parameters used for the color segmentation shadow suppression based on [Cucchiara et al., 2001].

Codebook Parameters

ε_1 : Threshold that determines the radius of the codeword during training. It is the maximum allowed chromaticity difference.

ε_2 : Threshold that determines the radius of the codeword during online classification. It is the maximum allowed chromaticity difference.

α : Used along with β to determine the height of the codeword based on $\check{I}$ and $\hat{I}$.

β : Used along with α to determine the height of the codeword based on $\check{I}$ and $\hat{I}$.

N_{train} : The number of frames used to train the background model.

T_λ : The threshold for Maximum Negative Run-Length (λ) during temporal filtering. Codewords with λ values above this threshold are removed from the codebook.

γ : The learning rate used by the adaptive filter to update the activated codewords.

N_{stable} : The number of consecutive frames a pixel must be classified as background in order to be considered stable background. Only stable background is updated during online classification.

$N_{train,\ online}$: The number of frames that a codeword must undergo online training before it can be changed to a non-permanent codeword.

$T_{\lambda,\ online}$: The online MNRL threshold. The λ of the online training codeword must be below this threshold before it can be changed to a non-permanent codeword.

$N_{expiration}$: The maximum allowed number of frames between activation of a non-permanent codeword. If more frames occur without activation, the codeword is removed from the codebook.

n_{median} : The size of the median filter used in post-processing.

Color Segmentation Shadow Suppression Parameters

α_V : Lower limit for drop in intensity allowed for a potential shadow.

β_V : Upper limit for drop in intensity allowed for a potential shadow.

τ_S : Threshold for the saturation component. The saturation component for shadow should either remain constant or be lowered.

τ_H : Threshold for change in the hue component. A potential shadow should not cause a change in the hue component.

E.2 Single View Tracking

In the following, the parameters for object classification, the tracks, probabilistic appearance model and track smoothness are listed.

Object Classification Parameters

T_{noise} : Threshold for the the size of detected objects. Detected objects smaller than this threshold are removed as noise.

T_{spread} : Threshold used for the spread metric. Detected objects with a spread lower than this threshold are classified as vehicles. Detected objects with a spread above this threshold are classified as humans.

N_{recent} : The number of latest object classification the recent history histogram is based on. It is typically set to a value corresponding to above one second of frames.

Track Parameters

T_{dead} : The number of frames a track is allowed to not be updated before the track is destroyed.

T_{stable} : The number of frames a track must be updated before it is considered stable.

T_{sentry} : The number of pixels used for the sentry border. A detected object must be within this border in order to be created as a track.

$T_{entrySize}$: If a vehicle object is above this threshold in size it is created as a track regardless of the sentry border.

Probabilistic Appearance Model Parameters

α :	The learning rate for blending new color measurements in the color model.
λ :	The learning rate for blending new foreground pixel measurements in the probability mask.
N_{fit} :	Size of the neighborhood used to perform track refinement.
$T_{numPixels}$:	The number of assigned pixels needed to update the track when segmenting inter-object occlusions.
T_{depth} :	The number of disputed pixels needed in order to perform depth estimation between two tracks.

Track Smoothness Parameters

T_{corner} :	The allowed displacement of a track's bounding box corner. If a corner is displaced more than this threshold the track motion is deemed as not smooth, which could indicate that the track is being "stolen".
$T_{bboxArea}$:	The maximum allowed increase in the track's area for smooth track motion.
$T_{bboxBox}$:	The maximum allowed increase in the track's bounding box area for smooth track motion.

E.3 Correspondence of Objects

In the following, the parameters used for finding correspondence of humans and vehicles are listed.

Correspondence of Humans Parameters

D_T :	Initial constraint on the correspondence distance for valid pairs. The distance must be lower than this threshold for a valid pair.
D_{T1} :	Threshold used in the modified correspondence algorithm for pairing groups. The distance in the view with an unpaired track must be below this threshold.
D_{T2} :	Threshold used in the modified correspondence algorithm for pairing groups. The distance from the warped ground point location to the intersection point in the view with an already paired track must be below this threshold.

Correspondence of Vehicles Parameters

$T_{history}$:	The needed number of frames where two tracks must have been paired consecutively before they have a historic stable correspondence relationship.

Appendix F

Description of Developed Software

This appendix gives a description of the developed software. First the implementation is described. This is followed by an explanation of the configuration files, which is used in the developed software. Then the homography estimation tool is described. Finally, the tracking system is described.

F.1 Implementation

All code are implemented in C++ using the development environment Microsoft Visual Studio 2005. The code make use of the open source computer vision library OpenCV developed by Intel. OpenCV contains many basic image analysis and computer vision algorithms. The installation of OpenCV version 1.0 is located on the DVD (◎ */installation/opencv/*).

The implementation is made in an object oriented in accordance with the modules described in the thesis. The names of the modules correspond with the theory described in the thesis.

The test videos in the Matthews Lane dataset are recorded using the DivX codec version 6.5 and are divided into 10 minutes long video files. To speed up processing, the foreground segmentation videos are also available, and these videos are stored using the Alparysoft Lossless Video Codec version 2.0. This codec is applied because it does not add extra frames to the compressed video file. The output videos are compressed by the Xvid codec version 1.1.2. This codec is used because it gave the best visual result. Installation files for the codecs can be found on the DVD (◎ */installation/codecs/*).

MSXML 6.0 must also be installed in order to use the XML configuration files described next. Installation file for this program can be found on the DVD (◎ */installation/msxml6.0/*).

F.2 Configuration File

A configuration file is an XML-file and is needed in order to run any of the developed programs. The configuration file lists the input video to the program, and it is possible to indicate which frames in the input the program should process. It also determines if any output is to be stored and in case of video output, which codec should be used. Modules like the shadow suppression and single view tracking can be enabled or disabled. Visualization options are also controlled by the configuration file. View invariant analysis and event recognition as described in Chapter 9 on page 117 are in the terminology of the configuration file referred to as the system being in alarm mode. The alarm mode can be disabled or enabled. Furthermore, the

parameters used in the modules are also listed in the configuration file. The configuration file has a predefined structure of elements, and a detailed description of each element can be found on the DVD (◎ */application/config_ decription.doc*).

When calling the executable file for the homography estimation tool or the tracking system program, a configuration file must be specified as option. Without this option or with additional options, an error is reported by the program.

F.3 Use of the Homography Estimation Tool

The homography estimation tool makes it possible to both estimate the planar homography for a two view camera setup and test the estimated homography. The tool is controlled using a control panel, which is shown in Figure F.1(a). Given that the homography is loaded, the virtual view and the shared region can be shown by clicking on the top two buttons. The control panel also makes it possible to go forward or backward in the video. The program closes by pressing the Esc-key.

Given that the homography is estimated, it can be tested either by warping a point or a line between views. A point is warped by pressing the Alt-key and left clicking in one of the views. The warped point is shown as a blue dot in the other view. A line can be warped by right clicking twice in one of the views. The warped line is shown in the other view as a green line. An example of warping a person's principal axis is shown in Figure F.1(b) and F.1(c). Along with the virtual view and the shared region these techniques are used to visually test the accuracy of the estimated homography.

The homography can be estimated or re-estimated, which requires four control points in each view. By pressing the Ctrl-key and left clicking in one of the views a control point is stored. When the four control points are located in both views, the new homography is calculated and stored with names provided by the configuration file. If the homography is already estimated, the program explicitly asks the user if a new homography should be re-estimated before overwriting the existing homography.

The source code for the homography estimation tool is located on the DVD (◎ */applica-*

(a) Control panel (b) View 1 (c) View 2

FIGURE F.1: The three windows used by the homography estimation tool.

tion/homographyEstimationTool/). An executable version is also located on the DVD (◎ */application/homographyEstimationTool/release/*). In the same library some shortcuts are also stored. Each shortcut calls the executable file for the homography estimation tool with a different configuration file. A walkthrough for running the program can be found on the DVD (◎ */application/walkthrough.doc*).

F.4 Use of the Tracking System

The tracking system performs tracking of humans and vehicles. When calling the executable file a configuration file must be listed as an option. Besides the configuration file the tracking system might need the following files:

Homography files A homography file for each view is always needed. Each homography maps from an image view to the virtual view.

Virtual view image It is possible to enable or disable visualization of the virtual view. Given that it is enabled in the configuration file, a virtual view image must also be specified on which the detected objects are drawn. If the virtual view image is a virtual top-down view as in Figure 4.3 on page 33 and 4.4(c) on page 34 it can be used to verify the tracking result.

Alarm mode virtual view image If the system is in alarm mode, an alarm mode virtual view image must be available to draw the tracking result.

Alarm mode road map If the system is in alarm mode, a map indicating the area defined as the road is needed. This is simply a binary image, where a white pixel represents a road pixel.

The tracking system performs tracking using the parameters listed in the configuration file. From the configuration file it is possible to enable and disable the output from the system and also to specify the output directory. The tracking system is able to output the following:

<filename>_foreground.avi The binary foreground mask for the input video file named <filename>.

<filename>_result.avi The tracking result is superimposed on top of the raw input video file named <filename>.

virtualview.avi This video file contains the tracking result visualized on top of the raw input video, the corresponding foreground masks and the tracking result shown on the virtual view image. It is all stored in a single video file, which makes it easy to validate the tracking result.

view<X>_track<Y>.txt A text file is created for each track in each view. <X> indicate the view number and <Y> indicate the tracking number. For each frame the object has been tracked, the following information is saved:

- Object classification

FIGURE F.2: Snapshot of the tracking system showing the foreground segmentation, virtual view and tracking result superimposed on the input video.

- Ground point location
- Centroid location
- Bounding box location
- Area
- Boolean value indicating if the object is occluded. If this is the case, the depth estimation is also stored
- Whether or not the track is paired with a track in the other view. If this is the case, the paired track id is also stored

- View invariant point location
- View invariant size
- View invariant speed

When the program is not running in alarm mode only the foreground segmentation and input video are shown for each view. Depending on the settings in the configuration file, the input video might also show the tracking result. If visualization of the virtual view is enabled this is also shown. Figure F.2 illustrates the system when showing the virtual view and superimposing the tracking result on the input video.

If the program is running in alarm mode, there is only a single window showing the virtual view and two input views. A snapshot of this is shown in Figure F.3.

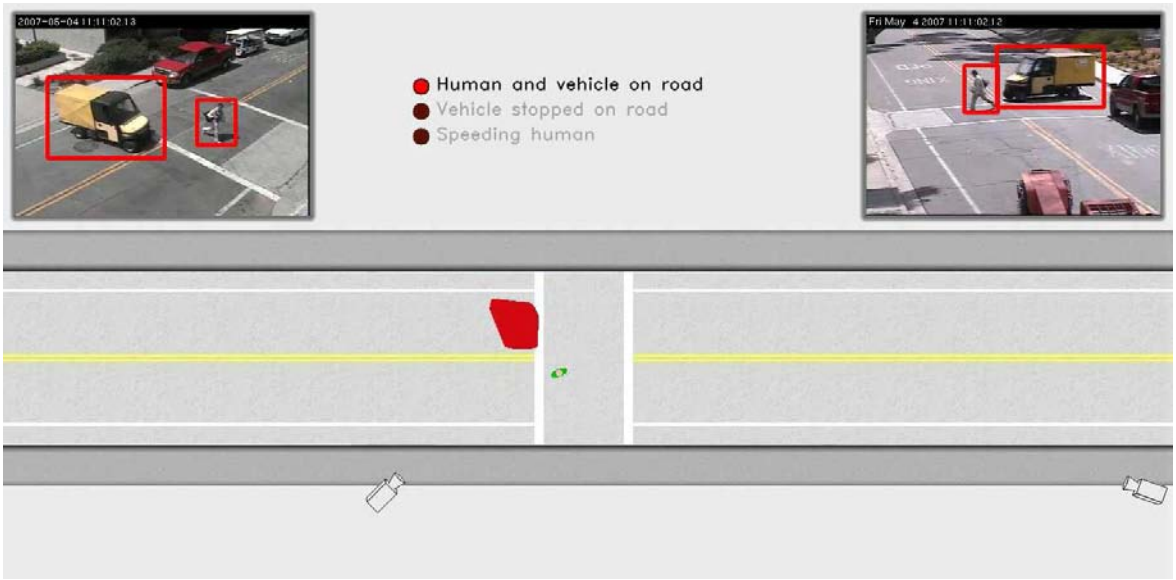


FIGURE F.3: Snapshot of the tracking system when running in alarm mode.

The program can be paused by pressing the S-key. It is possible to exit the program by pressing the Esc-key.

The source code for the tracking system is located on the DVD (◎ */application/humanVehicleTracking/*). An executable version is also located on the DVD (◎ */application/humanVehicleTracking/release/*). In the same library some shortcuts are also stored. Each shortcut calls the executable file for the tracking system with a different configuration file. A walkthrough for running the program can be found on the DVD (◎ */application/walkthrough.doc*).

Appendix G

Publication

This appendix contains the publication "Automatic Annotation of Humans in Surveillance Video", which was submitted and accepted for oral presentation at The First International Workshop on Video Processing and Recognition (VideoRec'07). The publication was made during the first two months of the stay at Computer Vision and Robotics Research laboratory at University of California, San Diego.

Automatic Annotation of Humans in Surveillance Video

D.M. Hansen, B.K. Mortensen, P.T. Duizer, J.R. Andersen and T.B. Moeslund
Laboratory of Computer Vision and Media Technology
Aalborg University, Denmark

Abstract

In this paper we present a system for automatic annotation of humans passing a surveillance camera. Each human has 4 associated annotations: the primary color of the clothing, the height, and focus of attention. The annotation occurs after robust background subtraction based on a Codebook representation. The primary colors of the clothing are estimated by grouping similar pixels according to a body model. The height is estimated based on a 3D mapping using the head and feet. Lastly, the focus of attention is defined as the overall direction of the head, which is estimated using changes in intensity at four different positions. Results show successful detection and hence successful annotation for most test sequences.

1. Introduction

Due to the many potential applications "Automatic Surveillance System" research has received much attention recently [9]. It is generally believed that a successful surveillance system will contain a figure-ground segmentation to detect humans in the images, tracking to maintain temporal coherence, and finally a recognition part to recognize identity, actions etc. [9]. So far no system has been able to successfully do any of the three parts in a robust manner and hence no commercial systems are available. While waiting for robust subsystems to build upon, different successful sub-applications have been built.

In our work we follow this trend and aim at an automatic annotation system for surveillance video. The long term goal is a network of connected cameras, each annotating every person passing by in terms of appearance, size, and behavior/action. In this particular paper we use one camera and limit the annotation to a few informative measures.

Different types of appearance measures exist such as color and style of hair, shirt, pants, shoes, beard, and glasses. In this work we focus on the most fundamental, namely the primary color of the hair, upper body clothing and lower body clothing. Regarding the size of a person many measures exist, but we focus on the primary one, namely the height of the person. Annotating the behavior/action of a person is a complex task. In this work we limit behavior to the attention of a person defined as the direction of the head. We are not aiming at actual head pose but rather the general direction of attention.

The paper is structured as follows. In Section 2 it is described how humans are segmented and tracked. In Sections 3, 4, and 5 we describe the actual annotation of the three measures: appearance, size, and behavior, respectively. In Section 6 results are presented and finally the paper is concluded in Section 7.

2. Segmentation of Humans

Before any annotation can commence, each human has to be segmented from the rest of the image, i.e., figure-ground segmentation. As this is the first step in many systems analyzing humans several approaches exist based on e.g., background subtraction [4, 15], motion [14, 16], appearance [5, 13], and shape [7, 17]. For an overview see [9].

We apply a robust version of background subtraction, known as the Codebook method [6], because it has been shown to operate for ten hours without losing significant selectivity [3]. The method contains three steps: modeling the background, pixel classification and model updating.

Each pixel is modeled as a group of codewords which correspond to the codebook for this particular pixel. Each codeword is a cylinder in RGB-space. In each new frame each pixel is compared to its codebook. If the current pixel value belongs to one of the codewords it is classified as background, otherwise foreground.

The codebooks are built during training and updated at run-time. The training phase is similar to the pixel classification mentioned above except that a foreground pixel results in the construction of a new codeword and a background pixel is used to modify the codeword it belongs to using a standard weighting scheme. The codebooks generated in this way during training will typically fall into three categories:

Fourth Canadian Conference on Computer and Robot Vision(CRV'07)
0-7695-2786-8/07 $20.00 © 2007 IEEE

Static codebook For example a pixel representing a road with no shadows or occlusions. Typically only one codeword is used.

Quasi-static codebook For example a pixel containing the sky, but sometimes occluded by vegetation due to wind gusts. During training typical two codewords will be constructed for this codebook, one for the sky and one for the vegetation.

Noisy codebook One of the above combined with "noise" in the form of a pedestrian, car etc. passing by the pixel or noise due to incorrect segmentation. The result will be an often high number of codewords for this codebook.

To handle the noisy codebooks a temporal filter is applied. It is based on the Max Negative Run-Length (MNRL), which is the longest time interval in which a codeword has not been activated. The filter effectively removes codewords with little support during the training phase, such as passing pedestrians.

During run-time the activated codewords are updated in two ways. Firstly, as described above, using a standard weighting scheme to ensure updates with respect to slow changes in the scene, e.g., the position of the sun. The second type of update handles fast changes in the scene. Imagine a car enters the scene and is parked inside the scene. Obviously the car will be segmented as a foreground object. If the car stays in the scene it is considered background and new foreground objects passing the car can be correctly classified as foreground objects. This is done by defining a new codeword whenever a pixel is classified as foreground. If this new codeword has support in terms of a small MNRL then it is defined as a codeword and added to the codebook for this pixel. These types of codewords are denoted *non-permanent codewords* and can be removed again if they lack support for some time. A codeword learned during training is denoted a *permanent codeword* and can not be removed. Using the two updates provides a robust figure-ground segmentation for further processing.

2.1. Tracking

After having performed figure-ground segmentation we need to filter the output first spatially to ensure that each blob corresponds to one and only one human and second temporally to obtain a track of each human over time.

We assume walking or standing humans and can therefore define an interval of acceptance for the ratio between the height and width of the bounding box. Furthermore, after improving the output using standard filter methods we introduce size criteria[1], see Figure 1, together with a proximity criterion which merge correct sized blobs located on top of each other [1].

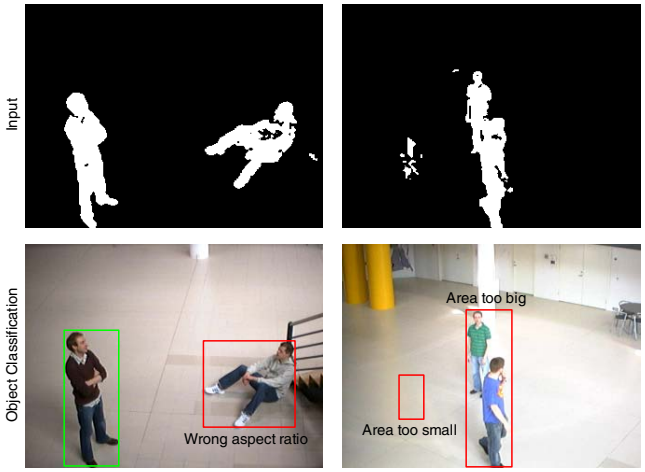


Figure 1. An illustration of the size criteria.

Each time a new track is initiated an ID is assigned. The tracking of IDs is done using temporal filtering with a zero'th order predictor and an Euclidean distance measure. Groups and partial occlusion will result in blobs that are too large or small respectively, which are ignored. This can result in small tracks or tracks not assigned an ID in every frame. To decide whether we want to keep a track or not for further processing we use the same principle as used for the MNRL [1].

3. Appearance: Color

Representing a human by colors has been used in tracking for some time. The standard approach is to divide the silhouette into a number of regions (usually three) and represent each region by either a Mixture of Gaussian or a histogram, see e.g., [10, 13] and [9] for an overview. These representations tend to describe the average color in a region. Furthermore, they often use fixed region borders resulting in a long shirt contributing to the colors of both the upper and the lower body parts.

Neither the averaging nor the fixed region borders constitute a problem in tracking. However, for annotation a different method is required, which allows for a detection of the primary color for both the upper and the lower body parts, respectively. Before explaining how this is done we first describe the chosen color space representation.

[1]These criteria are defined with respect to the camera's position (calibration), the layout of the scene, and the position of the human in the scene.

3.1. Color Space Representation

We use HSV colors since these are more practical for human interpretation[2] in an annotation system and at the same time decouples color and intensity allowing for less lighting dependent annotations.

Even though humans can distinguish between thousands of color shades, they are normally only able to remember 11 basic colors: red, green, yellow, orange, brown, pink, purple, white, gray, and black [2].

In order to convert from HSV color coordinates to these 11 color terms, the HSV-color space has to be subdivided. The hue-saturation color space is divided as shown in Figure 2 [1].

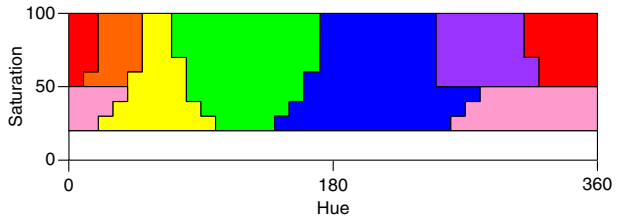


Figure 2. The hue-saturation space is divided into eight fields [1].

The hue-saturation space is divided into eight different fields. When the saturation is low, which is the white field in the figure, it is hard to separate different colors. Instead the color term is either black, gray or white, depending on the brightness of the color. The seven other fields represent the rest of the color terms. The brown color is missing in the figure, because it shares the hue and saturation with other fields. When the brightness is low, both orange, yellow and pink look brown. This is used to define the brown color.

3.2. Assigning Pixels to Body Parts

Taking the silhouette as input we are interested in first finding the color of the pixels belonging to the different body parts, see Figure 3(b), and then constructing a textual representation.

The input color image is segmented using the k-means clustering algorithm. The number of samples, n, is equivalent to the number of pixels in the image. The feature space is 3 dimensional, corresponding to the dimension of the color space. Experiments in this work show that better segmentation is achieved using all color components instead of only hue and saturation. The samples are divided into N_c clusters (20 in this work) using the k-means algorithm with an euclidian distance measure.

[2]For enquires into a database of annotations.

To compensate for oversegmentation, similar clusters are merged together. Two clusters are merged, if the euclidian distance in the hue-saturation space between their centers is below a predefined threshold.

A cluster can consist of a high number of blobs, which are not necessarily connected. We therefore apply a connected component analysis based on contours [1] to find all the individual blobs. The resulting blobs can be seen in Figure 3(c). For all blobs, the size and the mean color are found. Too small blobs are ignored as are blobs having a color similar to skin. We use a look-up-table in hue-saturation space to detect skin pixels [1].

The different blobs are assigned to one of the body parts based on the assumption that people are upright. We use the vertical position of their centers to assign blobs [12]: Head $\in [0, 16\%]$. Upper body $\in [16, 45\%]$. Lower body $\in [45, 100\%]$. 0% = top of bounding box and 100% = bottom of bounding box. See Figure 3.

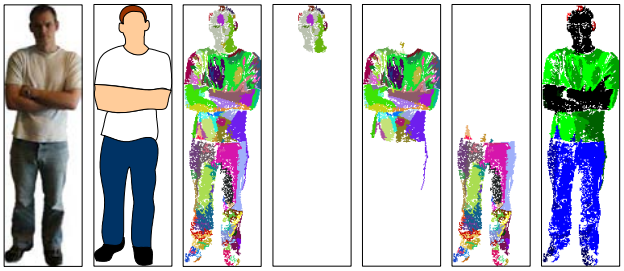


Figure 3. Assignment of blobs to body parts. (a) Input. (b) An ideal color segmentation of a person. (c) Blobs. (d) Head. (e) Upper body. (f) Lower body. (g) Merged blobs. Black represent skin, red the hair, green the shirt and blue the pants.

The body parts still consist of a large number of blobs. To lower this number blobs are merged in the same manner as for clusters. Only this time, the color threshold is set higher to merge more blobs. Different thresholds can be used in the three body parts. Normally, people wear uniform colored pants, but might wear a multicolored shirt. Therefore, the threshold should be lower for the upper body than the lower body.

The merged blobs are illustrated in Figure 3(g). In the figure the black color represent skin blobs, which have not been merged. The two red shades are hair, the three green shades are the shirt and the three blue shades are the pants. The largest of the merged blobs is used as output in the textual description, since only the most dominant color is desired. In order to obtain stable color estimates we filter the output for each track by selecting the most often occurring color along a track.

4. Size: Height

To estimate the height we assume walking or standing persons. A prerequisite of the method is a calibration between the camera and floor, yielding the projection matrix shown in Equation 1. A is a matrix containing the intrinsic parameters, R and $\vec{t}$ are the extrinsic parameters. The extrinsic parameters are chosen so the X and Y world axis are in the ground plane, which is assumed to be a two dimensional plane.

$$H = A\left[R\vec{t}\right] = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \end{bmatrix} \quad (1)$$

Two image points must be located to calculate the height. One point located on the ground between the person's feet called the ground point and a point at the top of the head. The ground point is estimated using convex hull points. In order to find the ground point, two foot points are located as the two convex hull points closest to the two lower corners of the bounding box, see Figure 4. The ground point is the intersection between a vertical line through the median point of the body and a line defined by the two foot points, see Figure 4. The head point is chosen as the topmost pixel in the body silhouette.

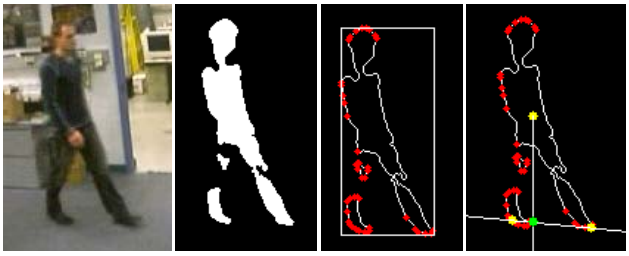


Figure 4. The principle used to find the top of the head and ground point.

After locating the ground point and head point, the next step is to map these points into world coordinates. The ground point is assumed to be in the ground plane, hence $Z = 0$ in world coordinates. As described in [8], H can be altered to represent a mapping from the image plane to the ground plane (where $Z = 0$) of the world coordinate system. This makes it possible to compute the person's ground plane position in world coordinates. Knowing the ground plane position, it is now possible to extend a line in the direction of the ground plane normal until it intersects a plane extended by the head point. This is illustrated in Figure 5. The height Z is calculated by Equation 2, where u and v are the head point coordinates and X and Y are the world coordinates for the ground point.

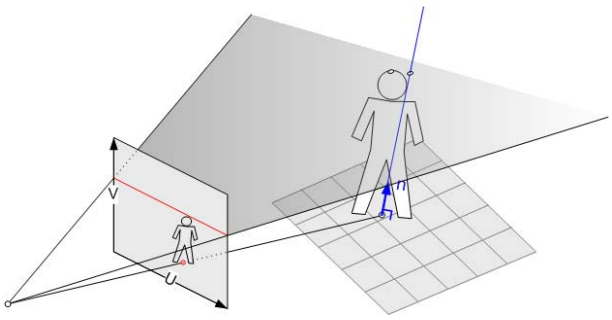


Figure 5. An illustration of the height estimate using the ground point and head point. The height is given by the Z world coordinate of the intersection point between the normal and the plane extended by the head point.

$$Z = \frac{(h_{21} - vh_{31})X + (h_{22} - vh_{32})Y + h_{24} - vh_{34}}{vh_{33} - h_{23}} \quad (2)$$

The height estimate is stored with the tracking ID for each frame the person is tracked. When a track is ended, e.g., by the person leaving the scene, we have several estimates of the person's height. However, only one height estimate is desired and this is chosen as the median value of all the height estimates. This makes the estimate robust towards outliers, which may be caused by poor segmentation in a few frames of the total tracking period. Furthermore, the method is based only on a couple of matrix operations and a convex hull method making it computationally inexpensive.

5. Behavior: Head Direction

The orientation of the person's head is used as an estimate for the person's attention. We first extract the head in an image and then classify the head direction into one of five different categories.

5.1. Head Extraction

Before the orientation of the head can be determined, template matching is first applied to extract the head.

The head is extracted from the silhouette of the person. The binary image allows for fast matching with exclusive-or operations. A number of preprocessing steps trim the size of the silhouette to further reduce processing time.

0-7695-2786-8/07 $20.00

The template consists of a head and torso whose size is changed dynamically for scale invariance. The head and torso is dimensioned based on typical human measures, and the size is determined by an estimate of the person's shoulder width. The shoulder width is estimated as 90 percent of the minimum distance from the median of the silhouette to either side of the person's bounding box. This approach provide a reliable measure invariant to typical segmentation errors. Figure 6 shows the steps in the head extraction process.

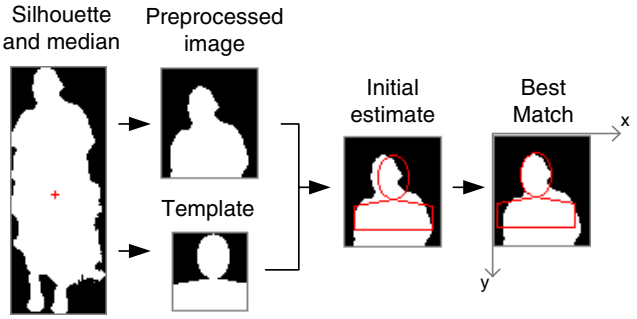


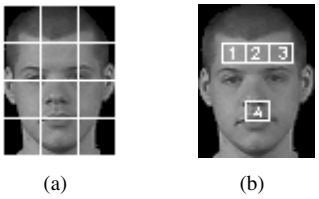
Figure 6. The steps in the head extraction process.

5.2. Head Direction Classification

The head is classified into one of five classes, corresponding to an orientation of Right 90° , Right 45° , 0° , Left 45° and Left 90° , where 0° is a person facing the camera. For determining the head orientation at low resolution, a view-based method inspired by [11] is used. The method in based on calculating moments of pixel intensity distribution in subimages of the extracted head. The moment features provide invariance to scaling and robustness toward imperfect segmentation and head extraction. In [11], the head is divided into twelve subimage in a 3×4 grid, as depicted in Figure 7(a), and for each subimage a feature value W is calculated using Equation 3.

$$W = \frac{mean(subimage) - mean(image)}{std(image)} \qquad (3)$$

Only those pixels that are within the silhouette of the person are part of the calculations. Furthermore, any highlights in the head region are removed beforehand, to reduce the influence of face items like eyeglasses or shiny skin. Inspection of these twelve features in the training data showed some features provide very little or no information. The corners had a poorly separated mean value and a high variance. Furthermore, many of the features are highly correlated. To improve this, 40 subimages in a 5×8 grid were used to



(a) (b)

Figure 7. (a) head region divided into twelve subimages. (b) four selected subimages of a 5×8 grid.

locate more specific areas in the head, and the four features depicted in Figure 7(b) were selected. The numbers in the subimages corresponds to the numbering of the features, and the plot in Figure 8 depicts their mean value and variance. As seen in the figure, these features are able to separate the classes in the training data, but does also raise the requirements for the training data to be representative. The different grid division gives areas in the head where e.g. the area is skin-only when the head is oriented 0° and hair-only when the head is oriented ±90° .

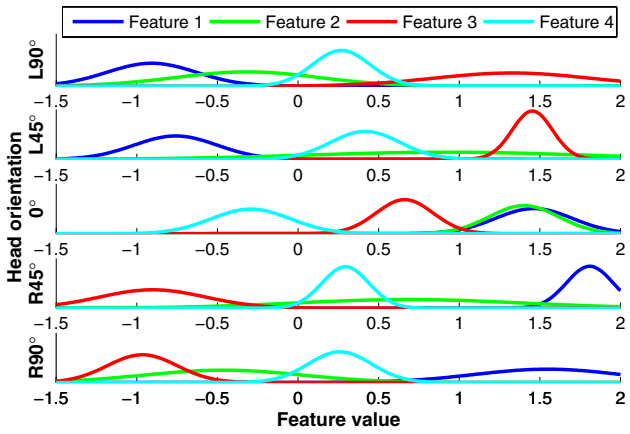


Figure 8. The mean and variance of the four features, calculated from the four selected subimages.

The four-dimensional feature vector is classified using k-nearest-neighbor using Euclidean distance. A temporal filter is applied to each track selecting the most occurring head orientation class within the last X frames as the output. Experiments have shown that using $X = 15$ frames gives the best output and seems to be a reasonable size to filter out sudden head movement. 15 frames correspond to one second meaning that attention is defined as a constant head direction for one second.

6. Results

In this section we present test results for the previous sections. All results are recorded with a standard web-camera with a resolution of 320x240 pixels and with a frame rate of 15 Hz.

6.1 Segmentation of Humans

111 frames are selected randomly from five different video sequences. 91 frames contained one of more humans. The pixels belonging to a human are found manually to obtain ground truth. On average the false acceptance rate[3] is 0.06% and the false rejection rate[4] is 5.95%. The errors originate especially from foreground camouflage and shadows. When the appearance of a person's clothing is similar to the background then a "hole" can appear in the silhouette leading to false rejections. This, however, seldom affects the rest of the system. Shadows can affect both the estimation of the colors as well as the height estimation. In most cases, however, the temporal filtering solves the problems[5].

The tracking works very well because occlusions are ignored. If the system is to be used to annotate individuals during occlusions, then better segmentation and tracking is required. Possible solutions can be found in [9].

6.2 Appearance: Color

The color descriptor is tested in two test cases. The first case tests the correctness of the annotated color term and the second case tests the variation of the extracted color values for a person walking through the scene.

The purpose of the first test case is to analyze the ability to annotate different colors. Furthermore, the transformation from HSV to the 11 color terms is tested. 33 input images are manually segmented, so only the person and a white background is present. Four people agreed on the color terms yielding ground truth. See examples in Figure 9. The first row is the ground truth for the upper body, while the second row is the annotated color terms.

All shirts are annotated correctly, except for the turquoise-green t-shirt (#6), which is annotated as blue. The reason for wrong assignment is that the HSV value lies in between the green and blue color term, where the border is sharply defined, despite it being rather fuzzy in reality. This problem also exist when humans determine a color and a suggestion is therefore to represent borderline colors by two different colors.

This test also highlights difficulties when a person is wearing shirt and pants of the same color. For test subject 2, it is difficult to find the border between the upper and lower body. The clusters often separates the upper and lower body, but after the merging of clusters, the upper and lower body can be fused. The annotated color terms are however often correct, because the largest color blobs of both the upper and lower body are still correct - blue in this particular case.

Lastly it was found that problems occur when people are wearing skin-colored clothes. In these situations, the produced color term might only represent the second most dominant color. This is the case with the pants of test subject 10. The pants are classified as skin making the white shoes the largest part of the lower body. The pants are therefore labeled as white. To handle such situations more advanced analysis of the blobs is required.

The second test case tests the color descriptor when connected to the tracking module. Instead of using manually segmented images, the output produced by the tracking module is used directly. The same test recordings as in the first test case are used.

In general, for all test subjects the same issues are observed. Outliers occur mainly because color descriptors are made even though the person is not completely in the scene. Also, misclassifications from the figure-ground segmenting caused by strong shadows lead to false color terms. However, these problems are eliminated (except for the hair where the resolution is too low to produce trustworthy results) by temporal filtering and the correct color terms are found in all cases except a few where blue and green are mixed up as illustrated in Figure 9.

6.3 Size: Height

We recorded a video with four people moving in different directions and at different speeds within the scene to test the height estimate. The results are shown in Table 1. A total of 17 tracks or height estimates are performed and the number of frames available for the height estimation is between 33 and 90 (2-6 seconds). The table shows the statistics of each person. Despite the segmentation issues and the merging of two tracks, the poorest height estimate is only 3.4 cm from the actual height.

6.4 Behavior: Head Direction

Two types of tests are conducted, a quantitative and a qualitative test.

In the quantitative test eight test subjects are instructed to put their heads in one of the five head positions for a certain period of time by standing still and looking at certain

[3] The false acceptance rate is defined as the number of background pixels falsely accepted as foreground pixels, divided by the total number of manually annotated background pixels.

[4] The false rejection rate is defined as the number of foreground pixels falsely rejected, divided by the total number of manually annotated foreground pixels.

[5] For more results on the figure-ground segmentation please refer to [3].

Figure 9. 9 test images. The first row of color terms is ground truth. The second row is the output from the system.

	Person 1	Person 2	Person 3	Person 4
# tracks	4	7	2	4
Height	182.0 cm	192.0 cm	170.0 cm	186.0 cm
Mean	181.4 cm	191.5 cm	171.9 cm	186.0 cm
Std.dev.	1.9 cm	1.9 cm	2.2 cm	0.5 cm
Min	179.7 cm	189.4 cm	170.3 cm	185.7 cm
Max	184.0 cm	193.7 cm	173.4 cm	186.7 cm

Table 1. The result of estimating the height for four people. # tracks lists the number of tracks for each person in the recording. Height is the actual height of the person.

Sub.	R90°	R45°	0°	L45°	L90°
1	100%	4.9%	75.8%	0%	100%
2	48.2%	100%	100%	90.1%	100%
3	100%	100%	100%	0%	100%
4	95.8%	45.6%	86.3%	75.3%	100%
5	20.8%	88.3%	100%	100%	100%
6	0%	97.6%	100%	100%	0%
7	100%	86.3%	100%	100%	100%
8	100%	88.1%	100%	100%	100%

Total correct classifications:	80.1%
Classifications in adjacent class(es):	19.9%
Classifications in other classes:	0.0%

Table 2. The result of estimating head direction for the test subjects.

markers mounted in the room. Two samples per class per subject is used to train a classifier which is tested on more than 2000 frames.

The results are shown in Table 2. The table shows the percentage of correctly classified frames for a specific class and test subject along with the total correct classifications. The system yields a classification rate of 80.1%. Of the 19.9% misclassifications all were classified within the adjacent class(es), which is $\pm 45^\circ$ of the correct class. If the five classes are merged into three classes (Right, Front and Left) the result is a 98.5% correct classification rate. The results vary between the test subjects. The poorest classification is for test subject 1 who has hair that sticks up. The hair moves the subimages to an undesired area causing more misclassifications. Test subject 5 and 6 are the same person without glasses and with glasses, respectively. With glasses it is harder to classify Left 90° correctly and it is misclassified as Left 45° . In the qualitative test 40 video sequences (7000+ frames) are used. In each sequence a test person walks towards the camera and changes his/her position in a supervised manner. When the subject is within 9 meters enough head pixels ($> 10x15$) are available for classification. The results depend on the distance to the camera. When the subjects are more than 4.5 meters away 49.0% is correctly classified. 75.4% was correctly classified for subjects closer than 4.5 meters. The total correct classification for the entire walking distance is 62.2%. By merging the five classes into three classes (Right, Front and Left) the far distance, close distance and total correct classification becomes 72.5%, 92.4% and 82.5%, respectively.

In real-life applications a frame-by-frame head direction is not always desirable and annotation of general tendencies in the form of a textual output rather than frame-by-frame output is more feasible, e.g., registration of how many customers are looking at a certain display window. In such applications, a coarse time location of a change in head position is sufficient. The filtered output from the head direction classification have stable periods where the same class is outputted for several frames. If we define a stable period

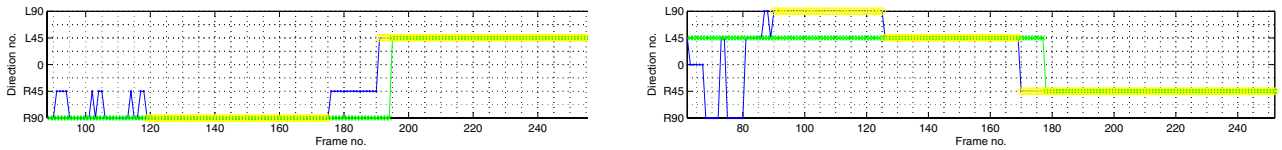


Figure 10. Head direction. Blue: Estimate. Green: Ground truth. Yellow: Stable period.

to be at least one second we can create a textual output. For the left graph in Figure 10, the textual output is {*Frame 119-175 looking Right 90°* , *Frame 191-270 looking Left 45°* } using the stable periods and is depicted graphically in the figure by yellow lines[6]. Besides the inaccurate location of the transition in head direction, the textual output is correct.

A problematic issue is shown in Figure 10 (right). In this case, an extra head direction is listed in the textual output. However, it is typically a head direction within 45° of the correct direction and occurs when the subject is located farthest away from the camera or during head direction transition. Hence in the context of for example determining if a customer is looking at the stores on his left or right, the proposed head direction provides a reliable result.

7. Conclusions

The paper has presented annotation of persons passing a surveillance camera in terms of primary color of upper and lower body part, the height of the person, and head direction to estimate attention. The annotations are incorporated into a system based on a robust segmentation of individuals.

One limitation of the system is its lack of occlusion handling. For dense areas this will reduce the number of persons that are actual annotated and a more advanced segmentation is required to handle such situation, see [9] for examples. In future work an active camera will be controlled by the segmentation to zoom in on the face for obtaining a quality picture to store along with the other annotations or for further processing, e.g., ID or facial expression recognition.

References

[1] J. Andersen, P. Duizer, D. Hansen, and B. Mortensen. Automatic Annotation of Humans in Surveillance Video Recordings. Technical report, Computer Vision and Media Technology, Aalborg University, Denmark, 2006.

[2] B. Berlin and P. Kay. *Basic Color Terms: Their Universality and Evolution*. University of California Press, Berkeley, CA, 1969.

[3] P. Fihl, R. Corlin, S. Park, T. Moeslund, and M. Trivedi. Tracking of Individuals in Very Long Video Sequences. In *Int. Symposium on Visual Computing*, Lake Tahoe, Nevada, USA, November 6-8 2006.

[4] M. Heikkila and M. Pietikainen. A Texture-Based Method for Modeling the Background and Detecting Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[5] M. Hu, W. Hu, and T. Tan. Tracking People through Occlusion. In *International Conference on Pattern Recognition*, Cambridge, UK, Aug 2004.

[6] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Real-time Foreground-Background Segmentation using Codebook Model. *Real-time Imaging*, 11(3):167–256, 2005.

[7] V. Krüger, J. Anderson, and T. Prehn. Probalilistic Model-Based Background Subtraction. In *Scandinavian Conference on Image Analysis*, Joensuu, Finland, Jun 19-22 2005.

[8] C. Madden and M. Piccardi. Height Measurement as a Session-Based Biometric for People Matching Across Disjoint Camera Views. In *Image and Vision Computing New Zealand*, Dunedin, New Zealand, 28 - 29 Nov 2005.

[9] T. Moeslund, A. Hilton, and V. Krüger. A Survey of Advances in Vision-Based Human Motion Capture and Analysis. *Journal of Computer Vision and Image Understanding*, 104(2-3), 2006.

[10] K. Okuma, A. Taleghani, N. Freitas, J. Little, and D. G. Lowe. A Boosted Particle Filter: Multitarget Detection and Tracking. In *European Conference on Computer Vision*, Prague, Czech Republic, May 11-14 2004.

[11] S. Park and J. Aggarwal. Head Segmentation and Head Orientation in 3D Space for Pose Estimation of Multiple People. In *IEEE proc. Southwest Symposium on Image Analysis and Interpretation (SSIAI 2000)*, Austin, TX, USA, 2000.

[12] S. Park and J. Aggarwal. Simultaneous tracking of multiple body parts of interacting persons. *Computer Vision and Image Understanding*, 102(1), 2006.

[13] D. Roth, P. Doubek, and L. Gool. Bayesian Pixel Classification for Human Tracking. In *IEEE Workshop on Motion and Video Computing (MOTION'05)*, Breckenridge, Colorado, Jan 2005.

[14] H. Sidenbladh. Detecting Human Motion with Support Vector Machines. In *International Conference on Pattern Recognition*, Cambridge, UK, Aug 2004.

[15] C. Stauffer and W. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. In *Computer Vision and Pattern Recognition*, Santa Barbara, CA, USA, June 1998.

[16] P. Viola, M. Jones, and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. *International Journal of Computer Vision*, 63(2), 2005.

[17] B. Wu and R. Nevatia. Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detection. In *International Conference on Computer Vision*, Beijing, China, Oct 15-21 2005.

[6] These frame indexes can be converted to 3D view direction using the camera calibration.

Bibliography

[Aalborg University, 2007] Aalborg University (2007). Pay As You Speed. `http://www.sparpaafarten.dk/en/index.php`.

[Agarwal et al., 2005] Agarwal, A., Jawahar, C., and Narayanan, P. (2005). A Survey of Planar Homography Estimation Techniques. Technical report.

[Andersen et al., 2006] Andersen, J., Duizer, P., Hansen, D., and Mortensen, B. (2006). Automatic Annotation of Humans in Surveillance Video Recordings. Technical report, Computer Vision and Media Technology, Aalborg University.

[Andersen and Corlin, 2005] Andersen, P. F. and Corlin, R. (2005). Tracking of Interacting People and Their Body Parts for Outdoor Surveillance. Master's thesis, Computer Vision and Media Technology, Aalborg University.

[Barreto, 2001] Barreto, H. (2001). An Introduction to Least Median of Squares.

[Blauensteiner et al., 2006] Blauensteiner, P., Wildenauer, H., Hanbury, A., and Kampel, M. (2006). Motion and Shadow Detection with an Improved Colour Model. In *Proc. of the IEEE Int. Conf. on Signal and Image Processing (ICSIP06)*, Hubli, India.

[Calderara et al., 2005] Calderara, S., Vezzani, R., Prati, A., and Cucchiara, R. (2005). Entry Edge of Field of View for Multi-camera Tracking in Distributed Video Surveillance. *IEEE Conference on Advanced Video and Signal Based Surveillance. (AVSS2005)*, pages 93– 98.

[Chalidabhongse et al., 2003] Chalidabhongse, T. H., Kim, K., Harwood, D., and Davis, L. (2003). A Perturbation Method for Evaluating Background Subtraction Algorithms. *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance.*

[Chang and Gong, 2001] Chang, T.-H. and Gong, S. (2001). Tracking Multiple People with a Multi-Camera System. *Proceedings of the IEEE Workshop on Multi-Object Tracking.*

[Christensen and Nikolajsen, 2005] Christensen, M. F. and Nikolajsen, J. (2005). Tracking of People in Airports. Technical report, Computer Vision and Media Technology, Aalborg University.

[Collins et al., 2000] Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., and Wixson, L. (2000). A System for Video Surveillance and Monitoring.

[Computational Vision Group, 2007] Computational Vision Group (2007). PETS: Performance Evaluation of Tracking and Surveillance. `http://www.cvg.rdg.ac.uk/slides/pets.html`.

[Criminisi et al., 1999] Criminisi, A., Reid, I., and Zisserman, A. (1999). A Plane Measuring Device. *Image and Vision Computing 17*, pages 625–634.

[Cucchiara et al., 2003] Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2003). Detecting Moving Objects, Ghosts and Shadows in Video Streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342.

[Cucchiara et al., 2001] Cucchiara, R., Grana, C., Piccardi, M., Prati, A., and Sirotti, S. (2001). Improving Shadow Suppression in Moving Object Detection with HSV Color Information. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE.*

[Cucchiara et al., 2004] Cucchiara, R., Grana, C., and Tardini, G. (2004). Track-based and Object-based Occlusion for People Tracking Refinement in Indoor Surveillance. *ACM 2nd International Workshop on Video Surveillance and Sensor Networks*, pages 388–393.

[Cutler and Davis, 2000] Cutler, R. and Davis, L. S. (2000). Robust Real-Time Periodic Motion Detection, Analysis, and Applications. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 22(8):781–796.

[Dockstader and Tekalp, 2001] Dockstader, S. and Tekalp, A. (2001). Multiple Camera Tracking of Interacting and Occluded Human Motion. *Proceedings of the IEEE*, 89(10):1441–1455.

[Doshi and Trivedi, 2006] Doshi, A. and Trivedi, M. (2006). Hybrid Cone-Cylinder Codebook Model for Foreground Detection with Shadow and Highlight Suppression. *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance (AVSS'06).*

[Elgammal et al., 2000] Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric Model for Background Subtraction. *6th European Conference on Computer Vision, Dublin, Ireland.*

[Endsley, 1988] Endsley, M. R. (1988). Situation awareness global assessment technique (SAGAT). *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference: NAECON 1988*, pages 789–795.

[Færdselssikkerhedskommisionen, 2000] Færdselssikkerhedskommisionen (2000). Hver ulykke er en for meget - Nye mål 2001-2012. `http://www.trm.dk/graphics/synkron-library/trafikministeriet/publikationer/pdf/039.pdf`.

[Færdselssikkerhedskommisionen, 2007] Færdselssikkerhedskommisionen (2007). Hver ulykke er en for meget - Nye mål 2001-2012 - Forslag til revision af strategier og indsatser. `http://www.faerdselssikkerhedskommissionen.dk/wimpshow.asp?type=image&id=77277`.

[Hansen et al., 2007] Hansen, D. M., Mortensen, B. K., Duizer, P. T., Andersen, J. R., and Moeslund, T. B. (2007). Automatic Annotation of Humans in Surveillance Video. *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, 00:473–480.

[Haritaoglu et al., 2000] Haritaoglu, I., Harwood, D., and Davis, L. S. (2000). W4: Real-Time Surveillance of People and Their Activities. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 22(8):809–830.

[Hartley and Zisserman, 2004] Hartley, R. and Zisserman, A. (2004). Cambridge University Press, second edition.

[HERMES, 2007] HERMES (2007). Human Expressive Representations of Motion and their Evaluation in Sequences. `http://www.hermes-project.eu/`.

[Hillerød Kommune, 2004] Hillerød Kommune (2004). Virkemiddelkatalog (Danish). `http://www.hillerod.dk/upload/teknik/planlaegning/byplan/pdf/trafik/virkemiddelkatalog_001.pdf`.

[Hu et al., 2004a] Hu, M., Hu, W., and Tan, T. (2004a). Tracking People through Occlusions. *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*, 2:724–727.

[Hu et al., 2006] Hu, W., Hu, M., Zhou, X., Tan, T., Lou, J., and Maybank, S. (2006). Principal Axis-Based Correspondence between Multiple Cameras for People Tracking. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 28(4):663–671.

[Hu et al., 2004b] Hu, W., Tan, T., Wang, L., and Maybank, S. (2004b). A Survey on Visual Surveillance of Object Motion and Behaviors. *IEEE Transactions On Systems, Man, and Cybernetics*.

[Jain and Jawahar, 2006] Jain, P. K. and Jawahar, C. V. (2006). Homography Estimation from Planar Contours. *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 877–884.

[Javed and Shah, 2002] Javed, O. and Shah, M. (2002). Tracking And Object Classification For Automated Surveillance. *Proceedings of the 7th European Conference on Computer Vision-Part IV*.

[Jeong and Jaynes, 2005] Jeong, K. and Jaynes, C. (2005). Moving Shadow Detection using a Combined Geometric and Color Classification. *IEEE Workshop on Motion and Video Computing. (WACV/MOTIONS2005)*, 2.

[Kang et al., 2003] Kang, J., Cohen, I., and Medioni, G. (2003). Continuous Tracking Within and Across Camera Streams. *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2003)*.

[Kastrinaki et al., 2003] Kastrinaki, V., Zervakis, M., and Kalaitzakis, K. (2003). A Survey of Video Processing Techniques for Traffic Applications. *Image Vision Computing*.

[Keck et al., 2006] Keck, M. A., Davis, J. W., and Tyagi, A. (2006). Tracking Mean Shift Clustered Point Clouds for 3D Surveillance. *ACM Press, Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*.

[Khan and Shah, 2000] Khan, S. and Shah, M. (2000). Tracking People in Presence of Occlusion. *Asian Conference on Computer Vision*, pages 263–266.

[Khan and Shah, 2003] Khan, S. and Shah, M. (2003). Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355– 1360.

[Khan and Shah, 2006] Khan, S. M. and Shah, M. (2006). A Multiview Approach to Tracking People in Crowded Scenes using a Planar Homography Constraint. *9th European Conference on Computer Vision*.

[Kim et al., 2005] Kim, K., Chalidabhongse, T. H., Harwood, D., and Davis, L. (2005). Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*.

[Kim and Davis, 2006] Kim, K. and Davis, L. S. (2006). Multi-Camera Tracking and Segmentation of Occluded People on Ground Plane using Search-Guided Particle Filtering. *European Conference on Computer Vision (ECCV)*.

[Koller et al., 1993] Koller, D., Daniilidis, K., and Nagel, H.-H. (1993). Model-Based Object Tracking in Monocular Image Sequences. *Kluwer Academic Publishers*.

[Krumm et al., 2000] Krumm, J., Harris, S., Meyers, B., Brumitt, B., Hale, M., and Shafer, S. (2000). Multi-Camera Multi-Person Tracking for EasyLiving. *Third IEEE International Workshop on Visual Surveillance*.

[Lipton et al., 1998] Lipton, A., Fujiyoshi, H., and Patil, R. (1998). Moving Target Classification and Tracking from Real-Time Video. *Proceedings of the IEEE Workshop Applications of Computer Vision*, pages 129–136.

[Mansouri, 2002] Mansouri, A.-R. (2002). Region Tracking via Level Set PDEs without Motion Computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):947–961.

[McKenna et al., 2000] McKenna, S. J., Jabri, S., Duric, Z., Rosenfeld, A., and Wechsler, H. (2000). Tracking Groups of People. *Computer Vision and Image Understanding*, 80(1):42–56.

[Mittal and Davis, 2003] Mittal, A. and Davis, L. S. (2003). M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. *International Journal of Computer Vision*, 51(3):189–203.

[Moeslund et al., 2006] Moeslund, T., Hilton, A., and Kruger, V. (2006). A Survey of Advances in Vision-Based Human Motion Capture and Analysis. *Journal of Computer Vision and Image Understanding*, 104(2-3).

[Oliver et al., 2000] Oliver, N. M., Rosario, B., and Pentland, A. P. (2000). A Bayesian Computer Vision System for Modeling Human Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Orwell et al., 1999] Orwell, J., Remagnino, P., and Jones, G. A. (1999). Multi-Camera Colour Tracking. *Proceedings of IEEE International Workshop on Visual Surveillance*, 28(4).

[Park and Trivedi, 2006] Park, S. and Trivedi, M. M. (2006). Analysis and Query of Person-Vehicle Interactions in Homography Domain. *IEEE Conference on Video Surveillance and Sensor Networks (VSSN2006)*.

[Prati et al., 2003] Prati, A., Mikic, I., Trivedi, M. M., and Cucchiara, R. (2003). Detecting Moving Shadows: Algorithms and Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Pérez et al., 2002] Pérez, P., Hue, C., Vermaak, J., and Gangnet, M. (2002). Color-Based Probabilistic Tracking. *Proceedings of the 7th European Conference on Computer Vision-Part I (ECCV 2002)*, pages 661–675.

[Qu et al., 2007] Qu, W., Schonfeld, D., and Mohamed, M. (2007). Distributed Bayesian Multiple-Target Tracking in Crowded Environments Using Multiple Collaborative Cameras. *EURASIP Journal on Advances in Signal Processing*, 2007.

[Roth et al., 2005] Roth, D., Doubek, P., and Gool, L. V. (2005). Bayesian Pixel Classification for Human Tracking. *Proceedings of the IEEE Workshop on Motion and Video Computing*, 2:78–83.

[Rousseeuw and Leroy, 1987] Rousseeuw, P. J. and Leroy, A. M. (1987). Robust regression and outlier detection. *Wiley Series In Probability And Mathematical Statistics*, page 329.

[Sen-Ching et al., 2004] Sen-Ching, Cheung, S., and Kamath, G. (2004). Robust techniques for background subtraction in urban traffic video. *Video Communications and Image Processing, SPIE Electronic Imaging*.

[Senior, 2002] Senior, A. (2002). Tracking People with Probabilistic Appearance Models. *Proceedings of the IEEE International Workshop Performance Evaluation of Tracking and Surveillance*, pages 48–55.

[Senior et al., 2006] Senior, A., Hampapur, A., Tian, Y.-L., Brown, L., Pankanti, S., and Bolle, R. (2006). Appearance models for occlusion handling. *Image and Vision Computing*, 24(11):1233–1243.

[Stauffer and Grimson, 1999] Stauffer, C. and Grimson, W. (1999). Adaptive Background Mixture Models for Real-Time Tracking. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Stauffer and Grimson, 2000] Stauffer, C. and Grimson, W. E. L. (2000). Learning Patterns of Activity Using Real-Time Tracking. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 22(8):747–757.

[Trivedi et al., 2005] Trivedi, M. M., Gandhi, T., and McCall, J. (2005). Looking-In and Looking-Out of a Vehicle: Selected Investigations in Computer Vision based Enhanced Vehicle Safety. *Proceedings on IEEE International Conference on Vehicular Electronics and Safety*.

[Valera and Velastin, 2005] Valera, M. and Velastin, S. (2005). Intelligent Distributed Surveillance Systems: A Review. *IEE Proceedings - Vision, Image, and Signal Processing*, 152(2):192–204.

[Veenman et al., 2001] Veenman, C. J., Reinders, M. J., and Backer, E. (2001). Resolving Motion Correspondence for Densely Moving Points. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 23(1):54–72.

[Wang and Suter, 2005] Wang, H. and Suter, D. (2005). A Re-Evaluation of Mixture-of-Gaussian Background Modeling. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2005)*.

[Welch and Bishop, 2001] Welch, G. and Bishop, G. (2001). An Introduction to the Kalman Filter.

[Wren et al., 1997] Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. P. (1997). Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Xu and Puig, 2005] Xu, L.-Q. and Puig, P. (2005). A Hybrid Blob- and Appearance-Based Framework for Multi-Object Tracking through Complex Occlusions. *Proceedings 2nd Joint IEEE International Workshop on VS-PETS*, pages 73–80.

[Yang and Levine, 1992] Yang, Y.-H. and Levine, M. D. (1992). The Background Primal Sketch: An Approach for Tracking Moving Objects. *Machine Vision and Applications*, 5(1):17–34.

[Yilmaz et al., 2006] Yilmaz, A., Javed, O., and Shah, M. (2006). Object Tracking: A Survey. *ACM Computing Surveys*, 38(4).

[Yue et al., 2004] Yue, Z., Zhou, S. K., and Chellappa, R. (2004). Robust Two-Camera Tracking Using Homography. *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04).*, iii- 1-4:3.

[Zhao and Nevatia, 2004] Zhao, T. and Nevatia, R. (2004). Tracking Multiple Humans in Complex Situations. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 26(9):1208–1221.

[Zhong and Sclaroff, 2003] Zhong, J. and Sclaroff, S. (2003). Segmenting Foreground Objects from a Dynamic Textured Background via a Robust Kalman Filter. *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV'03) - Volume 2.*