Visual Urban Scene Analysis by Moving Platforms

A dissertation submitted to the

ETH ZURICH

for the degree of

Doctor of Science ETH

presented by

Andreas Ess
M.Sc. ETH Zurich
born 15. December 1980
citizen of Austria

accepted on the recommendation of

Prof. Dr. Luc Van Gool, ETH Zurich and KU Leuven, examiner
Prof. Dr. Ram Nevatia, University of Southern California, co-examiner
Prof. Dr. Bastian Leibe, RWTH Aachen University, co-examiner

2009
To my family.
Abstract

In recent years, interest in mobile robots and intelligent vehicles that are able to act autonomously in scenarios of daily human living has been increasing constantly. This trend has been further fostered by advances in input sensor technology such as cameras or lasers, as well as in the corresponding algorithms for data interpretation. Due to their relatively low price and similarity to human vision, cameras are an especially intriguing sensor for such tasks.

This dissertation presents a purely vision-based system for the tasks of self-localization, scene analysis, and object tracking in semi-crowded urban environments, with a mobile platform as observer. Starting from a set of off-the-shelf components for object detection and stereo depth estimation, we first propose a probabilistic combination of these independent cues to simultaneously validate object hypotheses and estimate a common ground plane where these objects reside. Based on the improved detections and self-localization from visual odometry, we then introduce a multi-object tracking system that recovers the objects’ trajectories while explicitly reasoning about physical space requirements between each other. The different algorithms are closely coupled by a set of feedback channels, with failure detection mechanisms in each component to ensure robust operation in crowded scenarios.

The resulting, integrated mobile vision system is then applied to the task of pedestrian and car tracking in urban environments, where we demonstrate robust and computationally efficient tracking over prolonged time spans. Taking this system as a basic component, we then explore its use for creating dynamic occupancy maps that could serve as input to a path planning module. For pedestrians, we moreover show that the system can be used as a building block for articulated multi-body tracking; and that the motion model for pedestrians can be improved when
incorporating the interaction of agents along with the knowledge about the scene in an approach inspired by social simulations. Furthermore, we propose a method to analyze the scene by segmenting it into a set of texture labels. The resulting intermediate representation can be used to infer the road type ahead of the observer as well as the presence of various object classes, even without employing a dedicated detector.

All algorithms are evaluated on several challenging, realistic video sequences recorded in busy inner-city locations with a set of representative platforms. Our experiments corroborate our claim that sensor-related robotic tasks in daytime navigation and object tracking can be performed using vision only instead of using often considerably more expensive sensor arrays.
Zusammenfassung

In den letzten Jahren nahm das Interesse an mobilen Robotern und intelligenten Autos, die sich selbständig in natürlichen Umgebungen zurechtfinden können, ständig zu. Dieser Trend wurde weiter gefördert durch Fortschritte im Bereichen von Sensortechnologien, wie z.B. Kameras oder Laser, als auch den entsprechenden Algorithmen zur Dateninterpretation. Aufgrund ihres relativ niedrigen Preises und ihrer Ähnlichkeit zum menschlichen Sehen sind vor allem Kameras interessante Sensoren für Aufgaben im Bereich der Robotik.


Das daraus entstehende, integrierte mobile Bilderkennungssystem wird dann auf die Verfolgung von Fussgängern und Autos in städtischen Umgebungen angewandt, wobei eine robuste und rechnerisch effiziente Objektverfolgung über längere Zeiträume demonstriert wird. In einem wei-
ZUSAMMENFASSUNG

Zusammenfassung

Acknowledgements

This thesis would not have been the same without the invaluable support from a number of great people.

First and foremost I thank my advisor Prof. Dr. Luc Van Gool for offering me the Ph.D. position at his renowned lab, providing me with an inspirational working environment. His continuing support of my work, his stimulating suggestions, and expert advice to all aspects of my research were priceless. My thanks also go to Prof. Dr. Ram Nevatia for his interest in my work and agreeing to co-examine this thesis.

I am deeply grateful to my supervisors Prof. Dr. Bastian Leibe and Prof. Dr. Konrad Schindler for their extraordinary support, countless fruitful discussions that taught me a lot about research, and for being a driving force behind many publications, with late nights before conferences spent on getting all the details right.

Special thanks go to my master thesis supervisor Herbert Bay for siphoning me into the field of Computer Vision and being a constantly positively thinking friend.

All the people at and around BIWI provided a working atmosphere one always wants to return to. I would like to especially thank Michael Breitenstein, Nico Cornelis (KUL), Stephan Gammeter, Helmut Grabner, Raphael Höver, Tobias Jägglí, Stefano Pellegrini, Till Quack, Stefan Saur, Luciano Spinello (ASL), and Thibaut Weise for all the work- and non-work related discussions throughout the years, and more importantly for being great friends.

Special mention also goes to the table soccer crew Axel Krauth, Bryn Lloyd, Robert McGregor, Daniel Roth, and Philipp Zehnder; and the secretaries Christina Krüger, Vreni Vogt, and Barbara Widmer for their organizational magic.
My students also deserve credit, especially Tobias Müller and Manthos Takidis, for their dedication to their projects and valuable input to some new parts of the mobile vision system.

I am furthermore grateful to the great people at Toyota Motor Corporation for their challenging demands and financial support, and of course the fruitful collaborations and the culinary nights out at their Shizuoka, Japan site. Same goes for the partners from the European projects DIRAC and EUROPA, for interesting co-operations and financial support.

I would also like to express my gratitude to the late Rainer Aberer and Sharen Severson, who’ve been inspirational guides to me, especially in my early years as an engineer.

Last but not least, I would like to thank my friends and family, especially my parents who have fostered and believed in me throughout the years; and Olga for her love and bearing with overhours and prolonged trips to conferences.
# Contents

1 Introduction 1
   1.1 Contributions ............................................. 2
   1.2 Organization .............................................. 3

2 Capturing Setup and Preliminaries 7
   2.1 Employed Setups ........................................... 8
   2.1.1 Data Sets .............................................. 9
   2.2 Data Preprocessing ......................................... 10
   2.2.1 Camera Calibration ...................................... 11
   2.2.2 Radial Distortion ....................................... 12
   2.2.3 Debayering ............................................. 13
   2.3 Object Detectors ........................................... 15
   2.3.1 Pedestrian Detection .................................... 16
   2.3.2 Car Detection .......................................... 18
   2.3.3 Extension: Red Backlight Classification .............. 19
   2.3.4 Extension: Partial Occlusion Handling ............... 21
   2.4 Depth from Stereo ......................................... 24
   2.4.1 Employed Algorithms ................................... 26
   2.4.2 Confidence map ......................................... 28
   2.4.3 Accuracy ............................................... 29
   2.4.4 Occupancy Maps ....................................... 31
   2.5 Visual Odometry ........................................... 32
   2.5.1 Related Work .......................................... 33
   2.5.2 Algorithm .............................................. 33

3 Probabilistic Scene Analysis 37
   3.1 Related Work ............................................. 39
   3.2 Problem Formulation ....................................... 41
   3.3 Graphical Model ........................................... 42
3.3.1 Ground Plane ........................................... 44
3.3.2 Object Hypotheses ..................................... 45
3.3.3 Depth Map ............................................... 47
3.4 Training .................................................... 48
  3.4.1 Ground Plane ........................................ 48
  3.4.2 Object Hypotheses ................................... 49
  3.4.3 Depth Cues ........................................... 50
3.5 Inference ................................................... 51
  3.5.1 Belief Propagation ................................... 52
  3.5.2 Interaction Modeling ................................ 53
3.6 Results ..................................................... 55
  3.6.1 Systematic Experiments ............................... 56
  3.6.2 Experimental Evaluation .............................. 58
3.7 Preparation for Tracking ................................. 63
  3.7.1 Positioning .......................................... 63
  3.7.2 Clustering .......................................... 64
3.8 Conclusion ................................................ 65

4 Multi-Object Tracking ....................................... 67
  4.1 Related Work ........................................... 67
  4.2 Overview ............................................... 73
  4.3 Data Association ....................................... 76
    4.3.1 Dynamic Models ................................... 77
    4.3.2 Appearance Model ................................ 82
  4.4 Hypothesis Generation .................................. 83
    4.4.1 Independent Generation ............................. 84
    4.4.2 Trajectory Extension ............................... 86
    4.4.3 Occlusion Reasoning ............................... 86
  4.5 Interactions and Model Selection ....................... 88
    4.5.1 Optimization ..................................... 90
  4.6 Implementation Details ................................ 92
    4.6.1 Hypothesis Pruning ................................ 92
    4.6.2 Identity Management ............................... 93
    4.6.3 Trajectory Initialization and Termination ....... 94
  4.7 Results ................................................. 94
  4.8 Conclusion .............................................. 104

5 Integration & Extensions .................................. 107
5.1 Related Work ........................................ 108
5.2 Integration .......................................... 110
  5.2.1 Runtime ........................................ 110
5.3 Cognitive Loops .................................... 111
  5.3.1 Scene Analysis .................................. 111
  5.3.2 Visual Odometry ................................ 113
5.4 Failure Detection ................................. 117
  5.4.1 Visual Odometry ................................ 117
  5.4.2 Object Tracking ................................. 119
  5.4.3 Object Detection and Ground Plane Estimation . 119
5.5 Dynamic Occupancy Maps ....................... 119
  5.5.1 Method .......................................... 121
  5.5.2 Results .......................................... 123
5.6 Articulated Tracking ............................ 126
  5.6.1 Related Work .................................. 128
  5.6.2 Overview ...................................... 130
  5.6.3 3D Articulated Tracking ....................... 131
  5.6.4 Guided Adaptive Segmentation ............... 136
  5.6.5 Results ......................................... 140
5.7 Conclusion .......................................... 149

6 Modeling Social Behavior for Tracking .... 151
  6.1 Related Work ...................................... 153
  6.2 Modeling Social Behavior ....................... 155
    6.2.1 Static Obstacles .............................. 159
    6.2.2 Application of the Model ................... 159
  6.3 Training .......................................... 160
  6.4 Results .......................................... 161
    6.4.1 Prediction .................................... 162
    6.4.2 Patch-based Tracking ....................... 164
    6.4.3 Tracking with a Moving Observer .......... 166
  6.5 Conclusion ...................................... 168

7 Patch-based Scene Analysis .................. 171
  7.1 Related Work ..................................... 172
  7.2 Patch Classification ............................ 174
    7.2.1 Preprocessing ................................ 175
    7.2.2 Features ..................................... 176
7.2.3 Classifier ........................................ 179
7.2.4 Application ................................... 179

7.3 Spatial Smoothing .................................. 180
7.3.1 Training ......................................... 181
7.3.2 Inclusion of Objects ................................. 183
7.3.3 Application ...................................... 186

7.4 Segmentation-Based Urban Traffic Scene Understanding . 186
7.4.1 Features ......................................... 186
7.4.2 Learning .......................................... 188
7.4.3 Temporal Smoothing ................................. 189
7.4.4 Application ...................................... 189

7.5 Dataset ............................................ 189

7.6 Results ........................................... 191
7.6.1 Patch Classification ................................. 191
7.6.2 Traffic Scene Understanding ......................... 198

7.7 Conclusion ......................................... 203

8 Conclusions and Outlook ............................. 205
8.1 Contributions ...................................... 206
8.2 Perspectives ....................................... 208

A Class-Dependent Neighborhood ...................... 213
A.1 Optimization ....................................... 214

List of Publications .................................. 217

Bibliography .......................................... 219

Curriculum Vitae ...................................... 241
1

Introduction

Visual understanding of a scene has been the far-end goal of Computer Vision since the 1970ies. So far, however, efforts have often been stymied by the variability and the sheer complexity of many real-world scenes. One particularly interesting sub-problem is the understanding of urban street scenes, seen from an unconstrained mobile observer. This is a crucial requirement for many applications in the near future of mobile robotics and smart vehicles, such as path planning, driver assistance, or the general assessment of traffic situations. The observer could be any kind of platform, *e.g.* a robot, a car, or even a person wearing a backpack. Disregarding the specific implementation of the platform, we will use cameras as the only input sensor to our system. Specifically, most of the presented algorithms will employ the information from a pair of forward-looking cameras with an overlapping field of view, yielding additional information in depth.

In a sense, this dissertation thus tries to push the envelope on how far one can go with vision as the only sensor for self-localization, tracking, and scene analysis. Nowadays, cameras abound and are often cheaper than even a single sensor of the usually employed sensor arrays in robotics. On the downside, vision-inferred measurements can be brittle when compared to some of these more specialized modalities: vision-based self-localization will not be as drift-free as one fusing GPS and inertial measurement units; and distance estimates will be less accurate than those obtained with laser sensors. Still, as we will show in this thesis, one can gain a lot of useful information from camera images alone, allowing for a system that can safely operate in environments of daily human living.
Especially when observed from moving cameras, urban scenarios put extreme demands on the underlying vision algorithms: besides basic image artifacts due to motion blur, insufficient lighting, or cast shadows, the scene itself is often highly cluttered. Its analysis not only involves static structures such as the road, buildings, or street furniture, but also a variety of moving agents. In typical cases, a fair number of these will move independently throughout the scene, crossing and occluding each other, or being occluded by other standing structures. The low viewpoint dictated by constraints on the platform further reduces visibility and complicates localization in depth. Analyzing such a scene, especially keeping track of moving objects, will be one of the main topics of this work.

In recent years, many individual disciplines of Computer Vision, such as object detection, have advanced to a state where algorithms are becoming applicable for real-world tasks. Still, their application to such complex and unconstrained scenarios requires the interplay of multiple such components. This thesis is an attempt at combining recent research efforts in object detection, depth estimation, visual odometry, and tracking in order to construct a system that is able to reliably track multiple objects—here, pedestrians and cars—in challenging urban scenarios. Going further, we will also demonstrate possible applications of the system in path planning, articulated tracking, and advanced motion models; as well as first experiments in vision-based traffic scene recognition.

1.1 Contributions

The contributions of this thesis can be summarized as follows.

1. We present a mobile vision system that is able to simultaneously estimate its own location and the rough geometry of the scene, detect objects, and track them over time in challenging real-world scenarios and from video input. The approach integrates and closely couples the different vision components in a combined system.

2. Operating in a system view, we integrate multiple vision-based modules to arrive at a robust system that can reliably track more
than 15 people in almost real-time. During the entire approach, we specifically address the question of how to avoid system instabilities and guarantee robust performance. This is done by incorporating automatic failure detection and correction mechanisms, which work together to avoid error amplification.

3. In all modules, we employ a *hypothesize-and-test paradigm*, showing that later pruning in the presence of more information can effectively help in stabilizing a system: in visual odometry, camera pose hypotheses are verified against a geometric model with the help of object detection; object hypotheses from a common object detector have higher precision when verified against a model of the scene; and object trajectory hypotheses are more accurate when fit jointly rather than independently, because one can account for physical exclusion constraints.

4. In traffic situations, the information about an object’s approximate distance is a helpful, sometimes indispensable cue. Therefore, we point out how the *principled use of stereo data* can improve system performance in the various modules. As a largely independent source of information, it cannot only help in the localization of objects in depth, but also with structure analysis and hence image classification.

5. We recorded *several hours of test video* from a pair of synchronized cameras, mounted on various mobile platforms, both cars and robot-like. Many of the sequences are annotated, allowing for a thorough analysis of the different parts of the system. These sequences were also made available to the research community, thus allowing for comparisons between different approaches developed by researchers.

### 1.2 Organization

This thesis is structured as follows.

In Chapter 2, *Setup and Preliminaries*, several basics for the following chapters are introduced. This encompasses the mobile platforms we
constructed or used for creating our test data set, an introduction into the basic components used for object detection and depth generation, as well as a brief explanation of the developed visual odometry system.

In Chapter 3, *Probabilistic Scene Analysis*, a probabilistic method for single-frame scene analysis is described. The proposed method combines cues from object detection, stereo depth estimation, and ground plane reasoning in a Bayesian network. This network allows to simultaneously estimate a valid ground plane and according object detections. The system is tested on several long sequences, achieving a considerable improvement in detector precision with this additional information, irrespective of which actual detector is used as hypothesis generator. The output of this stage is hence also used for the upcoming tracking chapter, as it effectively filters detections and helps in 3D localization of objects using the ground plane. The chapter is based on research originally presented in [Ess et al., 2007a].

In Chapter 4, *Multi-Object Tracking*, a detection-based multi-object tracking system is presented. The tracker operates in a hypothesize-and-select framework, first generating a set of possible explanations for the current time-window and then applying an optimization strategy to find the mutually best solution. Operating in 3D world coordinates, the system can model actual physical exclusion between objects and can handle arbitrary object classes given their basic motion model. We demonstrate this for both pedestrians and cars and analyze the resulting system on several long and challenging sequences. This chapter summarizes the tracking framework introduced in [Ess et al., 2008; 2009d] with considerably more detail, and additional results on car tracking and an improved global optimization stage.

In Chapter 5, *Integration & Extensions*, the integration of the modules from the previous chapters (visual odometry, scene analysis, and object tracking) is investigated in more detail, yielding a robust, integrated system that is able to perform tracking in challenging scenarios in near real-time. We show that in order to construct such a system, feedback channels between the different modules, including failure detection, are of prime importance. Furthermore, we demonstrate how to employ the resulting system as a component for higher-level tasks. Specifically, its application to the construction of dynamic occupancy maps, as well as articulated multi-body tracking is investig-
1.2. Organization

ated. The chapter fuses research presented in [Ess et al., 2009a; 2009b; Gammeter et al., 2008] and adds some further experiments.

In Chapter 6, Modeling Social Behavior for Tracking, a dynamic model for pedestrian motion inspired by social simulations is presented. Specifically, we propose to replace the common, independent linear extrapolation by a model that simultaneously predicts all agents’ future motion, avoiding collisions and taking into account possible goal locations. This model is based on a social simulation whose parameters are optimized from semi-crowded top-down videos and then applied to multi-person tracking. We show that using such a model can help re-finding people after occlusions and generally gives better predictions, and should thus also be better suited for path planning. The chapter is largely based on [Pellegrini et al., 2009].

In Chapter 7, Patch-based Scene Analysis, a texture-based classifier is explored as a way to a more holistic scene understanding. The proposed method is able to segment an image into a set of urban texture classes, such as building, street, or car. Similar to Chapter 3, we demonstrate how joint inference over class labels and object detections in a Markov random field can improve both modules, yielding a better labeling as well as better precision for objects. Additionally, we show how to use the labeling as an intermediate representation, based on which one can automatically infer the upcoming road type and detect the presence of objects such as pedestrian crossings or cars. The latter method compares favorably to a state-of-the-art scene descriptor, significantly outperforming it for object classes. This chapter is an extended version of the research published in [Ess et al., 2009c].

Chapter 8 concludes the dissertation and gives an outlook on possible future research directions.

Due to the relative independence of the research areas, all chapters but Chapter 2 have separate literature reviews. Furthermore, most of the chapters, except for Chapter 5, are self-contained, and can therefore be read independently.
2

Capturing Setup and Preliminaries

In this thesis, we will mainly focus on vision algorithms for mobile platforms that are equipped with a pair of synchronized, forward-looking cameras. The advantage of such a system is that depth maps can be readily constructed and used as a powerful asset in various vision-based algorithms. However, at the time our experiments were started, no appropriate data set for testing the algorithms was available. We thus recorded several data sets with different platforms [Ess et al., 2007a; 2008; 2009a] that have meanwhile been used in several other publications of the vision community. In this chapter, we will first briefly describe the capturing methodology and the data sets used throughout the rest of the thesis.

As raw input to our system, we will use off-the-shelf implementations of object detectors, as well as stereo estimators. Due to recent research efforts, a plethora of options is available for both problems. We will investigate a few viable alternatives for both and briefly review this chosen subset. Furthermore, we will discuss the implications of using depth information. Note that while most algorithms in this work were designed with a stereo setup in mind, many parts can be readily applied to single camera streams as well.

In Section 2.1, we will first describe the employed capturing setups, along with the data sets used throughout the rest of the thesis. In Section 2.2, we briefly summarize the preprocessing steps to obtain an image from the raw camera data. We then discuss the input cues, including both
object detection (Section 2.3) and depth map generation (Section 2.4). Finally, Section 2.5 describes the employed visual odometry system.

2.1 Employed Setups

To assess the performance of our system for both robotic and autonomous driving applications, four different platforms were constructed or employed and a set of videos was recorded with each (see Fig. 2.1). In all cases, a pair of forward-looking AVT Marlin F033C cameras\(^1\) was used, which delivered a resolution of \(640 \times 480\) pixels (Bayered) at about 13–14 frames per second. The cameras use an external trigger to give perfectly synchronized images. The trigger is controlled via a USB interface based on the IOwarrior chip\(^2\). Internally, the AVT cameras operate at a higher bit depth, we use the built-in Gamma correction to compress the dynamic range into the commonly used 8 bit. The shutter time is usually fixed while the gain is set to automatic to allow for slight changes in brightness. To optionally normalize the images, the camera’s chosen gain was logged for every frame.

The first platform, termed CharioBot, is a child stroller with the cameras mounted at a height of about 1 m and a baseline of 40 cm. The employed lenses have a focal length of 4.8 mm, giving a field-of-view of about 65° with acceptable radial distortion artifacts. This platform is thought as a substitute for a robot roaming through busy pedestrian zones. As

---

\(^1\)http://www.alliedvisiontec.com/avt-products/cameras/marlin/f-033-b-bs-c.html

\(^2\)http://www.codemercs.de/
data was recorded in winter, the streams sometimes suffer from missing contrast.

The second platform, CharioBot Mk. II, is a very similar setup, with cameras mounted slightly lower and closer to the vehicle’s front, resulting in close pedestrians being only partially visible. This platform was used for recordings on a few sunny days in fall; the videos are thus richer in contrast, and people wear more varied clothing. Again, the recorded sequences are interesting for prototyping robotic applications; we will also use them to demonstrate the breaking point of the system.

For datasets resembling automotive applications, we used the SmartTer vehicle [Kolski et al., 2007], with the cameras mounted on the roof at a baseline of about 60 cm, and a focal length of 8 mm, corresponding to a field-of-view of about 50°. The elevated viewpoint (1.3 m) results in less occlusions but more objects at the same time, and the higher speed makes tracking more challenging, as many people are often only seen for very short timespans. The videos used in this thesis are sub-sequences from a 30 minute long drive through the center of Zurich in winter, again with sometimes bleak contrast and pedestrians wearing similar clothing, thus making appearance-based data association difficult.

The system was also tested using the AWEAR platform [Havlena et al., 2009], a backpack featuring two frontal-facing panoramic cameras. As depth reconstruction from such cameras was not available at the time of development, only the left camera’s undistorted and stabilized output was used to test the system on a few short sequences. A few results obtained using this setup are presented in Chapter 5.

### 2.1.1 Data Sets

With each of the setups, a set of sequences was recorded. An overview of the resulting datasets is given in Tab. 2.1. These sequences form the basis for the system evaluation in this thesis. In many of them, the bounding boxes of pedestrians with a height $> 60$ pixels were annotated for quantitative evaluation. In some cases, only every fourth frame is annotated, indicated by a (4) in the table. Example images can be found in the results sections of the upcoming chapters. A brief description is given in the following.
Table 2.1: Overview of used test sequences (platform, frames, approx. travelled distance, pedestrian annotations).

Seq. Bahnhofstrasse was taken on a cloudy day, strolling on a sidewalk. Its main challenges are a large number of trees and dust bins that result in false positives from object detection, persons getting off public transport, as well as reflections from shopping windows. Seq. Jelmoli shows a stroll over a busy square, with people moving in all directions. The square lies in the shade, resulting in low contrast and thus increased difficulty for the detector.

Seq. Linthescher was recorded with CharioBot Mk. II, and shows a walk through a fairly busy square, with people often obstructing large parts of the camera’s view. Seq. Paradeplatz is a short sequence over a square, with people stopping abruptly and frequently occluding each other. Seq. Crossing starts at a busy pedestrian crossing before continuing on a sidewalk, again with many only partially visible people obstructing the view, which often leads to detector failure.

Finally, in order to demonstrate the system’s applicability to automotive tasks, we use three sequences recorded with the SmartTer platform, where Seq. Loewenplatz is a drive through an area with many pedestrians, whereas Seq. City and Seq. Bellevue will be used to demonstrate the system’s ability to track cars and pedestrians at the same time.

2.2 Data Preprocessing

The image formation process for a single camera is shown in Fig. 2.2. After passing the lens (or an array thereof), the incoming light reaches
2.2. Data Preprocessing

![Image formation diagram](image.png)

**Figure 2.2**: Image formation for a single camera of the employed setup.

the imaging sensor (here, a CCD chip) via a so-called Bayer pattern [Bayer, 1976]. This pattern is a filter mask that is tuned to a different spectral bandwidth for every pixel, effectively allowing the recording of color images. Usually, cameras themselves offer the option to assemble a color image from this raw data. We will perform this processing outside of the camera for increased flexibility.

After describing the calibration of the lens including radial distortion, we will detail the process of debayering. The resulting debayered and undistorted image will be denoted $I$ throughout the rest of this thesis.

### 2.2.1 Camera Calibration

For the lenses, we assume the standard pinhole camera model [Hartley and Zisserman, 2004], where an internal calibration matrix suffices to characterize a camera:

$$
K = \begin{bmatrix}
    f_u & s & p_u \\
    0 & f_v & p_v \\
    0 & 0 & 1
\end{bmatrix},
$$

with $f_u$ and $f_v$ denoting the focal length of the camera in pixel units in horizontal and vertical direction, respectively. Their aspect ratio is given by $s_{vu} = f_u/f_v$, in our case $s_{vu} \approx 1$. $s$ is the skew of the pixels, we assume $s = 0$. $p_u$ and $p_v$ are the coordinates of the principal point, i.e., the intersection of the camera’s optical axis with the image plane. For all setups, these values were determined using the MATLAB Calibration Toolbox.\(^3\)

\(^3\)http://www.vision.caltech.edu/bouguetj/calib_doc/
**Projection.** $K$ transforms points from 3D camera coordinates to the 2D projective space according to the following equation:

$$\mathbf{x} = K \mathbf{X}_{\text{cam}},$$  \hspace{1cm} (2.2)

with the notation $\mathbf{X}_{\text{cam}}$ emphasizing the assumption that the camera is located at the origin of the Euclidean coordinate system (camera center $C = 0$). The principal axis coincides with the $z$-axis.

The obtained point $\mathbf{x}$ lies in the space of 2D homogeneous coordinates. To arrive at the actual 2D image point $(u, v)$, it is projected by dividing through the third, homogeneous component,

$$u = x_1/x_3 \quad v = x_2/x_3 \quad .$$  \hspace{1cm} (2.3)

**Backprojection.** Given an image point, the ray on which the corresponding 3D point resides is obtained as

$$\mathbf{V}_{\text{cam}} = K^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad .$$  \hspace{1cm} (2.4)

In order to obtain the actual 3D point, one needs to know its depth $d$. Then, $\mathbf{X}_{\text{cam}} = \mathbf{V}_{\text{cam}}d$. $d$ could be obtained from a depth map or another distance sensor. Alternatively, if the same point is observed from another camera, the two corresponding rays can be intersected to obtain the 3D point. This is referred to as triangulation [Hartley and Zisserman, 2004].

### 2.2.2 Radial Distortion

As the CharioBot platform has a rather wide-angle lens, it is important to account for aberrations from the linear pinhole camera model due to lens distortions. In this thesis, we will restrict ourselves to a radial distortion model which accounts for the predominant effects of the employed lenses.

Radial distortion is dependent on the distance $r$ from the optical axis. Usually, the distortion function is modeled as a symmetric polynomial centered at the principal point,

$$L(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 \quad .$$  \hspace{1cm} (2.5)
2.2. DATA PREPROCESSING

Figure 2.3: Debayering on Seq. Bahnhofstrasse. (a) Original input image from camera. (b) Image debayered with Larocheprescott algorithm, including close-up of typical bayering artifact. (c) Undistorted image.

Before projection on the image plane, a point \((\tilde{x}, \tilde{y})\) with distance \(\tilde{r}\) from the center is thus distorted nonlinearly,

\[
\begin{pmatrix}
  x_d \\
  y_d
\end{pmatrix} = L(\tilde{r}) \begin{pmatrix}
  \tilde{x} \\
  \tilde{y}
\end{pmatrix}.
\]  (2.6)

After distortion, the calibration is applied to arrive at the actual image point.

To circumvent the solution of the above polynomial when undistorting an image, it is common to reverse the process by performing one lookup per pixel in the undistorted image to its distorted counterpart. We perform this in parallel on the GPU, using bilinear interpolation for better results.

2.2.3 Debayering

One of the most typical and cheapest methods to digitize a color image is Bayering [Bayer, 1976]. Instead of using multiple CCD chips with different filters to account for respective color bands, a mosaic of filters is placed in front of a single image sensor (Fig. 2.4 (a)), where each block of four pixels comprises one red, two green, and one blue pixels. Green light is used twice to account for the human eye’s sensitivity. While this method can save money and chip space, it comes at the expense of an
image that needs to be de-mosaiced, or debayered, before it can be used for further processing.

There exist various algorithms to debayer an image, some based on down-sampling, while others try to keep the original sensor resolution by performing interpolation. We will look at the latter, where available methods range from simple bilinear interpolation over methods that adapt to edges [Laroche and Prescott, 1994] to ones that measure the homogeneity in the image [Hirakawa and Parks, 2005]. In this project, we use the method of [Laroche and Prescott, 1994], which adapts the interpolation depending on whether the predominant edge direction is horizontal or vertical. In the following, $A_i$ can denote either a red or blue pixel, with $C_i$ its blue or red counterpart. Given the Bayer pattern of the employed camera, the two gradients at non-green pixels (Fig. 2.4 (b)) are calculated as

\[
\nabla H = \left| \frac{A_3 + A_7}{2} - A_5 \right| \quad \nabla V = \left| \frac{A_1 + A_9}{2} - A_5 \right| .
\]

The missing green is then interpolated based on the dominant gradient,

\[
G_5 = \begin{cases} 
\frac{G_2 + G_8}{2} & \text{if } \nabla H > \nabla V \\
\frac{G_4 + G_6}{2} & \text{if } \nabla H < \nabla V \\
\frac{G_2 + G_8 + G_4 + G_6}{4} & \text{if } \nabla H = \nabla V
\end{cases}
\]

\[ (2.8) \]
With all green values specified, two cases remain. The first are pixels that have already either red or blue given, with green interpolated, Fig. 2.4 (c). The remaining color can then be calculated as
\[ A_5 = \frac{A_1 - G_1 + A_3 - G_3 + A_7 - G_7 + A_9 - G_9}{4} + G_5 \]. (2.9)

The second case are the original green pixels (Fig. 2.4 (d)), which need to interpolate both red and blue colors from their neighbors,

\[ A_5 = \frac{A_2 - G_2 + A_8 - G_8}{2} + G_5 \] \hspace{1cm} (2.10)

\[ C_5 = \frac{C_4 - G_4 + C_6 - G_6}{2} + G_5 \]. (2.11)

An example debayering obtained with this method is shown in Fig. 2.3 (b). For most objects, this produces clearer edges than linear interpolation. However, hard edges in the image (e.g., the striped blind in the back) produce artifacts. In our experiments, this did not have a major effect on either detectors or depth estimation. While there exist better methods, they will all eventually produce some artifacts. The advantage of this method is its quick implementation on the GPU with only two shader passes, requiring about 2 ms per image.

### 2.3 Object Detectors

In recent years, human detection has reached an impressive level [Dalal and Triggs, 2005; Felzenszwalb et al., 2008; Leibe et al., 2008a; Tuzel et al., 2007; Viola et al., 2005; Wu and Nevatia, 2007a], with many systems also being able to estimate the silhouettes of the detected pedestrians [Gavrila and Munder, 2007; Leibe et al., 2008a; Sharma and Davis, 2007; Wu and Nevatia, 2007b]. For automotive applications, two recent surveys [Dollar et al., 2009; Enzweiler and Gavrila, 2009] conduct extensive experiments over many hours of urban driving to assess the performance of current detection algorithms for automotive tasks. In short, it turns out that a conceptually simple Histogram-of-Oriented-Gradient detector [Dalal and Triggs, 2005] performs very well, especially for frontal
pedestrians at intermediate sizes (> 60 pixels), while Haar-based detectors [Viola et al., 2005] are more suited to finding smaller pedestrians. For larger sizes and more articulated objects, including side views, component-based methods [Leibe et al., 2008a; Felzenszwalb et al., 2008] are expected to perform better.

While many of these algorithms focus on human detection, they can often be applied to the detection of other objects, such as cars. After introducing the employed pedestrian detection algorithms in the upcoming section, we will discuss the car detector including two extensions for better accommodating the challenges of car detection.

2.3.1 Pedestrian Detection

In this thesis, we investigated three different, publicly available detectors: the Implicit Shape Model (ISM) [Leibe et al., 2008a], the Histogram-of-Oriented-Gradient (HOG) detector [Dalal and Triggs, 2005], as well as a recent part-based detector [Felzenszwalb et al., 2008]. For pedestrians, all detectors are applied out-of-the-box with a low confidence threshold. In order to account for the detectors’ rather large minimum scale, the input images are rescaled with bicubic interpolation to twice their original size.

The ISM detector [Leibe et al., 2005; 2008a] is a generative detector based on local feature points that vote for a common object center. During training, local feature points are detected and clustered based on their descriptor similarity, yielding a codebook. Each codebook entry is associated with a distribution of relative object center positions obtained from annotations in the training set. When applying the detector, detected feature points are associated with the codebook, yielding probabilistic votes for object centers. Voting maxima then correspond to object detections. Optionally, the codebook can be coupled with groundtruth segmentations, yielding an approximate “top-down” segmentation for a detection.

A conceptually simpler approach is taken by the HOG detector [Dalal and Triggs, 2005]. Based on gradient features that are binned into histograms, a discriminative classifier (SVM, [Vapnik, 1995]) is learned that can tell, for a given image window, whether this window contains an
2.3. Object Detectors

Figure 2.5: Typical raw detections obtained by applying the detectors of Leibe et al. (top row), Felzenszwalb et al. (center row) and Dalal & Triggs (bottom row) on Seq. LINTHESCHER and Seq. LOEWENPLATZ.

instance of the object or not. This window is then swept over all possible image positions at multiple scales, giving the actual detections after running a non-maximum suppression step.

One major disadvantage of ISM is its reliance on generic image features that are not directly interpretable for object detection. The part-based detector of [Felzenszwalb et al., 2008] therefore combines the advantages of both ISM and HOG and uses discriminative, HOG-based classifiers to identify body parts and then tries to combine them in a probabilistic model. We mostly include this detector in our comparison due to its good performance in the VOC object detection challenge [Everingham et al., 2008].

Sample detections for a few typical images are shown in Fig. 2.5. As is already evident from these images, the HOG detector achieves the best detection results, with both few false positives and few missing detections. This is also in accordance with the findings of [Dollar et al., 2009]. Another advantage of this approach is that it is highly parallelizable. In fact, GPU-based implementations [Wojek et al., 2008; Prisacariu and Reid, 2009] achieve up to 10 frames per second on VGA-
Figure 2.6: Car detection. (a) To capture the orientation-dependent variability of cars, multiple detectors are used. (b,c) Front-back misclassifications are alleviated by applying a red-backlight classifier that measures the red color inside each cell of a coarse grid.

sized images. In our experiments, we will thus mostly use this detector, though we will also take advantage of ISM’s ability to infer a rough pedestrian segmentation for articulated tracking, as described in Chapter 5.

Maximum Distance. The standard INRIA pedestrian model for the HOG detector can detect pedestrians at a minimum scale of $h_{px} = 96$ pixels, or 48 pixels on the double image size. The corresponding maximum distance $d_{max}$ can be found under the assumption of a pedestrian height of $h_w = 1.8$ m, its bounding box in the image center, $x = 320$ px, $y_{bot} = 240$ px,

$$h_w = \left\| d_{max} \left( K^{-1} \begin{pmatrix} x \\ y_{bot} \end{pmatrix} - K^{-1} \begin{pmatrix} x \\ y_{bot} - h_{px} \end{pmatrix} \right) \right\|$$  \hspace{1cm} (2.12)

For the child strollers (CharioBot, CharioBot Mk. II), this corresponds to a distance of about 19 m, for the SmartTer platform, 30 m.

2.3.2 Car Detection

For detecting cars, we will also use the off-the-shelf HOG detector with a custom training set. To account for the changing appearance and
classifier window size under different viewpoints, we will use a total of 7 detectors, distributed as seen in Fig. 2.6 (a). The advantage of using multiple detectors is that the directional information can be used directly in the tracker. On the downside, detection time scales linearly with the number of detectors, and running them independently can often yield multiple detections per object. Using gradient information only, it is especially difficult to tell apart frontal and rear views. This problem can be either solved by using multi-aspect detectors followed by a regression yielding the viewpoint, or by dedicated post-processing steps, as described next.

**Maximum Distance.** The employed car detectors have an average window height of about $h_{px} = 25$ pixels, giving maximum distances of 26 m (CharioBot) and 42 m (SmartTer), respectively, when assuming a vehicle height of 1.3 m.

### 2.3.3 Extension: Red Backlight Classification

The employed battery of car detectors often gives multiple detections for a single object. If we want to take advantage of the directional information from the detectors, it is important to filter out reversed views that cannot be determined by gradient information alone. For that purpose, we develop a classifier that is able to discriminate between rear and non-rear views.

In order to classify rear views, we propose a color-based method that first extracts the red color information in a grid laid over the detection window (Fig. 2.6 (b)) and then uses a support vector machine (SVM, [Vapnik, 1995]) to classify the detection into being a rear view or not. This then allows to filter detections from the rear-view car detector that have non-matching color appearance (and can be either false positive front views or other structures). Similarly, it also allows to filter out front-view detections that are actually rear views.

Specifically, we first calculate the “redness” of each pixel $p$ inside the bounding box as

$$red(p) = \frac{2R(p)}{G(p) + B(p)}, \quad (2.13)$$
with $R$, $G$, and $B$ being the red, green, and blue channels of the image $I$. The output $red$ is then mean-filtered and normalized. Based on this, a histogram of size $3 \times 3$ is constructed, where we use a rather coarse grid to account for detection inaccuracies. For each bin, \[
h_{m,n} = \sum_{p \in \text{cell}(m,n)} \text{red}(p) > \theta ,
\]
\[
\tilde{h}_{m,n} = \frac{1}{m \cdot n} h_{m,n} ,
\]
where the threshold $\theta = 0.5$ in our implementation, c.f. Fig. 2.6 (c). For each rear-view ($60^\circ$, $90^\circ$, $120^\circ$), we train an SVM classifier on this histogram. The positive training set corresponds to the one used by the respective rear-view detector, whereas the negative training set encompasses both reversed views as well as other negatives. Training is then done using 8-fold cross-validation. The respective correct rate is shown in Fig. 2.7 (a).

The red backlight classifier is applied as a post-processing step to every positive HOG response from all car detectors except the side-view ones: for the $60^\circ$, $90^\circ$, and $120^\circ$ detectors, only detections classified as a rear-view by this color-based method are accepted. For the $240^\circ$ and $300^\circ$ views, detections classified as rear-views are discarded. As can be seen in Fig. 2.7 (b,c), the majority of the detected reversed views and also false positives can hence be filtered out.

<table>
<thead>
<tr>
<th>View</th>
<th>Correct rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$60^\circ$</td>
<td>0.8416</td>
</tr>
<tr>
<td>$90^\circ$</td>
<td>0.9394</td>
</tr>
<tr>
<td>$120^\circ$</td>
<td>0.8905</td>
</tr>
</tbody>
</table>

Figure 2.7: (a) Cross validation errors for red backlight detection. (b,c) Example filtering achieved by the red backlight classifier. As input, the rear-view detector is used, the windows classified as rear views by the red backlight classifier are drawn in red.
2.3. Object Detectors

(a) Full  
(b) Left half  
(c) Right half

Figure 2.8: Raw full and half responses of rear-view detectors without occlusion reasoning. The half-detector’s responses are in general less reliable than their full-view counterpart.

2.3.4 Extension: Partial Occlusion Handling

In busy real-world scenarios, occlusions between objects or of objects by the static scenery are frequent. This especially is the case in our application, given the rather low camera placement. Due to their size, cars are especially likely to partially occlude other objects, which often happen to be other cars in the same lane. As the HOG detector per se cannot handle partially occluded objects, we deal with this case explicitly.

To this end, we propose to generate separate detectors for the left and right halves of a car’s front and rear view, accounting for the most typical cases in our application: cars in the same lane, partially occluding each other; and cars being cut off at the image border due to the limited field-of-view. As with the full detector, the part-based detectors are also realized using the HOG detector [Dalal and Triggs, 2005]. The training data thus remains the same, though divided into a left and right half.

Due to the smaller classifier window, a part detector will produce more false positives compared to its full-view counterpart, see Fig. 2.8. In order to extract useful part detections, we therefore propose an occlusion reasoning step similar to [Wu and Nevatia, 2007a] that only accepts part detections which can be explained by the rest of the scene.

\[\text{Recently, [Wang et al., 2009] proposed an extension for HOG that is able to deal with partial occlusions. This would however need some further investigation.}\]
Definitions. A part hypothesis is described by a 3-tuple $\mathbf{rp}_i = \{l_i, \text{bbox}_i, v_i\}$, where $l_i$ is the label indicating the part type (with $FC$, $LH$, and $RH$ representing the full car, the left half, and the right half, respectively), $\text{bbox}_i$ is the bounding box in the image, and $v_i$ is the viewpoint of the car the part response is associated with. The union of all the part detectors’ and full detectors’ output yields the set of part hypotheses $RP = \{\mathbf{rp}_i\}$. Example detections are shown in Fig. 2.8.

The set of all full car detection responses with viewpoint $v$ is denoted as

$$F_v = \{\mathbf{rp}_i | l_i = FC \land v_i = v\}. \quad (2.16)$$

Similarly, the set of all half detection responses associated to view $v$ is defined as

$$P_v = \{\mathbf{rp}_i | l_i \in \{LH, RH\} \land v_i = v\}. \quad (2.17)$$

The hypothesized counterpart of a part response $\mathbf{rp}_i$ is defined as $\mathbf{rp}_j$. I.e., if $\mathbf{rp}_i$ is a left half, then $\mathbf{rp}_j$ denotes the bounding box corresponding to its right half, and vice versa. The set of all counterparts of the half responses is denoted as $\mathbf{P}_v$.

As we want to reason about occlusion caused by any car, it is important to take all detectors’ responses into account. Therefore, similar to the above definition, we define the set of all full-view car detections as

$$Q(I) = \{F_v | v \in \{0^\circ, 60^\circ, 90^\circ, 120^\circ, 180^\circ, 240^\circ, 300^\circ\}\}. \quad (2.18)$$

Problem statement. Given the image $I$ and the corresponding car hypotheses ($Q \cup P_v$) consisting of all full and half responses, we are interested whether $\mathbf{rp}_i \in P_v$ can be explained by the rest of the scene. In other words, we check whether its hypothesized counterpart $\mathbf{rp}_j$ is occluded by $\hat{S}_v \cup Q \cup \overline{I}$. $\hat{S}_v$ denotes the current set of valid responses, as described in the algorithm below. $\overline{I}$ is the complement of $I$, describing the region outside of the image $I$.

Our method starts in every image $I$ from the set of full car detections, $\hat{S}_v = F_v$. The algorithm then iteratively allows additional half-parts $\mathbf{rp}_i$ to be included into the set $\hat{S}_v$, if there is enough evidence to declare its
2.3. Object Detectors

Figure 2.9: Results of applying occlusion reasoning on the full (blue), left half-part (red), and right half-part (green) responses. (a) Input $Q$ and $P_v$ (b) Obtained output.

counterpart $\overline{rp}_i$ as occluded or as lying outside of the image. A half response $rp_i$ is considered occluded, when the expression

$$\frac{|bbox_i \cap \bigcup_{s \in \tilde{S}_v \cup Q} bbox_s|}{|bbox_i|} > \theta$$

(2.19)

is true, where $\theta$ is a threshold lying in $[0, 1]$. The denominator corresponds to the area of $rp_j$’s bounding box, and the nominator is the area of the intersection of $a$ with the union of all elements of $\tilde{S}_v$ and $Q$, respectively. In our experiments, we choose $\theta = 0.8$, meaning that 80% of $\overline{rp}_i$ must be occluded for $rp_i$ to be accepted. The definition of lying outside of the image can be expressed in a similar way.

Extending the initial set of car hypotheses $\tilde{S}_v$ by valid half responses $rp_i \in P_v$ amounts to the iterative process described in Algorithm 1. The algorithm terminates after at most $|P_v|$ iterations, as at each iteration, at least one partial detection is removed, or there is no change, which corresponds to the algorithm’s termination criterion.

Including part detections with this method allows for the earlier detection of partially visible cars, as well as tracking of cars that partially leave the field-of-view. Quantitatively, the maximally reachable recall of the pure detections usually goes up between 2–5%, depending on the scene.
Algorithm 1 Occlusion reasoning for image $I$ and viewpoint $v$.

**Input:**
- $\tilde{S}_v$ // Initial set of full detections with viewpoint $v$
- $Q$ // Output of all full-detectors
- $P_v$ // Part-detector responses

**Output:**
- $\tilde{S}_v$ // Full detections with viewpoint $v$, including valid half detections

$B = 0$

while $B \neq |\tilde{S}_v|$ do
  // Repeat until no change
  $B = |\tilde{S}_v|$  
  for all $rp \in P_v$ do
    if $\overline{rp}$ outside $I$ then
      // Accept, as other part is outside of image
      $\tilde{S}_v \leftarrow \tilde{S}_v \cup rp$
      $P_v \leftarrow P_v \setminus rp$
    else
      for all $s \in (\tilde{S}_v \cup Q)$ do
        if $rp$ occluded by $s$ then
          // Ignore this half detection, it’s already covered
          $P_v \leftarrow P_v \setminus rp$
          break
        else if $\overline{rp}$ occluded by $s$ then
          // Detection explained by scene, accept
          $\tilde{S}_v \leftarrow \tilde{S}_v \cup rp$
          $P_v \leftarrow P_v \setminus rp$
          break
      end if
    end for
  end if
end for
end while

2.4 Depth from Stereo

In order to obtain meaningful predictions for a path planning algorithm, it is advisable to use physically sound motion models in 3D world co-
ordinates. To localize an object in an uncontrolled environment, depth from passive stereo is currently the most reliable way in computer vision.

Another vision-only option for 3D localization is the backprojection of the bounding box footpoint and the intersection with a groundplane [Gavrila and Munder, 2007; Leibe et al., 2007a; Havlena et al., 2009]. While this method is also applicable to single cameras, it is usually considerably less accurate. On the one hand, this is due to the fact that the person’s foot often does not coincide with the bounding box’s bottom. On the other hand, the ground plane surface often cannot be approximated by a plane very accurately, especially in the presence of bumps, sidewalks, *etc.* We thus believe that stereo is currently inevitable for robotic or automotive applications when vision is the only sensor to be used.

The downside of using passive stereo estimation is its dependence on sufficiently distinctive texture. Urban scenarios often contain many homogeneous areas and repeated patterns, where typical stereo algorithms often fail to give accurate measurements without assuming stronger models. Similarly, specularities from windows make stereo matching impossible. Last but not least, as for object detection, the algorithm depends on a good input image and will fail in oversaturated or shadowy areas, or in insufficiently lit scenarios. While these problems persist with current stereo matching techniques, we should see improvements in the future due to the tighter coupling with scene understanding and object detection, better optimization strategies, and also the use of high dynamic range cameras.

Active sensors such as lasers are not susceptible to most of the aforementioned problems and yield very accurate distance estimates. However, many current lasers only yield measurements for a few horizontal planes. Also, as the sensor does not integrate incoming light but rather gives measurements for very specific points, it is easy to miss thin objects such as pedestrians (or their legs, at whose height lasers are often mounted) even at moderately short distances: with the common radial resolution of 0.5°, the rays will be roughly 0.17 m apart at a distance of 20 m. In comparison, the radial resolution of our SmartTer vision setup is 0.07° at a FOV of 50°, and would still be 0.28° with a FOV of 180°. Still, sensor fusion is an important option to consider in future work.
2. Capturing Setup and Preliminaries

Figure 2.10: Depth estimation can be reduced to a 1-dimensional search for the correct disparity value (a) if the images are rectified: given a pixel in the left image, its corresponding pixel lies on the same scan line of the image (b). Note the ambiguity when basing decisions on local neighborhoods only: when comparing local pixel intensities between the images, the cost function has numerous minima. The correct solution (red vertical line) is actually only close to the minimum. Smoothness constraints are thus often used to aid in the search for the correct disparity.

2.4.1 Employed Algorithms

The goal of passive stereo is to estimate the depth of each image point, given two or more views of the same scene taken at the same time. As opposed to active stereo, passive stereo does not take advantage of patterns projected by an additional light source; it uses standard images as input. Stereo algorithms usually search for corresponding image points using simple criteria such as similarity of intensity or normalized cross correlation. The search is often subject to a smoothness constraint to help in areas with insufficient texture. Given the correspondences, the depth can be obtained by triangulation. When both images are rectified, i.e. their epipolar lines are parallel to the image rows, corresponding image points will have the same $y$-coordinate in the image. The estimation can then be cast as a 1-dimensional search for the correct disparity value, which refers to the difference in $x$-position of an object seen by two cameras, as shown in Fig. 2.10. The disparity is inversely proportional to the depth. This relation is further investigated in Section 2.4.3. In the following, we will use the symbol $\mathcal{D}$ for the depth map corresponding to an image $\mathcal{I}$. The 3D point corresponding to a given pixel $\mathbf{p}$ is denoted as $\mathcal{D}(\mathbf{p})$, its depth component alone as $\mathcal{D}(\mathbf{p})^{(3)}$. 
Figure 2.11: Stereo depth maps for an example image pair from Seq. Loewenplatz. From left to right: GPU-based, belief-propagation based, global optimization algorithm. Parts that are believed to be inaccurate (by a left-right check) are painted black. More advanced algorithms give visually better results, but take more time and are often not necessary.

Nowadays, a plethora of stereo algorithms is available. A quantitative evaluation on artificial images is conducted in [Scharstein and Szeliski, 2002]. The requirements for an algorithm in our application are both speed and the ability to deal with the large number of untextured areas. To this end, we investigated three different algorithms of varying speed and quality (see Fig. 2.11 for their output when given a typical street scene). In all cases, we use a maximum of 128 disparity levels.

The first algorithm is GPU-based. It is a hierarchical version of [Cornelis and Gool, 2005] that takes about 20 ms per image. The algorithm alternates between the assignment of the best-matching disparity and Gaussian smoothing to cover untextured areas. This iterative structure is fast, but can lead to oscillations and wrong disparity estimates. This method is thus mainly considered for the online application of the system.

In most cases, we will employ a publicly available, belief-propagation based algorithm [Felzenszwalb and Huttenlocher, 2006] that combines a hierarchical approach and smoothness constraints within an Markov Random Field to obtain good disparity estimates, even for untextured regions. Implemented on the CPU, it takes about 20–30 s per image.

Arguably the best results are obtained by a recent global optimization algorithm [Zach et al., 2009]. It can be implemented on the GPU to run at about 500 ms per image for a downscaled (320 × 240) image with 32 possible disparity levels, or at 1 s for 64 disparity levels. For larger images, current GPUs’ memory does not suffice. We hence use the
output as an initialization of the belief-propagation algorithm, yielding smoother results at a total runtime of about 20–30 s per image.

Instead of just remapping the left disparity map to the right image, we usually calculate a second right-to-left one independently with the idea of comparing the two maps with each other. We then label pixels that only got an estimate due to smoothing, as described next.

### 2.4.2 Confidence map

As indicated above, disparity maps will suffer from various artifacts due to deficiencies in the algorithm (smoothing), the inherent formulation of the problem (specularities, untextured areas), and frequent occlusions between both cameras. Usually, algorithms do not directly handle occlusions, but will still give (wrong) results even for insecure areas. To prevent such erroneous information from propagating through the system, we try to infer and label bad pixels according to the following two rules:

- **Appearance.** If the sum of absolute intensity differences (no neighborhood) between two matched pixels exceeds a threshold, the pixel is labeled as occluded. This usually identifies most incorrect and occluded labelings.

- **Disparity.** Especially homogeneous areas are often only labeled due to the smoothing of the disparity estimator. Starting from an anchor point, this smoothing will give different results depending on the source image of the disparity estimator. Hence, pixels that do not have a sufficient data term will have differing disparities in the left and the right image and can thus also be marked. This step usually helps greatly in identifying incorrect labels for untextured regions.

This binary labeling will be captured in the confidence map $C$, with $C(p) = 1$ indicating a confident pixel $p$, and $C(p) = 0$ the converse. Pixels identified with invalid disparities are shown in black in the images of Fig. 2.11. As can be seen, the straightforward smoothing of the GPU-based estimator results in far more invalid pixels. These pixels will be ignored in calculations in the following steps.
2.4. Depth from Stereo

Figure 2.12: Accuracy of depth estimation is dependent on the chosen focal length and the baseline. (a) Relation of disparity to depth for both setup types. (b) Localization accuracy at given depths. Higher values correspond to more inaccurate measurements. SmartTer’s larger baseline allows accurate localization at larger depths.

2.4.3 Accuracy

Given the focal length $f_u$ in pixels, the camera baseline $B$ in metric units and the disparity $d$ from a stereo estimator, the depth $z$ of an image point can be calculated as

$$ z = \frac{f_u B}{d} . \quad (2.20) $$

Thus, the working range and respective accuracy of the system is mostly determined by the focal length and the baseline of the employed cameras. Fig. 2.12 illustrates the relationship between disparity and depth for the CharioBot and SmartTer setups, as well as the obtained accuracy in depth. Defining the working range to reach from the minimum distance to the point where the localization error exceeds 1 m, this yields 1.5 to 15 m for the CharioBot platforms. The pedestrian detector theoretically can go up to 19 m, giving still a localization accuracy of 1.5 m. For SmartTer, the working range is 3.8 to 22 m, with the detector being able to find pedestrians up to 30 m away, where the localization accuracy is about 1.8 m. Note that doubling the image resolution would result in twice the maximal distances, as the focal length in pixels will also double.
Due to the nonlinear relationship between disparity and depth, it is important to properly account for localization errors in our algorithms. Using error propagation, the localization uncertainty of the stereo system can be inferred from the uncertainties \( (\sigma_u, \sigma_v) \) of a pixel with position \((u, v)\) (principal point \((p_u, p_v)\) subtracted), and the uncertainty \(\sigma_d\) of the disparity estimate \(d\). Assuming small errors in pixel position measurements, we empirically set \(\sigma_u = \sigma_v = \sigma_d = 0.5\).

We can write the backprojection as

\[
f(u, v, d) = \begin{pmatrix} u \\ v \\ f_u \end{pmatrix} \frac{B}{d}. \tag{2.21}
\]

Using forward error propagation and \(s_{vu} \approx 1\), this yields the uncertainty covariance of a reconstructed 3D point as

\[
C = \left( \frac{\partial f}{\partial u} \right)^\top \begin{pmatrix} \sigma_u & 0 & 0 \\ 0 & \sigma_v & 0 \\ 0 & 0 & \sigma_d \end{pmatrix} \left( \frac{\partial f}{\partial u} \right) \tag{2.22}
\]

\[
= \begin{pmatrix} \sigma_u + \sigma_db^2u & \sigma_db^2uv & \sigma_db^2u \\ \sigma_db^2uv & \sigma_v + \sigma_db^2v & \sigma_db^2v \\ \sigma_db^2u & \sigma_db^2v & \sigma_d fb^2 \end{pmatrix}, \tag{2.23}
\]

with \(b = \frac{B}{fd}\). Thus, uncertainty in depth grows inversely proportional with the quadratic disparity and hence quadratically with depth. Increasing baseline or image resolution linearly will also result in a linear decrease in uncertainty.

Alternatively, if disparity is not directly available (e.g., in non-rectified images), the uncertainty of a 3D point can be calculated given the camera matrices using error backpropagation. Assuming additive white noise with covariance \(C_{2D} = [\sigma_u \ 0 \ ; \ 0 \ \sigma_v]\) on pixel measurements,

\[
C = (F^{(1)})^\top C_{2D}^{-1} F^{(1)} + (F^{(2)})^\top C_{2D}^{-1} F^{(2)} - 1, \tag{2.24}
\]

where \(F^{(j)}\) are the Jacobians of a full projection using camera matrix \(j\), i.e., including the world-to-camera transformation (Eq. (2.29)) and the stereo rig’s extrinsic calibration. In our experiments, we used this second option.
As we will show in this thesis, depth is a useful complementary cue for many algorithms and almost indispensable for accurate localization of objects when no other sensors are available. When using depth, however, one should account for its deficiencies, i.e., insecure measurements in areas of specularities, occlusion, or missing texture, as well as its quadratically growing uncertainty in localization. Therefore, throughout the rest of this work, we will use the measures given in this section when dealing with depth information.

2.4.4 Occupancy Maps

For static obstacles, we construct a stochastic occupancy map based on an algorithm by [Badino et al., 2007]. The occupancy map is modeled as a polar grid on the ground, which provides a constant depth resolution and has a direct correspondence with the camera’s viewing rays. Thus, one point in the depth map will affect at most one column in the polar occupancy map. A 3D point $X_{\text{cam}}$ in camera coordinates is thus transformed as

$$\theta = \arctan \frac{X_{\text{cam}}^{(3)}}{X_{\text{cam}}^{(1)}}, \quad (2.25)$$

$$r = \sqrt{(X_{\text{cam}}^{(1)})^2 + (X_{\text{cam}}^{(3)})^2}. \quad (2.26)$$

An input depth map is first pruned to a corridor of interest in vertical ($y$-) direction, consistent with the vehicle’s height. Then, every remaining 3D point is transformed and discretized onto the grid, where we account for the uncertainty in depth by distributing the point’s votes along the viewing ray corresponding to Eq. (2.24). To arrive at the actual occupancy measure, we compare the number of votes in cell $L_{ij}$ with the maximally possible number of votes. Disregarding the vertical filtering, the number of votes per row corresponding to a given angle $\theta$ is constant. Also, as we distribute votes over multiple radial bins to account for uncertainty in depth, the radial distribution can be regarded as approximately uniform. Thus, a normalization by a constant factor $s$ (dependent on the radial resolution) yields the occupancy

$$O_{ij} = L_{ij} s. \quad (2.27)$$
2. Capturing Setup and Preliminaries

Figure 2.13: Occupancy maps can be calculated based on depth maps. Given an image (a,c), these capture the location of obstacles (b,d) in a polar grid. The intensity corresponds to the probability of occupancy.

As occlusions or missing estimates happen frequently, it is useful to integrate obstacle information over time. For this, the previous frame’s map, taken at camera position $P^{(t-1)} = (R^{(t-1)}, t^{(t-1)})$ is transformed into the new frame’s camera coordinate system, where the camera position is $P^{(t)} = (R^{(t)}, t^{(t)})$:

$$X^{(t)}_{\text{cam}} = R^{(t)}(R^{(t-1)})^\top \left( X^{(t-1)}_{\text{cam}} - t^{(t-1)} \right) + t^{(t)} .$$ (2.28)

The old and the new map are then integrated using a weighted sum. We set the mixing coefficient to 0.5, a Kalman-filtering-based approach could potentially give better results. Two obtained sample maps are shown in Fig. 2.13. In both cases, the obstacles are reflected rather accurately in the occupancy map. Based on such maps, free space for driving can be computed using dynamic programming [Badino et al., 2007].

One major drawback of these maps is that they do not account for dynamic objects: firstly, it would be advantageous to know which parts of the map are moving objects and cast corresponding predictions for path planning, c.f. Fig. 2.13 (c,d). Secondly, the temporal integration will exhibit smearing artifacts in case of non-static objects. These two points will be addressed in Chapter 5.

2.5 Visual Odometry

To allow reasoning about object trajectories in the world coordinate system, the camera position and rotation $P = (R, t)$ for each frame are
estimated using visual odometry. In this thesis, we will usually assume transformations from world to camera coordinates,

\[ \mathbf{X}_{\text{cam}} = \mathbf{R} \mathbf{X}_{\text{world}} + \mathbf{t} \]  

(2.29)

Visual odometry is mostly used as a necessary means in our system and will thus only be discussed briefly. After reviewing related work, we will give an outline of the employed algorithm, along with a few example camera trajectories.

2.5.1 Related Work

The majority of the work in visual odometry is based on local features and RANSAC-type hypothesize-and-test frameworks [Cornelis et al., 2008; Nistér et al., 2004; Se et al., 2002]. Some other approaches include Hough-like methods [Makadia et al., 2005], methods based on points and lines [Ess et al., 2007b], or recursive filtering [Davison, 2003; Eade and Drummond, 2006]. Most of latter have however not been demonstrated on extended runs in realistic outdoor scenarios. The main problem with all these methods is the assumption that a dominant part of the scene changes only due to camera egomotion. As a result, these approaches are prone to failure in crowded scenes with many independently moving objects. While there has been work on multi-body Structure-from-Motion [Li et al., 2007; Ozden et al., 2007], most systems are still constrained to short videos, and more importantly, assume sufficiently large, rigidly moving objects. In robotics, various approaches for SLAM in dynamic environments exist [Bibby and Reid, 2007; Hähnel et al., 2003; Wang et al., 2007], related to the above, but mostly focusing on range data. In this section, we will first describe the basic system used in this thesis, which builds on and extends previous work by [Nistér et al., 2004]. In Chapter 5, we propose to explicitly feed back information from object tracking to egomotion estimation, thereby introducing semantics.

2.5.2 Algorithm

An overview of the employed visual odometry algorithm is given in Fig. 2.14 (a). In short, each incoming image is divided into a grid of
2. Capturing Setup and Preliminaries

Figure 2.14: (a) Overview of the employed visual odometry system. (b) Feature binning ensures an even feature distribution suitable for stable localization. (c) Runtimes for $2 \times 300$ keypoints and $5,000$ tested hypotheses.

10 $\times$ 10 bins and an approximately uniform number of feature points is detected in each bin using a Harris corner detector [Harris and Stephens, 1988] with locally adaptive thresholds. The binning encourages a feature distribution suitable for stable localization, Fig. 2.14 (b). In the initial frame, stereo matching and triangulation provide a first estimate of the 3D structure. In subsequent frames, we use 3D-2D matching to get correspondences, followed by camera resection (3-point pose) with RANSAC [Nistér, 2004]. The camera position is optimized given all inliers from RANSAC using Levenberg-Marquardt, which also yields an uncertainty estimate for the final solution. Bundle adjustment is run on a sliding window of $n_b = 18$ past frames to further polish the raw camera and point estimates. Older frames are discarded, along with points that are only supported by these removed frames.

Important details for reliable performance are the use of 3D-2D matching to bridge temporally short occlusions of feature points and to filter out independently moving objects at an early stage, as well as a Kalman filter to predict the next camera position for feature detection (leading to a feature detection strategy similar to the “active search” paradigm in SLAM, e.g., [Davison, 2003]). Scene points are directly associated with a viewpoint-invariant SURF descriptor [Bay et al., 2008] that is adapted over time. In each frame, the 3D-2D correspondence search is then constrained by the predicted camera position. As mentioned above, only scene points without support in the past $n_b$ frames are discarded. This allows one to bridge temporally short occlusions (e.g., from a person passing through the image) by re-detecting 3D points that carry...
2.5. Visual Odometry

![Camera trajectories for Seq. Loewenplatz and Seq. Bellevue, obtained with the stand-alone visual odometry system developed in this thesis. Red: with bundle adjustment and double precision (CPU). Blue: without bundle adjustment and single precision (GPU).](image)

Figure 2.15: Camera trajectories for Seq. Loewenplatz and Seq. Bellevue, obtained with the stand-alone visual odometry system developed in this thesis. Red: with bundle adjustment and double precision (CPU). Blue: without bundle adjustment and single precision (GPU).

information from multiple viewpoints and are therefore already reliably reconstructed.

The system is implemented largely on the graphics card, taking advantage of both GPU SURF [Cornelis and Gool, 2008] for feature description and the parallel nature of RANSAC to simultaneously generate and test multiple pose hypotheses. While some researchers [Engels et al., 2006] have shown bundle adjustment to be feasible in real-time, we employed a slower, publicly available package [Lourakis and Argyros, 2004]. As the construction of motion models used for tracking allows for a slight drift, we omit bundle adjustment in our application. Overall, attractive runtimes can be obtained for the visual odometry part, Fig. 2.14 (c).

In static and moderately dynamic scenes, this system manages to yield accurate positioning over long distances, with low drift that mostly depends on the actual camera calibration and the single-precision accuracy of the GPU. A few sample camera trajectories obtained on sequences recorded with the SmartTer platform are shown in Fig. 2.15. Both the more accurate CPU version with bundle adjustment as well as the faster GPU version are shown. While a small drift is noticable, especially for the GPU version, the estimates are generally smooth and rather close to the actual vehicle’s path on the map. In fact, driftless global localiza-
tion using only a moving camera rig is inherently impossible (except in retrospect in the case of loop closure). We believe that this capability, if needed, is best achieved by integrating other sensors, such as GPS or inertial measurement units, as also argued, e.g., in [Zhu et al., 2006b].

As we will show later in this thesis, an independently operating system is prone to failure in highly dynamic scenarios, as RANSAC is missing the appropriate semantic information that can help in its search for valid hypothesis. In Chapter 5, we will show that only a tight coupling of the vision components can help in such situations.
Detecting pedestrians reliably from a moving platform is a fundamental asset for obstacle avoidance and path planning, with numerous applications in autonomous driving and mobile robotics. The focus of our work are video streams as recorded from a pair of forward-looking, street-level cameras. The detection task in this scenario is extremely challenging due to a variety of factors. Firstly, images from unconstrained video streams exhibit a much lower quality than their photographed counterparts due to motion blur, debayering artifacts, and varying lighting conditions. Secondly, the large number of independently moving objects, covering sometimes up to 50% of the image, leads to frequent partial occlusions between pedestrians, which is problematic for standard object detection and tracking techniques. Thirdly, even state-of-the-art pedestrian detectors are challenged by the large range of scales, the multitude of viewpoints, and the ambiguity of side vs. semi-frontal views of pedestrians. Finally, the suboptimal camera placement dictated by constraints on the platforms (approx. 0.9–1.3 m above ground) has adverse effects on the accuracy of distance measurements. Building a reliable system in such highly dynamic scenes thus calls for a tight interaction between multiple cues.

In this chapter, we will focus on the robust detection of pedestrians in a single video frame. Using input from pedestrian detection and dense stereo (Chapter 2), we want to jointly estimate scene geometry and object locations to obtain the mutually best explanation for the given image in 3D. For this, we will choose a simplified model of the 3D scene focusing on the parts that are most important for a later tracking module. Specifically, we want to jointly reason about the valid object hypotheses, find the ground plane they are standing on, and know for which objects
we can trust the corresponding depth map information. As an output, a more refined set of detections with less false positives is obtained, where the detections can be localized in 3D with the help of ground-plane information. As introduced in later chapters, this then enables more robust object tracking with physically plausible motion models.

By modeling the problem in a Bayesian network, inference can be conducted in all directions, that is, the deficiencies of one cue can be made up for by another one. A typical case for this is ground-plane estimation: if a large part of the image is covered by the ground plane, it can be robustly estimated, and can thus be used to constrain object locations. If there are however too many objects, usual methods for ground plane estimation will not work, but the objects themselves can actually be used to constrain the ground plane. By using a Bayesian network as presented in this chapter, both cases can be resolved elegantly within one single model.

The main contributions found in this chapter are: 1) A method to simultaneously estimate scene geometry and detect objects in challenging real-world scenarios (from video input), in particular integrating cues from dense stereo, object detection, and ground-plane estimation. 2) The principled modeling of this integration using a Bayesian network that allows depth measurements to benefit from object detection and vice versa. 3) The validation of the proposed approach on challenging real-world data, which in later chapters shows to be an ideal basis for tracking. 4) For the ISM detector, an iterative method is proposed that can deal with implicit loops in the Bayesian network stemming from overlapping detections.

The chapter is structured as follows. In Section 3.1, we discuss related work for pedestrian detection and integration of context. Then, Section 3.2 gives an overview of the problem formulation. Section 3.3 introduces the graphical model used for improving object detection based on cues from depth and ground plane. Training and inference are discussed in Section 3.4 and Section 3.5, respectively. Following that, systematic and experimental results are presented in Section 3.6. Section 3.7 discusses the steps used to prepare detections for tracking, before the chapter is concluded in Section 3.8.
3.1 Related Work

In recent years, object detection has reached a level where it becomes interesting for practical applications, e.g., for detecting pedestrians in real-world scenes [Dalal and Triggs, 2005; Felzenszwalb et al., 2008; Leibe et al., 2005; Mikolajczyk et al., 2006; Viola and Jones, 2004; Wojek et al., 2009; Wu and Nevatia, 2005; Zhu et al., 2006a]. Still, pedestrian detection remains a very difficult task due to the large degree of intra-category variability, changing scale, articulation, and frequent partial occlusion (see also the rather low recall achieved in recent performance evaluations [Dollar et al., 2009; Everingham et al., 2008]). To achieve robustness to adverse imaging conditions, the importance of context has been widely recognized. Depending on the authors, the rather loose notion of “context” can refer to different types of complementary information.

Scene geometry. A popular method for improving object detection is to model the geometric properties of the scene where the objects are supposed to reside. In the easiest case, this corresponds to calibrating a ground plane and pruning detections accordingly (e.g., [Gavrila and Munder, 2007; Okuma et al., 2004; Huang et al., 2008]). More sophisticated algorithms try to automatically estimate the ground plane from a single image or video streams [Breitenstein et al., 2008; Ess et al., 2007a; Hoiem et al., 2006; Leibe et al., 2007a]. Going beyond geometrical modeling, some researchers use semantics of other image regions, including the entire scene, to influence object detection [Sudderth et al., 2005; Torralba, 2003; Murphy et al., 2003; Ommer and Buhmann, 2005; Li et al., 2009].

In two recent publications, [Hoiem et al., 2006; 2008] showed how geometric context can be inferred from a single image in conjunction with object detection. We build upon these ideas and extend them for our scenario, with a considerable scale range in detections, frequent partial occlusions, and integrating stereo depth.

In another related system, [Leibe et al., 2007a] detect objects from a moving vehicle, integrating detection and Structure-from-Motion. In their imagery, objects typically appear in a well-contained scale range.
They fit a ground plane through past wheel contact points and then fix it for the detection stages. In our experiments, such a fitting required a temporal look-ahead. In contrast, our framework integrates multiple cues to explain the scene causally, i.e., using only information from the current and previous frames.

**Temporal continuity.** Several researchers have investigated temporal context in the form of using tracking or spatial proximity to drive object detection in video imagery [Ess et al., 2009b; Leibe et al., 2008b; Li and Nevatia, 2008; Wu and Nevatia, 2007a]. Others have examined the temporal dynamics of the human gait cycle [Andriluka et al., 2008]. A more local cue that can be directly integrated into the feature pool of a detector is motion from optic flow [Dalal et al., 2006; Viola et al., 2003; Wojek et al., 2009].

In this chapter, we will largely ignore temporal context. As our system derives temporal information from tracking, we will postpone this discussion to Chapter 5. Note however that local temporal information could, e.g., be included at this point by using a motion-based object detector [Wojek et al., 2009].

**Segmentation.** The entanglement of object detection and segmentation has been explored as well, often by means of Conditional Random Fields (CRFs) [Lafferty et al., 2001]. These can be thought of as the discriminative counterpart to Markov Random Fields (MRFs). However, many authors ignore the explicit notion of objects, such that several approaches have trouble with the limited neighborhood of MRFs as well as the unknown object scale [Carbonetto et al., 2004; He et al., 2004; Shotton et al., 2006]. Some researchers employ separate object detectors and model their interplay with the scene segmentation, either in a separate layer [Hoiem et al., 2006; Kumar and Hebert, 2005] or jointly [Wojek and Schiele, 2008].

Focusing on car detections, [Wojek and Schiele, 2008] couple recent work in texture classification with object detections in a dynamic CRF. While such a system can give a class label to every patch in an image while at the same time reasoning about object detections, the intricate modeling takes more computation time, cannot reason about the ground plane,
and in our experiments for pedestrians (Chapter 7) also has problems due to the coarse resolution of the patch grid used.

In this chapter, we will focus on a simplified model of the scene accounting for the objects and the ground plane they reside on.

**Stereo depth.** The use of depth cues for improving detections suggests itself in systems equipped with camera pairs. The most notable recent systems taking advantage of depth include the ones by [Giebel et al., 2004] and [Gavrila and Munder, 2007]. However, their pedestrian tracking systems work with the assumption of a fixed groundplane, interactions between pedestrians are not modeled, and no results for busy scenes are shown. In the test sequence of [Gavrila and Munder, 2007], only 1,000 out of 20,000 test images contain pedestrians—often, only one. Here, Seq. BAHNHOFFSTRASSE alone contains 5,500 annotated pedestrians in 1,000 frames. Another approach, using no appearance but stereo data for human detection and tracking only was presented by [Bajracharya et al., 2009]. We present a comparison to their system in Chapter 4.

### 3.2 Problem Formulation

Given as inputs pedestrian detections in a single video frame $I$ and the corresponding depth and confidence map $D, C$, we are interested in simultaneously inferring the ground plane and the set of valid pedestrian hypotheses. The hypotheses $o_i = \{c_i, v_i\}$ are obtained from a standard pedestrian detector and are decomposed into an object center $c_i$, as well as a validity flag $v_i$. Geometric reasoning is conducted over the scene’s ground plane $\pi$ and the object bounding boxes, where the latter are adapted automatically by our algorithm for better scene explanation. Depth cues $d_i$ are introduced in a robust way, such that the system can cope with faulty depth maps. These components and their interactions are formalized in a Bayesian network, which is generated on a per-frame basis.

In short, the probabilistic modeling combines the following measurements: for each object, a backprojection of the bounding box onto the
ground plane yields a real-world distance and height. The former measurement can be compared with the median stereo depth measured inside the bounding box, where we expect the two values to be equal within the given uncertainty bounds. The measured real-world height can be validated under a population density (stating, \textit{e.g.}, that pedestrians usually are around 1.7 m high). The quality of a detection per se is further influenced by the detector score as well as the flatness of its depth profile: within the accuracy of the depth map, we assume objects to have a largely flat profile. All objects are furthermore supposed to reside on a common ground plane, whose position is inferred based on prior knowledge and depth information, as well as indirectly from the object detections. The proposed Bayesian network allows joint inference about the most probable scene explanation given this model.

For the ISM detector, we also propose a method to handle interactions between objects in a computationally feasible way, by splitting the reasoning into two stages. We first obtain an initial estimation of the scene geometry, disregarding overlapping hypotheses. Next, the obtained MAP estimates are passed to a global optimization stage that handles interactions on a pixel level.

In the following, the discussion focuses on a single image. However, multiple images, along with their respective cues, can be added in a straightforward manner. In our integrated setup, we conduct the reasoning jointly over both images of the camera rig. The obtained results are then used as a basis for tracking-by-detection, which will be explained in Chapter 4.

### 3.3 Graphical Model

Fig. 3.1 shows the Bayesian network we use for per-frame inference over object hypotheses \( o_i = \{c_i, v_i\} \), object depth \( d_i \), and the ground plane \( \pi \). Inference in this model is performed as follows:

\[
P(\pi, c_i, v_i, d_i, \mathcal{E}) \propto P(\pi)P(\pi_D|\pi)Q \tag{3.1}
\]

\[
Q = \prod_i P(c_i|\pi, d_i)P(v_i|c_i, \pi)P(v_i|d_i)P(d_i|C)P(\mathcal{I}|v_i)
\]
### Figure 3.1: The Bayesian network for single-frame detection with additional information from depth maps.

<table>
<thead>
<tr>
<th>Var.</th>
<th>Meaning</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{I}$</td>
<td>Images of camera pair</td>
<td>Observed</td>
</tr>
<tr>
<td>$\mathcal{C}$</td>
<td>Confidence maps</td>
<td></td>
</tr>
<tr>
<td>$\pi_D$</td>
<td>$\pi$ cue inferred from $D$</td>
<td></td>
</tr>
<tr>
<td>$c_i$</td>
<td>Object center point and scale</td>
<td>${{k, l}</td>
</tr>
<tr>
<td>$v_i$</td>
<td>Object validity</td>
<td>${0, 1}$</td>
</tr>
<tr>
<td>$d_i$</td>
<td>Validity of depth per object</td>
<td>${0, 1}$</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Ground plane</td>
<td>${{\phi, \theta, \pi^{(4)}}</td>
</tr>
</tbody>
</table>

**Table 3.1: Variables of the model, along with their domains.**

where $\mathcal{E} = \{\mathcal{I}, \mathcal{C}, \pi_D\}$ is the evidence observed in the current frame. $\mathcal{I}$ is the image information, $\mathcal{C}$ the confidence map from depth generation (Chapter 2), and $\pi_D$ ground plane evidence inferred from the depth map. The actual depth map $D$ is used implicitly when modeling the other variables’ probabilities. An object $o_i = \{c_i, v_i\}$ is decomposed into its center, whose probability $P(c_i|\pi, d_i)$ depends on the geometric consistency of depth map and ground plane localization; and a validity flag that encodes both its geometric world features (distance, size) $P(v_i|c_i, \pi)$ and its correspondence with the depth map (assumption of uniform depth) $P(v_i|d_i)$. $P(\mathcal{I}|v_i)$ is the object probability estimated by the employed detector. $P(d_i|\mathcal{C})$ encodes the reliability of the depth map. Following standard graphical model notation [Bishop, 2006], the plate indicates repetition of the contained parts for the number of objects $n$. The variables, along with their domains, are summarized in Tab. 3.1.
In the following, the components of this Bayesian network are described in detail. All 3D calculations are executed in camera coordinates, i.e. the camera position is $P = (I, 0)$. This not only simplifies calculations and parameterizations, but it also keeps the set of possible ground planes in a range that can be trained in a meaningful way. For the subsequent tracking stage, the results are easily transferred into world coordinates, using the camera orientation provided by visual odometry (Section 3.7).

### 3.3.1 Ground Plane

As shown in previous publications [Hoiem et al., 2006; Leibe et al., 2007a; Gavrila and Munder, 2007], the ground plane helps substantially in constraining object detection to meaningful locations. It is defined in the current camera frame as $\pi = (n, \pi^{(4)})$, where the normal vector is parameterized by spherical coordinates, $n(\theta, \phi) = (\cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi)$. To compensate for changes in the scene geometry (hills, etc.) or platform tilt due to the suspension, the ground plane is not fixed a priori. Rather, an entire set of possibilities is considered for every new video frame.

The ground plane parameters $\pi$ are inferred from a combination of a prior from the previous frame, object bounding boxes, and the depth map evidence $\pi_D$, so that the system does not critically depend on any one individual cue. While accurate ground planes can be estimated...
directly from clean depth maps (see below), such methods break down in outlier-ridden scenarios. Thus, $\pi_D$ will just act as an additional cue in our Bayesian Network. Specifically, we consider the depth-weighted median residual between $\pi$ and $D$, averaged over three horizontal stripes $S_i$ to account for unequal sampling:

$$r_i(\pi, D)^2 = \text{med}_{\{p \in S_i | C(p) = 1\}} \frac{1}{\sigma_r} (n^T D(p) - \pi^{(4)})^2,$$

(3.2)

$$r(\pi, D)^2 = \left( \sum_{i=1}^{3} r_i(\pi, D_i)^2 \right) / 3.$$

(3.3)

Here $p \in S_i$ denotes the pixels from a vertical stripe of $D$, deemed valid by the confidence map ($C(p) = 1$). $\sigma_r$ is a constant encoding the uncertainty of distance measurements. To account for the increasing number of points at short distances, the height $h_y(i)$ of the stripes $S_i$ gets bigger the closer we get to the lower image border (we use the progression $h_y(i) = \frac{h}{2(i+1)} = \{120, 80, 40\}$, with $h$ the total image height; Fig. 3.2). The sets are further pruned according to the vehicle’s maximally expected tilt angle and restricted to the lower part of the image for increased robustness to outliers.\(^1\) In Eq. (3.2), $\Sigma_D$ accounts for the 3D point’s uncertainty in the plane-to-point measurement. Given this robust estimate, we set\(^2\)

$$P(\pi_D | \pi) \propto e^{-r(\pi, D)^2}.$$

(3.4)

The prior $P(\pi)$ is also learned from a training set, as described in Section 3.5.

### 3.3.2 Object Hypotheses

Object hypotheses $o_i = \{v_i, c_i\}$, $(i = 1 \ldots n)$ are created from the output of a pedestrian detector for each frame. To obtain maximum recall, the threshold is set low, yielding typically 10–100 detection hypotheses at each time step. These consist of a validity flag $v_i \in \{0, 1\}$ and a 2D center point with scale $c_i = \{x, y, s\}$. Given a specific $c$ and a standard object

\(^1\)In the future, the pruning could be done more effectively based on a texture classification, as introduced in Chapter 7.

\(^2\)Note that since $r(\pi, D)$ is obtained as a median, $P(\pi_D | \pi)$ should more appropriately be modeled as a Laplacian density. This made no difference in our experiments.
size \((w,h)\) at scale \(s = 1\), a bounding box can be constructed. From the box base point in homogeneous image coordinates \(g = (x, y + sh/2, 1)\), its counterpart in world coordinates is found by backprojecting a ray and intersecting it with the ground plane, yielding the 3D point

\[
G = -\frac{\pi(4)K^{-1}g}{n^tK^{-1}g}.
\]  

(3.5)

\(K\) denotes the camera’s internal calibration matrix (Section 2.2.1). The object’s depth is thus \(z(o_i) = \|G_i\|\). The box height \(G^h_i\) is obtained in a similar fashion, by intersecting another ray through the bounding box’s top point with a fronto-parallel plane, orthogonal to the ground.

Because of the large localization uncertainty of appearance-based detection, the detector outputs for center and scale are only considered as estimates, denoted \(\tilde{x}_i, \tilde{y}_i, \) and \(\tilde{s}_i\). Taking these directly may yield misaligned bounding boxes, which can in turn result in wrong estimates for distance and size. We therefore try to compensate for detection inaccuracies by considering a set of possible bounding boxes \(b^{\{k,\ell\}}_i\) for each \(o_i\). These boxes are constructed from a set of possible real centers \(c_i = \{y_i, s_i\}\) (fixing \(x_i = \tilde{x}_i\) due to its negligible influence), which are obtained by sampling around the detection, \(y_i = \tilde{y}_i + k\sigma_y\tilde{s}_i, s_i = \tilde{s}_i + \ell\sigma_s\tilde{s}_i\). \(\sigma_y\) and \(\sigma_s\) are step sizes inferred from a training set, modeling the typical inaccuracies of the bounding box localization. The number of samples, \(i.e.\) the range of \(\{k, \ell\}\), is the same for every object. An object hypothesis thus yields a discrete set of allowed changes in position and scale, \(e.g.,\) allowing for \(3 \times 3\) different choices, each one corresponding to a label of \(c_i\), and hence a real-world object of differing height and distance. In the following, we omit the superscripts for readability.

By means of Eq. (3.5), \(P(v_i = 1|c_i, \pi) \propto P(G^h_i)P(z(o_i))\) is expressed as the product of a distance prior \(P(z(o_i))\), encoding plausible object distances; and of a size prior \(P(G^h_i)\) for the corresponding real-world object.

The detection reliability \(P(I|v_i)\) is learned from a training set of correct and incorrect detections with corresponding scores using logistic regression.
3.3.3 Depth Map

The depth map $D$ is a valuable asset for scene understanding that is readily available in a multi-camera system. However, stereo algorithms frequently fail, especially in untextured regions. Using the confidence map, we integrate depth into our framework in a robust manner: each object hypothesis is augmented with a depth flag $d_i \in \{0, 1\}$, indicating whether the depth map for its bounding box is reliable ($d_i = 1$) or not. This flag’s evidence is inferred from the confidence map $C$ and is encoded in $P(d_i|C)$.

First, we evaluate the stereo depth measured inside $b_i$ and its consistency with the ground plane depth $z(o_i)$ as an indicator for $P(c_i|\pi, d_i = 1)$. Second, we test the depth variation inside the box and define $P(v_i = 1|d_i = 1)$ to reflect our expectation that the depth is largely uniform when a pedestrian is present. The measurements are defined as follows: the median depth inside a bounding box $b_i$,

$$z(D, b_i) = \operatorname{med}_{\text{pixel } p \in b_i} D(p)^{(3)},$$

yields a robust estimate of the corresponding object’s depth. Based on Eq. (2.24), we infer the measurement uncertainty as $\sigma^2_{(z),i} = C_i^{(3,3)}$. This yields

$$P(z),i(a) \propto N(a; z(D, b_i), \sigma^2_{(z),i}).$$

(3.7)

$P(z),i(a)$ thus models the probability that a given distance measurement $a$ corresponds to the robustly estimated depth of the bounding box. As described later, it can be used to model $P(c_i|\pi, d_i = 1) = ZP(z),i(z(o_i))$, with $Z$ a normalization factor. $P(c_i|\pi, d_i = 0)$ is assumed uniform.

For reasoning about depth uniformity, we consider the depth variation for all pixels $p$ within $b_i$, $V = \{D(p)^{(3)} - z(D, b_i)|p \in b_i\}$. To be robust against outliers, the estimate is restricted to the interquartile range $[LQ(V), UQ(V)]$, and depth uniformity is measured by the normalized count of pixels that fall within the confidence interval $\pm \sigma_{(z),i}$,

$$q_i = \frac{|\{x \in [LQ, UQ]|x^2 < \sigma^2_{(z),i}\}|}{UQ - LQ}.$$  

(3.8)

This robust “depth inlier fraction” serves as basis for learning $P(v_i|d_i = 1)$, as will be described in Section 3.5. The probability $P(v_i|d_i = 0)$ is
assumed uniform, since an inaccurate depth map gives no information about the object’s presence. We learn $P(d_i|C)$ from a training set based on the data from the confidence map.

### 3.4 Training

The system’s parameters have been trained on a sequence with 490 frames, containing 1,578 annotations. Tab. 3.1 summarizes the variables of the model and their respective domains. For learning the ground plane prior, we considered an additional 1,600 frames from a few selected environments with hardly any moving objects.

#### 3.4.1 Ground Plane

In input images with few objects, $D$ can be used to infer the ground plane using Least-Median-of-Squares (LMedS) by means of Eq. (3.3),

$$
\pi = \min_{\pi_i} r(\pi_i, D).
$$

(3.9)

Related but less general methods include, e.g., the $v$-disparity analysis [Labayrade et al., 2002]. All such methods break down if less than 50% of the pixels in $D$ support $\pi$. For training, we use the estimate from Eq. (3.9), with bad estimates discarded manually.

For reasons of tractability, the ground plane parameters $(\theta, \phi, \pi(4))$ are discretized into a $6 \times 6 \times 20$ grid, with bounds inferred from the training sequences. The discretization is chosen such that quantization errors are below 0.05 for $\theta$ and 0.01 for $\phi$, resulting in component-wise aberrations of maximally $5 \cdot 10^{-7}$ from the original $n$. In our tests, the errors ensuing from the discretization of $\pi$ were below 0.2 meters in depth for a pedestrian 15 meters away. Note that other choices of spherical coordinates for the normal vector would be better suited to the dominant variability of the tilt angle. However, the described parametrization is sufficient, and alternative choices for discretization turn out to be more cumbersome because of switches from $-180^\circ$ to $180^\circ$. The training sequences also serve to construct the prior distribution $P(\pi)$. Fig. 3.3 visualizes $P(\pi)$ for the platform CharioBot in two projections onto $\pi(4)$ and $(\theta, \phi)$,
3.4 Training

Figure 3.3: Learned priors for (a) the ground plane normal $(\theta, \phi)$ and (b) its distance to the origin $\pi^{(4)}$; projected onto $\pi^{(4)}$ and $(\theta, \phi)$, respectively. (Values are for platform CharioBot)

exhibiting a clear peak at the platform’s stationary position, i.e. flat road and no tilt due to the suspension.

3.4.2 Object Hypotheses

Object detections can be generated with any state-of-the-art pedestrian detector, parametrized in a conservative way so as to avoid false negatives as much as possible. In our experiments, we will first focus on the ISM detector [Leibe et al., 2005], before investigating the other methods (HOG [Dalal and Triggs, 2005] and part-based [Felzenszwalb et al., 2008], c.f. Chapter 2).

As the original detected locations $\tilde{x}, \tilde{y}, \tilde{s}$, and hence the bounding boxes, may not always be sufficiently accurate for reliable distance estimation, we model the offset between real and detected object centers by Gaussians. For this, we collect detections over the training sequence and compare them to ground-truth annotations. Fig. 3.4 shows the resulting scale-normalized measurements $(\tilde{y} - y) / \tilde{s}, (\tilde{s} - s) / \tilde{s}$ used to learn $(\sigma_y, \sigma_s)$. As can be seen from the figure, the Gaussian approximation is justified.

For ISM, we obtained $\sigma_y = 7.81$ and $\sigma_s = 0.099$.

The object size distribution is a population density as chosen in [Hoiem et al., 2006], $P(G^h) \sim \mathcal{N}(1.7, 0.085^2)$ [m], though we consider different
standard deviations $\sigma_h$ in a first systematic experiment in Section 3.6. This is mainly to account for children and for the remaining discretization errors due to the sampling of $c_i$. The distance distribution $P(z(o_i))$ is assumed uniform in the system’s operating range (2–30 m for CharioBot and CharioBot Mk. II; 3–50 m for SmartTer).

3.4.3 Depth Cues

In the experiments presented in this chapter, we will us the algorithm of [Felzenszwalb and Huttenlocher, 2006] for obtaining the depth map $D$ for each frame. See Fig. 3.5 for two example depth maps. The true distribution of $P(c_i|\pi, d_i = 1)$ given the object’s depth $z(o_i)$ and the depth map estimate $z(D, b_i)$ is very intricate to find. It involves many factors: first, the uncertainty of the object’s center propagated to its distance. Due to the sampling of $c_i$, we can neglect this factor. Second, it depends on $P_{(z),i}$ as defined in Eq. (3.7). Finally, using a fixed set of disparities introduces a quantization error, which is only to some extent covered by $P_{(z),i}$.

In Section 3.6, we therefore compare two ways for modeling $P(c_i|\pi, d_i = 1)$. The first option uses a non-parametric distribution $P(v_i|z(o_i) - z(D, b_i))$, learned from the training sequence. The second option models it using the dominant factor $P_{(z),i}(z(o_i))$ only.
3.5. Inference

Figure 3.5: Example depth maps. Most of the time, useful cues can be inferred (a), but robust measures have to account for faulty depth maps, e.g., missing ground plane (b).

For learning \( P(v_i | d_i = 1) \), we find the percentage \( q_i \) of pixels that can be considered uniform in depth for correct and incorrect bounding boxes using Eq. (3.8). As can be seen in Fig. 3.6, \( q_i \) is a good indicator of an object’s presence. Using logistic regression, we fit a sigmoid to arrive at \( P(v_i | d_i = 1) \). In Section 3.6, we also test the use of \( P(v_i = 1 | d_i = 1) = \max_{k,l} P(c_i^{(k,l)} | d_i = 1) \).

The depth validity flag’s estimate is chosen based on information from the confidence map, \( C \). Let \( C = 1 \) denote the case when more than 50% of the pixels inside the object’s bounding box are marked “confident”, i.e. the depth information is assumed to be reliable. We use the same training set as above to infer \( P(d_i = 1 | C) \), obtaining \( P(d_i = 1 | C = 1) \approx 0.96 \). We set \( P(d_i = 1 | C = 0) = 0 \).

3.5 Inference

Depending on the employed detector, we will use different techniques to conduct inference over the proposed model.

When using the originally employed ISM detector, the obtained confidence maps can be used to do pixel-level reasoning: the exact modeling of interactions between different hypotheses is of paramount importance in highly dynamic scenarios, where pedestrians overlap frequently and
3. Probabilistic Scene Analysis

Figure 3.6: Distribution of depth inliers for correct (a) and incorrect (b) detections, learned from 1,578 annotations and 1,478 negative examples. Based on these distributions, we learn a classifier using logistic regression.

compete for the same pixels. To overcome the missing notion of exclusion in a Belief Propagation framework, we use a two-stage procedure that first infers geometric context using possibly overlapping bounding boxes and then applies an optimization step that models interactions between different objects on a pixel level, using the detector’s confidence maps.

If the detector however does not provide a confidence map, a standard non-maximum suppression will be used prior to running Belief Propagation. This is often already integrated into the detector when using an off-the-shelf implementation.

3.5.1 Belief Propagation

The graph of Fig. 3.1 is constructed for each frame of the video sequence, based on the object detections, as well as the per-frame depth-map cues for the ground plane. All variables are modeled as discrete entities and their conditional probability tables (CPTs) are filled in defined as described above, see Tab. 3.2 for a summary. Inference is conducted using Pearl’s Belief Propagation [Pearl, 1988], using the sum-product al-
### 3.5. Inference

#### CPT | Description
--- | ---
Ground plane

\[ P(\pi) \] | prior learned from sequence, Fig. 3.3
\[ P(\pi_D | \pi) \] | diagonal entries only, Eq. (3.4)

**Objects**

\[ P(c_i | \pi, d_i = 1) \] | distance correspondence, Eq. (3.7)
\[ P(c_i | \pi, d_i = 0) \] | uniform distribution, no comparison possible
\[ P(v_i | d_i = 1) \] | assumption of object flatness, Eq. (3.8)
\[ P(v_i | d_i = 0) \] | uniform distribution
\[ P(v_i | c_i, \pi) \] | height and distance assumptions, \( P(G_i^h)P(z(o_i)) \)
\[ P(I | v_i) \] | object detector probability

**Depth**

\[ P(d_i | C) \] | confidence map-dependent depth prior

**Table 3.2:** Summary of conditional probability tables (CPTs) employed in the model, with their respective factors.

Algorithm to obtain the marginals for every variable. Due to the loopy nature of our model, this yields only an approximate solution. We found this to be more than sufficient in our application, which is also confirmed by other researchers’ experience [Murphy et al., 1999].

#### 3.5.2 Interaction Modeling

As stated above, the reliance of object hypotheses on a common image introduces implicit loops in our graphical model, since overlapping detections cannot be considered independent. This is especially the case when using the ISM detector, as we use its output before any non-maximum suppression stage. Intuitively, each image pixel can only be explained by a single object, therefore some detections are mutually exclusive. The idea of our approach is to make this dependence explicit and use a Quadratic Pseudo Boolean Optimization formulation to select a subset of object detections that are mutually consistent.
Starting from the validity flags $v_i \in \{0,1\}$, we want to optimize the function

$$\max_v v^\top Q v = \max_v v^\top \begin{bmatrix} q_{11} & \cdots & q_{1n} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nn} \end{bmatrix} v$$  \hspace{1cm} (3.10)$$

where the interaction matrix $Q$ contains individual merit terms in the diagonal elements $q_{ii}$ and (negative) interaction terms in the off-diagonal elements $\{q_{ij}, q_{ji}\}$.

Using a similar derivation as in [Leibe et al., 2008a; Leonardis et al., 1995], we express a detection’s score in terms of the pixels $p$ it occupies (normalized by the detection scale),

$$P(o_i|\pi, d_i, I) = P(o_i|\pi, d_i) P(o_i|I)$$  \hspace{1cm} (3.11)$$

$$\sim P(o_i|\pi, d_i) \prod_{p \in o_i} P(p|o_i)$$

with $P(o_i|\pi, d_i)$ the MAP estimate from Belief Propagation. Let $L_i = \log P(o_i|\pi, d_i)$ and $F_i(p) = P(p|o_i)$. We define the cost of a detection by its log-likelihood in a first-order approximation,

$$S = \log \left[ P(o_i|\pi, d_i) \prod_{p \in o_i} F_i(p) \right]$$  \hspace{1cm} (3.12)$$

$$= L_i + \sum_{p \in o_i} \log F_i(p)$$  \hspace{1cm} (3.13)$$

$$= L_i - \sum_{p \in o_i} \sum_{n=1}^{\infty} \frac{1}{n} (1-F_i(p))^n$$  \hspace{1cm} (3.14)$$

$$\approx L_i - N + \sum_{p \in o_i} F_i(p) .$$

Following [Leibe et al., 2008a], we thus arrive at the following merit terms

$$q_{ii} = -\kappa_1 + \sum_{p \in o_i} ((1 - \kappa_2) + \kappa_2 F_i(p)) + \kappa_2 L_i$$  \hspace{1cm} (3.15)$$

where $\kappa_2$ is a regularization term to compensate for unequal sampling and $\kappa_1$ is a counterweight. Two object detections $o_i$ and $o_j$ interact if they compete for the same pixels. In this case, we subtract the support
of the detection \( o_k \in \{ o_i, o_j \} \) that is farther away from the camera in the overlapping image area, assuming it is partially occluded:

\[
q_{ij} = -\frac{1}{2} \left( \sum_{p \in o_i \cap o_j} \left( (1 - \kappa_2) + \kappa_2 F_k(p) \right) + \kappa_2 L_k \right)
\]  \hspace{1cm} (3.16)

Using this formulation, we implement the following iterative procedure. We initialize the Quadratic Problem with the MAP estimate \( P(o_i | \pi, d_i) \) and then solve it using standard optimization techniques [Leibe et al., 2008a; Leonardis et al., 1995]. This results in a subset of mutually consistent detections \( \{ o_i^* \} \), which are then again used to obtain a more stable ground plane estimate. In our experiments, this procedure converged to a stable solution in only few iterations.

3.6 Results

We experimentally validate our system on 3 test sequences of busy shopping streets (Seqs. Bahnhofstrasse, Jelmoli, and Loewenplatz), taken on different days and under different weather conditions. In the following, we perform systematic experiments to underline some design choices and then apply the system with fixed parameters. In the comparisons, annotations and detections smaller than 60 px in size are filtered out. For a detection to be counted as correct, it has to overlap with an annotation by more than 50% using the intersection-over-union measure [Everingham and others (34 authors), 2006]: let \( A \) and \( B \) be the image areas occupied by the annotation and the detection bounding boxes, respectively. Then, the criterion for accepting a detection as correct is

\[
\frac{A \cap B}{A \cup B} > 0.5
\]  \hspace{1cm} (3.17)

Only one detection per annotation is counted as correct, the rest are considered false positives. For the experiments, only the left camera is evaluated. Performance could be improved further by integrating the right camera and adding temporal smoothing [Ess et al., 2009b; Giebel et al., 2004; Leibe et al., 2007a], which is not yet done in the following experiments.
Figure 3.7: (a) Influence of center/scale sampling and $\sigma_h$ on performance. In all future experiments, we use $3 \times 3$ sampling and $\sigma_h = 0.12$. (b) Influence of depth term choice on performance, a parametric distribution performs better.

### 3.6.1 Systematic Experiments

The experiments in this section are performed on the training sequence using the ISM detector. They are used to determine the remaining parameters of the Bayesian network before it is applied to the test sequences.

First, we consider the standard deviation $\sigma_h$ of the size prior, along with the sampling range $\{k, \ell\}$ in which the graphical model can shift the object center location $c_i$. We consider no sampling, $3 \times 3$ ($k, \ell \in \{-1, 0, 1\}$), and $5 \times 5$ ($k, \ell \in \{-1, -0.5, 0, 0.5, 1\}$) sampling. Fig. 3.7 (a) shows the resulting detection performance. As expected, a higher $\sigma_h$ yields better precision at first, but recall grows too slowly. Due to the increased number of choices in Belief Propagation, the use of $5 \times 5$ sampling steps has also a negative effect on the performance. By just fixing the object center, recall is limited, as the algorithm cannot compensate for misaligned bounding boxes. A $3 \times 3$ sampling with $\sigma_h = 0.12$ thus seems a good compromise.

Secondly, we experimentally establish how to integrate the depth cues into our system. For $P(c_i| \pi, d_i = 1)$, we consider either the learned non-parametric distribution $P(v_i | z(o_i) - z(D, b_i))$ (“npar”) or a normal distribution inferred from Eq. (2.24) (“par”). As can be seen from the
3.6. Results

Figure 3.8: (a) Entropy of message from objects to ground plane as a function of the number of objects. (b) Entropy of message from depth map to ground plane as a function of the image area covered by objects (and thus obstructing view on the ground plane).

result plot (Fig. 3.7(b)), the non-parametric distribution for $P(c_i | \pi, d_i = 1)$ performs worse. This is mostly due to a relatively small number of samples (especially at larger depths) for creating the necessary tables, as well as to a bias introduced by annotations and the training ground plane.

Our probabilistic approach to ground plane estimation was motivated by the idea that stereo depth based ground plane estimation and object detection can compensate for each other’s weaknesses. In order to verify if this is indeed the case, we present the following experiment. In Fig. 3.8 (a), we measure the entropy of the incoming messages from the objects to the ground plane node. As can be seen, the larger the number of objects, the lower the entropy, i.e., the presence of many objects constrains the ground plane in a meaningful way. On the other hand, when there are hardly any objects, most of the depth map will contain evidence for the ground plane and will thus constrain it well. This is reflected in Fig. 3.8 (b): the more image area is covered by objects, the less is covered by the ground plane. Thus, the entropy of the message from depth map to ground plane gets higher, as almost all ground planes become equally likely (in this case, a uniform distribution corresponds to an entropy of 5.3, indicated by the dotted red line in the plots).
Figure 3.9: Performance of different system parts and baselines for Seq. Bahnhofstrasse (a) and Seq. Jelmoli (b), using the ISM detector. The interaction of cues yields a substantial increase in performance. The square indicates the operating point used for the sample images.

3.6.2 Experimental Evaluation

With all parameter choices motivated in the previous sections, we now apply the proposed system to a set of challenging test sequences of strolls through busy pedestrian passages. In these experiments, we also compare our system to a set of baseline configurations emulating other approaches from the literature. Fig. 3.9 shows the corresponding performance plots. “ISM” refers to the output of the ISM pedestrian detector, without its global optimization stage. This is the input to our system. “ISM+opt” includes the optimization and is therefore a fair baseline comparison. Neither of these two approaches use scene geometry. The setup “ISM+real GP” is motivated by [Leibe et al., 2007a]. It does not consider depth cues and is obtained by pre-selecting a ground plane for each frame (determined using robust, LMedS-based plane fitting through reconstructed wheel contact points), with a temporal lookahead corresponding to a travelling distance of $\approx 5$ m. Note that without this lookahead, we could not get usable estimates for the ground plane using our hardware setup. “BN” stands for the MAP estimate obtained using Belief Propagation, “Full sys.” is the final output of our proposed ap-
Figure 3.10: Experimental results obtained on Seq. Bahnhofstrasse (top) and Seq. Jelmoli (bottom) with the ISM detector as hypothesis generator. Red boxes indicate false positives.

On its own, the detector’s precision is low, as its score is not distinctive enough. Slightly better results are obtained by including the global optimization. Substantially better results however ensue from including scene and depth information using the graphical model (resulting in an 8% gain in recall at 1.5 FP/image). This still disregards object-object interactions. The baseline “ISM+real GP” considers these, but relies on a pre-selected groundplane, yielding an advantage compared to the detector, with a slight improvement over the MAP estimate. Compared to this baseline, the full system increases recall by a significant 19% at 1.5 FP/image. By replacing the depth uniformity cue with \( P(v_i = 1|d_i = 1) = \max P(c_i|\pi, d_i = 1) \), performance drops by 7%, showing that this additional depth information is indeed beneficial. The plot for Seq. Jelmoli further corroborates the advantage of an integrated approach as presented here.
Figure 3.11: Top: Distribution of pedestrians over distance. (a) annotations, (b) correct detections from ISM. Bottom: Recall and FP/image at globally fixed operating points over distance in Seq. BAHNHOFSTRASSE.

Example Images. Some example detections on the test sequences are displayed in Fig. 3.10. Note the level of interaction between pedestrians (frequently overlapping bounding boxes). The images also show some typical false positives in red (trees, child strollers, signs, mannequins). These false positives are in most cases consistent, and will thus also produce false positive trajectories. This is however not too problematic in applications for path planning, as they can still be regarded as obstacles.
3.6. Results

Figure 3.12: Comparison of different detectors on (a) Seq. Bahnhofstrasse and (b) Seq. Loewenplatz. While the various detectors differ in performance, they all benefit from the additional context obtained from depth maps and ground plane.

Depth Dependency. Note that even at the chosen low threshold for the pedestrian detector, with many false positives, only a recall of about 70% is reached. The reason for this is our challenging test set with significant partial occlusions, many pedestrians appearing at small scales, as well as ISM’s preference for side views that are rather rare in these sequences. To further investigate the influence of a pedestrian’s distance on recognition performance, we compare the average distance distribution of annotated pedestrians in Seq. Bahnhofstrasse together with the agreeing detections (Fig. 3.11 (a,b)). For distant pedestrians, the detector becomes less reliable. For Fig. 3.11 (c,d), we fixed the operating point at 1 resp. 1.5 FP/image on the global curve, and plot recall and FP/image over depth (full system). Recall is considerably higher for distances up to 15 m and rapidly decreases after that, which coincides with the number of available detections.

Detectors. The scene analysis system is independent of a particular detector choice. Fig. 3.12 therefore shows a comparison between the three detectors introduced in Chapter 2: the originally used ISM detector [Leibe et al., 2005], the part-based model by [Felzenszwalb et al.,]
Probabilistic Scene Analysis [Dalal and Triggs, 2005]. As mentioned in Section 3.5, the global optimization stage is replaced by standard non-maximum suppression in the case of the part-based and the HOG detector.

In general, the HOG detector gives the best results in terms of raw detections, with the part-based model of Felzenszwalb a close second. ISM detection performance is slightly worse, mostly due to the fact that it was not run on the double image size and due to its preference for side-views, which are rather rare in these sequences. As expected, the output of the graphical model considerably reduces the number of false positives by introducing scene knowledge, regardless of the raw detection input. Maximally reachable recall is hardly affected, i.e., the model only seldomly discards correct detections. Still, the better the detector, the less improvement is obtained by the scene analysis system. More involved models accounting for partial occlusions and complex ground surfaces are expected to bring an improvement here.

Several other authors reported detection performance on Seq. BAHNHOFSTRASSE. Delivering both detection and segmentation based on edgelet features, [Wu et al., 2008] clearly outperform the only system in our benchmark that also provides a segmentation, ISM, with a recall of 60% at 2 FP/image. This is slightly worse than the HOG detector (64% at 1 FP/image), which was however run on an upscaled image. Using a combination of HOG features and local motion cues, [Wojek et al., 2009] reach a recall of about 64% at 1 FP/image. Their detector performs better than HOG especially on small pedestrians that are not included in our evaluation. Note that both detectors could be used as additional input to our system, with the hope of further improving their performance through context.

Due to its superior performance, we will restrict ourselves to experiments with the HOG detector in the following chapters, unless noted otherwise.

Runtime. Runtimes for the system on Seq. BAHNHOFSTRASSE are given in Tab. 3.3, where we report both statistics on the total time, as well as normalized by the numbers of detections. The generation of the ground-plane evidence from the map has a rather constant overhead, whereas the generation of the CPTs takes up most of the time,
3.7. Preparation for Tracking

Table 3.3: Runtimes for scene analysis system on Seq. Bahnhofstraße.

<table>
<thead>
<tr>
<th>Component</th>
<th>Time (ms)</th>
<th></th>
<th></th>
<th>Time (ms/det.)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td>mean</td>
<td>med</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>Ground-plane evidence</td>
<td>21</td>
<td>65</td>
<td>34</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPT generation</td>
<td>20</td>
<td>116</td>
<td>67</td>
<td>68</td>
<td>3.1</td>
<td>4.6</td>
</tr>
<tr>
<td>Belief propagation</td>
<td>2</td>
<td>20</td>
<td>8</td>
<td>8</td>
<td>0.3</td>
<td>1.2</td>
</tr>
<tr>
<td>Scene analysis total</td>
<td>43</td>
<td>199</td>
<td>109</td>
<td>110</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

scaling rather linearly with the number of detections. As the CPTs are
independent of each other, this generation could be parallelized in future work. Also the current implementation of the belief propagation
algorithm is more tuned to generality than speed, but performs well enough for our purposes.

3.7 Preparation for Tracking

To use the detections in the tracking system that will be introduced in Chapter 4, they need to be placed in a common 3D world coordinate system, as described in this chapter.

3.7.1 Positioning

After passing the Bayesian network, the detections can be located in the camera system. If enough depth information is available, \( i.e. P(d_i = 1|\mathcal{E}) > 0.5 \), we opt to use the depth map to infer the position,

\[
d = \min_{\text{pixel } p \in \text{bbox}_i} D(p),
\]

\[
X_{\text{cam}} = K^{-1} \begin{pmatrix} x \\ y + s \frac{h}{2} \\ 1 \end{pmatrix} d.
\]

Using a stereo rig, this is the best option, giving good results within the accuracy bounds as discussed in Section 2.4.3. If not enough depth
information is available, the bounding box-ground plane localization (Eq. (3.5)) is used as a fallback,

\[ \mathbf{X}_{\text{cam}} = \mathbf{G} \quad . \quad (3.20) \]

Given the location in the camera frame \( \mathbf{X}_{\text{cam}} \), the corresponding world coordinates are inferred by inverting Eq. (2.29),

\[ \mathbf{X}_{\text{world}} = \mathbf{R}^\top (\mathbf{X}_{\text{cam}} - \mathbf{t}) \quad , \quad (3.21) \]

with \((\mathbf{R}, \mathbf{t})\) inferred from visual odometry as described in Section 2.5.

**Cars.** One problem for tracking is that the position measured using either the depth map or the bounding box is not consistent with the actual barycenter of the object. For pedestrians, this is negligible, as their footprint is approximately equilateral and smaller than the accuracy that we can obtain.

In contrast, cars violate both statements due to their rectangular and rather large footprint. Most importantly, their barycenter shifts from the measured point, depending on the viewpoint. Especially for cars that undergo a change in orientation along their trajectory, this shift needs to be compensated. Not knowing the true dimensions of a car, we thus assume a standard size \((1.8 \times 4 \text{ m})\) and add a viewpoint-dependent vector to the measured point.

### 3.7.2 Clustering

Using both cameras and multiple detectors (for cars) results in multiple image responses per real-world object. To reduce the computational overhead in later stages, it is advantageous to cluster highly similar points. To this end, we employ a single-link agglomerative clustering on ground-plane measurements \( z_k = [X_{\text{world}}^{(1)}, X_{\text{world}}^{(3)}, \theta_k] \) inferred from the objects \( o_k \). The similarity function is based on the closeness of two detections in world coordinates, as well as their similarity in appearance,

\[ d(i, j) = \exp \left( -\frac{1}{2} (z_i - z_j)^\top (C_i + C_j)^{-1} (z_i - z_j) \right) \cdot d_{\text{app}}(i, j) \quad , \quad (3.22) \]
3.8 Conclusion

In this chapter, we have presented a system that integrates depth and appearance information for robust object detection and simultaneous ground-plane estimation from video streams. Based on input cues from object detection and depth maps, it constructs a Bayesian network and resolves it in a novel two-stage process. The advantage of an integrative approach combining scene context and detection is shown in a series of experiments. The key message of our experiments is that, given a reasonable pedestrian detector, our algorithm gives it a considerable boost in performance. This is due to the integration of spatial constraints in the form of robust depth cues and the ground plane, our system’s ability to compensate for inaccuracies of the detector, and to resolve object-object interactions.

While the model consistently improves the results of a basic object detector, the margins become smaller the better the detector gets. On the one hand, this is to be expected, as a perfect detector would not need any additional reasoning anymore. On the other hand, the margins could still be improved by a more intricate modeling of the scene. On the geometric side, an introduction of more flexible ground surfaces that account for hilly scenarios or sidewalks could help. The employed depth cues are currently also quite basic, employing more features could help filtering out false positives such as trees. Going in the opposite direction, depth could also be used to trigger active search of an appearance-based detector, or its estimation be coupled more deeply with the model. Another thing

with $C_i$ a detection’s positional uncertainty as defined in Eq. (2.24), and $d_{app}(i,j)$ appearance similarity based on color histograms, as described later in Section 4.3.2.

In future work, the problem of multiple responses could be largely resolved by applying a multi-class/multi-viewpoint detector, which would only give one response per image region. At the time of writing, no such detector was available and we found that this conservative clustering procedure resolves problems obtained from multiple detections in a satisfying manner.
also ignored in the current state is partial occlusion reasoning, which, however, is also not yet accounted for by the used detectors.

Alternatively, the model could be integrated with patch-wise scene reasoning as introduced in Chapter 7, including a probabilistic ground plane estimation might however render this infeasible for real-time applications.

As introduced in this chapter, the model ignores temporal context from tracking and visual odometry. After discussing tracking in the upcoming chapter, this missing link will be introduced in Chapter 5.
Multi-Object Tracking

The aim of the tracking stage is to group detections of the past and current frames into meaningful and physically plausible trajectories, which can be used to, e.g., cast predictions of a specific person’s future motion. To this end, we build upon a hypothesize-and-test framework which uses model selection to optimize object detection and trajectory estimation in an alternating fashion [Leibe et al., 2008b]. Here, we adopt this framework for our application, as it proved to be a powerful means to handle busy urban scenarios from a moving platform [Ess et al., 2008; 2009a].

This chapter is structured as follows: after reviewing related work in Section 4.1, an overview of the algorithm is given in Section 4.2. Data association by means of both a dynamic and an appearance model is treated in Section 4.3. The actual tracking framework works by first sampling hypotheses (Section 4.4) and then selecting the globally best solution (Section 4.5). Implementation details are discussed in Section 4.6. Quantitative and qualitative results on several challenging test sequences are presented in Section 4.7, before the chapter is concluded in Section 4.8.

4.1 Related Work

Tracking as a classical data association problem has been investigated for a long time. Especially in radar tracking, several seminal approaches were proposed as early as in the seventies [Morefield, 1977; Reid, 1979; Fortmann et al., 1983]. Some of these methods are still employed, and
the basic ingredients of transition (dynamic) model describing the objects’ motion patterns, measurement model relating the object state to the observed data, and optimization strategy to infer the most likely state from the model were established already back then. A main advantage of vision as opposed to point tracking is the rich appearance information, which can alleviate the problem of data association in case of close-range targets. On the downside, the measurement model is often very noisy, both due to the actual detector as well as to the nature of perspective image projection (i.e. occlusions, scale changes, ...). Also, practical robotic setups are monocular or binocular with small baseline, which makes localization in 3D relatively inaccurate (c.f. Chapter 2). In the following, we will review different tracking approaches under the aspects of the employed dynamic, appearance, and measurement models, as well as the optimization scheme. A special focus will be given to recent, vision-based tracking-by-detection approaches, as these come closest to the work at hand.

Dynamic Model. Although the dynamic model of the target plays a central role in data association, there are only few commonly used approaches. In general, one can distinguish between models operating in physically meaningful 3D world coordinates and models operating directly in the image plane. In the case of tracking in 3D, e.g. based on a known ground plane calibration or stereo depth, a constant-velocity model in physical coordinates is the standard choice [Ess et al., 2008; Gavrila and Munder, 2007]. When tracking is performed in the image plane, 3D position is often replaced by image position and object scale [Okuma et al., 2004; Wu and Nevatia, 2007a; Zhang et al., 2008], but the dynamic models usually remain of first order, which is a sufficient approximation for most applications. Few authors investigated higher-order models for erratic motions, e.g., in sports [Okuma et al., 2004], or for interacting targets [Khan et al., 2005]. A recent development is to learn image flow fields as dynamic models for densely crowded scenarios [Ali and Shah, 2008]. This method does not easily generalize to different scenarios, particularly not to moving cameras, where the flow field constantly changes.

In this work, we will track in 3D world coordinates and use a constant-velocity model for pedestrians, respectively the Ackermann model for
In Chapter 6, we also propose a more elaborate model that takes into account the tracked person’s environment and that anticipates collisions.

**Appearance Model.** Also for the appearance model, conceptually similar approaches have been used in the past, the bulk of which is based on color histograms [Ess *et al.*, 2008; Leibe *et al.*, 2008b; Nummiaro *et al.*, 2003; Okuma *et al.*, 2004; Wu and Nevatia, 2007a]. Recently, some researchers started to employ online learning to generate more accurate models (*e.g.*, [Grabner and Bischof, 2006]), which proves especially helpful when an object’s appearance is discriminatively trained against all other objects [Breitenstein *et al.*, 2009; Song *et al.*, 2008]. Other methods use additional gait information inferred from images to aid in the association problem [Andriluka *et al.*, 2008]. We found a color histogram in HSV space to be sufficient for our application, even when tracking around 15 people in a busy scenario.

Given both dynamic and appearance model, it is the question how to weight their influence when associating tracks with observations. Recently, [Li *et al.*, 2009] suggested to learn the association metric between tracklets for an offline tracker. In its present formulation, this can however not yet be applied to an online tracker as presented here. In our work, we found a multiplication of the factors to yield good results.

**Measurement Model.** The measurement model has evolved considerably over the past years. Many approaches, especially early ones, depend on background subtraction [Stauffer and Grimson, 1999; Toyama *et al.*, 1999] followed by blob detection to generate measurements. This allows for simple, but general blob-based tracking [Isard and MacCormick, 2001; Berclaz *et al.*, 2006; Lanz, 2006], and can be extended to more intricate shape models, *e.g.*, [Zhao *et al.*, 2008]. A major limitation of background subtraction is the need for a static camera. This constraint is relaxed by using image information such as edges [Isard and Blake, 1998] or local regions [Bibby and Reid, 2008]. However, such low-level structures are susceptible to image clutter. Alternatively, some researchers combine the measurement model directly with the appearance model [Avidan, 2005; Grabner and Bischof, 2006] by some sort of
template matching, \textit{e.g.}, using online boosted classifiers [Grabner and Bischof, 2006].

In recent years, appearance-based object detection has made considerable progress (\textit{c.f.} benchmarks such as [Dollar \textit{et al.}, 2009; Everingham \textit{et al.}, 2008; Enzweiler and Gavrila, 2009]). As a consequence, using the output of an object detector as measurement model has become increasingly popular [Andriluka \textit{et al.}, 2008; Avidan, 2005; Breitenstein \textit{et al.}, 2009; Ess \textit{et al.}, 2008; Huang \textit{et al.}, 2008; Okuma \textit{et al.}, 2004; Gavrila and Munder, 2007; Leibe \textit{et al.}, 2008b; Wu and Nevatia, 2007a; Zhang \textit{et al.}, 2008]. By having an actual notion of the object class, these approaches can restrict tracking efforts to promising image regions and also help in re-initialization after failure. Most detection-based trackers operate on the post-processed bounding boxes delivered as a final output of the detector, although some [Breitenstein \textit{et al.}, 2009; Ess \textit{et al.}, 2008; Leibe \textit{et al.}, 2007b] employ a deeper coupling, directly using the (discrete) distribution of detection probabilities.

In our experiments, we rely on either one of two state-of-the-art detectors [Dalal and Triggs, 2005; Leibe \textit{et al.}, 2005] (Chapter 2), but improve the reliability of their output by introducing additional scene context as proposed in Chapter 3.

**Optimization.** The last required component is an optimization strategy, in order to infer the most likely solution under the three models introduced above. Many trackers follow a first-order Markov assumption. Under this assumption, the state posterior can be estimated only from the state in the previous frame and the new observation by means of either a (Extended) Kalman filter [Gelb, 1996], Mean-Shift tracking [Comaniciu \textit{et al.}, 2003], a Joint Probabilistic Data Association Filter (JPDAF) [Fortmann \textit{et al.}, 1983], a particle filter [Isard and Blake, 1998], or combinations thereof [Schulz \textit{et al.}, 2001; Rasmussen and Hager, 2001].

These methods however quickly reach their limits as the number of tracked targets increases. To prevent the ensuing combinatorial explosion of the state space, researchers typically use one independent Markovian filter per target, with the filters only interacting in the data association stage. This interaction mostly amounts to optimizing the data assignment per-frame using, \textit{e.g.}, the Hungarian method [Munkres,
1957]. Since the state information from all frames but the last one has been discarded (the first-order Markov assumption), a powerful observation model is required to reduce the risk of drifting away from the targets, or at least drifting between targets (switching identities) in busy scenarios.

Drifting can be reduced by optimizing data assignment and considering information over several time steps. In Multi-Hypothesis Tracking (MHT) [Reid, 1979; Cox, 1993], the k-best assignment algorithm [Murty, 1968] is used in each step to generate the k-best data associations (including virtual associations with missing detections), thus generating an entire tree of hypotheses, where each one corresponds to one possible state of the entire world. This method is especially popular in robotics [Arras et al., 2003]. Without careful pruning (usually done using N-scan back [Kurien, 1990]), MHT quickly becomes prohibitive. E.g., to account for short occlusions, a tree size of \( N = 10 \) is at least necessary, which, when starting from two hypotheses, already gives \( 2^{10} = 1024 \) hypotheses to handle after 10 frames. Even though methods exist to obtain the k-best global hypotheses [Cox and Miller, 1995], the originally assumed uniform entrance/exit probabilities are hardly justified in videos seen from a street-level observer. Furthermore, by operating on the data assignment itself, physical exclusion is only handled indirectly by the detector stage in the image plane (e.g., by non-maximum suppression of detection outputs)—without a measurement, exclusion cannot be modeled and the tracker can assume two objects at the position. We will refer to such a physically impossible result as a space-time violation.

The combinatorial explosion can be limited by taking a more global view and first generating secure “tracklets”, thereby reducing the state space. These tracklets are then linked to a set of trajectories by global optimization [Andriluka et al., 2008; Huang et al., 2008; Kaucic et al., 2005; Li et al., 2009; Nillius et al., 2006; Perera et al., 2006; Yan et al., 2006]. While this gives high-quality results, it is much more suited for offline applications, because in the presence of complex interactions, tracklet generation fails and the full trajectories can only be recovered in hindsight.

Instead of limiting the state space of associations, [Berclaz et al., 2006] suggest to discretize the object state (location) to a grid. In this case, the global optimum for a single trajectory can be calculated using the Viterbi
algorithm [Viterbi, 1967]. However, the extension to multiple targets is done in a greedy fashion, iteratively running the algorithm and removing the most probable candidate, which turns out to be a heuristic for the exact algorithm defined in [Wolf et al., 1989]. For mobile applications, defining a grid is difficult and the reduced positional accuracy is also unwanted for path prediction.

Recently, some authors suggested the use of Markov chain Monte Carlo (MCMC) sampling to find the (approximate) optimal solution in the joint tracking space [Khan et al., 2005; Song and Nevatia, 2007; Yu et al., 2007; Zhao et al., 2008]. This method generates a sequence of states, which collectively approximates the target distribution. Coupled with a Viterbi algorithm, this can be thought of as a stochastic version of hypothesize-and-test, where the world state is represented by a set of samples.

[Jiang et al., 2007] propose the use of Integer Linear Programming (ILP) [Schrijver, 1998] to find a globally consistent solution, accounting for exclusion and occlusion between objects. While the authors provide a description to its online application, the approach is rather slow, in the presented formulation susceptible to changes in the bounding box, and needs to know the number of objects in the scene a priori.

An elegant solution to the data association problem, inspired by graph theory, was proposed in [Zhang et al., 2008]. In their approach, each detection is represented as two nodes in a graph, and edges are used to model transition, enter, and exit probabilities, respectively. The globally optimal assignment of detections to trajectories is then obtained by repeatedly running a min-flow algorithm with different flows, where the flows intuitively correspond to the number of people in the scene. Explicit occlusion reasoning can then be done in a second step by constructing a set of virtual, occluded detections and re-running the algorithm. Similar to the methods above, this one operates on detections only, thus failing to model space-time violations. Furthermore, possibly missing detections need to be accounted for by adding extra edges to the graph, which can also become prohibitive if prolonged misses due to, e.g., unknown scene occluders need to be taken into account. Lastly, the method is more suited for offline application: as with the tracklet-combining algorithms, the method per se does not generate an output for objects without a measurement in the current time frame. Note that
this algorithm can also be combined with the tracklet methods introduced above, where detection nodes would be replaced by tracklet nodes [Li et al., 2009].

[Morefield, 1977] was probably the first to suggest tracking by means of a two-stage hypothesize-and-test framework. In his method, an overcomplete set of trajectory candidates is generated, given an entire batch of measurements from all time steps. ILP is then used to find a consistent subset, with the constraint that each data point (i.e. detection) is used only once. This can be thought of as an instance of the so-called set packing problem, one of Karp’s 21 NP-complete problems [Karp, 1972]. Independently, [Leibe et al., 2007a] proposed Space-time Event Cone Tracking (SpECTr), where hypotheses are generated in an online fashion and selected using statistical model selection. To account for missing measurements, interactions are not only modeled by penalizing the repeated use of the same detection, but also the simultaneous occupation of space-time volume. This framework might not be the best choice for offline applications or static cameras, as the hypothesis space still needs to be controlled and could potentially have problems with objects that stay static for multiple minutes. However, we showed its applicability to busy urban scenarios filmed from a mobile observer [Ess et al., 2008; 2009a]. In our work, we specifically focused on: the introduction of scene knowledge to make the underlying detector and hence hypothesis generation more reliable (Chapter 3), improved hypothesis generation including incremental updates of the interaction matrix (Section 4.4), a faster optimization scheme (Section 4.5), integration with detection and visual odometry (Chapter 5), extension of the system to the class car, and a C/C++ implementation that achieves frame rates of up to 20 fps (tracking only).

4.2 Overview

Given the 3D information from Chapters 2 and 3, we can localize detections in a common world coordinate frame and thus opt to perform tracking in world coordinates on the ground plane. The basic units of a tracker in the hypothesize-and-test framework are hypotheses (candidates) for possible object trajectories. Such a trajectory hypothesis
is defined as $H_j = [S_j, M_j, A_j]$, where $S_j$ denotes its supporting detections, $M_j$ the employed motion model, and $A_j$ the appearance model. The set of all candidate hypotheses at time step $t$ is denoted $H^t_{\text{cand}}$. This set can be (and usually will be) redundant, with many spurious hypotheses. The verification stage then selects a mutually consistent subset of trajectories $H^t_{\text{sel}}$.

The basis for tracking-by-detection are the detections $o_{i,t_i} = [x_i, C_i, t_i, a_i]$, with $x_i$ the object’s 2D ground plane position, $t_i$ its time-stamp, $C_i$ the covariance matrix capturing its positional uncertainty (Eq. (2.24)), and $a_i$ its appearance. For the sake of brevity, we will mostly omit the subscript $t_i$ in the following. Based on the output of the scene analysis (Chapter 3), we use the notation $p(o_i)$ to define an object’s probability given its detector score and the available scene knowledge. The detections are accumulated in a space-time volume $O$ that spans all previous time steps up to the current frame. To keep the method computationally tractable, $O$ only contains the last few hundred time steps, with $t_0$ the smallest time step still considered by the tracker. For the set of candidate objects at time $t$, we write $O_t$. The aim of the tracking step is thus to fit smooth trajectories $H_j$ to the detected object locations $[x_i, t_i]^\top$ in a 3D spacetime volume $O$.

Unlike traditional Markovian trackers, our approach applies a hypothesize-and-test strategy in order to find the set of trajectories that provides the best explanation for the observed evidence from past and present detections. This step is carried out by sampling a large, redundant set of candidate trajectories and pruning that set to a minimal consistent subset with model selection.

In a first step, the set of candidate trajectories is generated by running the bi-directional trajectory-following method described in Section 4.4, starting from all detections within a large temporal window (for computational efficiency, the candidates from previous frames are cached and extended, and only those starting from new detections are generated from scratch). Each filter generates a candidate trajectory which obeys the physical motion constraints of a walking person/driving car and which bridges short temporal gaps due to occlusion or detection failure. Note that candidates do not only originate from the accepted tracks of the last frame (like in classical trackers built on a first-order Markov assumption).
The obtained candidate trajectories are not independent because of the twin constraints that two objects cannot occupy the same location on the ground plane at the same time and that each object detection can only belong to a single trajectory. Thus, we sample an overcomplete set of trajectories, which is then pruned to a minimal consistent explanation using model selection in a second step. This step simultaneously resolves conflicts from overlapping trajectory hypotheses by letting trajectories compete for detections and space-time volume. In a nutshell, the pruning step employs quadratic pseudo-boolean optimization to pick the set of trajectories with maximal joint probability, given the observed evidence over the past frames. This probability

- increases as the trajectories explain more detections and as they better fit the detections’ 3D location and 2D appearance through the individual contribution of each detection;

- decreases when trajectories are (partially) based on the same object detections through pairwise corrections to the trajectories’ joint likelihoods (these express the constraints that each object can only follow one trajectory and that two objects cannot be at the same location at the same time);

- decreases with the number of required trajectories through a prior favoring explanations with fewer trajectories—balancing the complexity of the explanation against its goodness-of-fit in order to avoid over-fitting (“Occam’s razor”).

Fig. 4.1 visualizes the generation and selection of candidate trajectories for an example scene. There, people are standing closely together, which results in trajectory hypotheses that contain detections from several actual persons (note, e.g., the long curve going to the left). Selecting such a candidate is however suboptimal from a global perspective, as the above-mentioned constraints would preclude the simultaneous selection of other candidates that are based on the same detections. Hence, it is better to select candidates that are mutually consistent with each other.

The most important features of the method are automatic track initialization (usually, after about 2–3 detections) and the ability to recover from temporary track loss and occlusion. In the following, we will describe the various components involved in more detail.
Figure 4.1: Tracking by means of a hypothesize-and-test framework: given the detection output from the current and past frames (a), an over-complete set of hypotheses is constructed (b) and pruned to a minimally consistent set using model selection (c), yielding the final trajectories (d).

4.3 Data Association

In order to reliably associate a trajectory hypothesis with candidate detections, we employ for each hypothesis $H_j$ both a dynamic model $M_j$ as well as an appearance model $A_j$. Together, these can be used to evaluate an observation $o_i$ under $H_j$,

$$p(o_i|H_j) = p(o_i|A_j)p(o_i|M_j) .$$

(4.1)

The probability $p(o_i|H_j)$ is used to score all observations of a time-step against a trajectory hypothesis, as described in Section 4.4. A
4.3 Data Association

The nearest neighbor strategy is then used for updating a trajectory, where \( p(o_i|H_j) \) is gated to include only feasible observations. In the following, we describe the employed dynamic and appearance models.

4.3.1 Dynamic Models

Our application scenario suggests tracking in world coordinates. In accordance with several works in the tracking literature, we use an Extended Kalman Filter (EKF) [Gelb, 1996] to describe an object’s motion in a physically plausible way. Before describing the actual motion models, we briefly review the generic model, before describing the actual motion models for our object categories.

Mathematical Model. In short, an EKF is an instance of recursive Bayesian filtering (for an overview, see e.g. [Arulampalam et al., 2002; Gelb, 1996]) that iteratively repeats the two steps of prediction and update in order to estimate the optimal system state \( x_t \) based on the measurements \( Z_t = \{z_1, \ldots, z_t\} \) up to time \( t \) (Fig. 4.2). In recursive Bayesian filtering, a model-dependent transition function \( f(\cdot) \) defines the state propagation belief \( p(x_t|x_{t-1}) \). Assuming a Markovian system model, the \emph{a priori} distribution of the next time step can be calculated.

In literature, \( k \) is usually used as the time index, we adapt this here to be consistent with the rest of the thesis.
Given measurements up to time $t-1$ via the Chapman-Kolmogorov equation,

$$p(x_t|Z_{t-1}) = \int p(x_t|x_{t-1}) p(x_{t-1}|Z_{t-1}) \, dx_{t-1} \quad .$$

(4.2)

Thus, $p(x_t|Z_{t-1})$ depends only on $p(x_{t-1}|Z_{t-1})$ and not on any function prior to $t-1$.

By taking a new measurement $z_t$ into account, the predicted distribution is updated according to Bayes’ rule to arrive at the a posteriori distribution

$$p(x_t|Z_t) = \frac{p(z_t|x_t)p(x_t|Z_{t-1})}{p(z_t|Z_{t-1})} ,$$

(4.3)

where the normalization factor $p(z_t|Z_{t-1}) = \int p(z_t|x_t)p(x_t|Z_{t-1}) \, dx$ depends on the likelihood of the measurements $p(z_t|x_t)$.

The recursive Bayesian filter is mainly considered theoretically for state estimation. Due to the large state space for multidimensional state vectors, the evaluation of the prior probability of each point quickly becomes intractable.

The employed EKF framework, an extension of linear Kalman filtering, assumes a unimodal Gaussian distribution of the current state, and is specified by defining the transition function $f(\cdot)$, the measurement function $h(\cdot)$, as well as their respective Jacobians $F$ and $H$. Fig. 4.3 gives the respective process equations for prediction and filter update.

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Prediction} & \textbf{Update} \\
\hline
(1) $s_{t+1} = f(s_t, 0)$ & (1) $K_{t+1} = P_t^{-}H_t^T (H_t P_t^{-} H_t^T + V_t R_t V_t^T)^{-1}$ \\
(2) $P_{t+1} = F_t P_t F_t^T + W_t Q_t W_t^T$ & (2) $s_{t+1} = s_t^{-} + K_t (z_t - h(s_t^{-}, 0))$ \\
\hline
\end{tabular}
\end{center}
4.3. Data Association

Another instance of recursive Bayesian filtering that can handle multimodality would be particle filters. Initial experiments showed a rather similar behavior to an EKF, even when measurements are missing (i.e., a radially growing uncertainty ellipse). Furthermore, as they are not as easily amenable to the implementation of physically sound motion models and overlap in function with the multi-hypotheses approach (which should cater for multi-modality on the hypothesis level), we settled for the EKF.

In the employed EKF framework, motion models only differ in the choice of the state transition function $f_t$ and its noise vector. The measurements $z_t$ are typically the 2D locations of the detections on the ground plane. In the following, we will introduce the models used for pedestrians and cars.

**Pedestrians.** For pedestrians, we assume a constant-velocity model, i.e., the state space is defined as

$$s_t = [x_t, y_t, \theta_t, v_t]^T \quad ,$$  

(4.4)
with \((x_t, y_t)\) the 2D position, \(\theta_t\) the pedestrian’s orientation, and \(v_t\) its speed (Fig. 4.4 (a)). The latter two are initialized to 0, as a detection itself only indicates the position of the person. The corresponding transition function is thus

\[
f(s_{t-1}, w_{t-1}) = \begin{pmatrix} x_{t-1} + v_{t-1} \cos(\theta_{t-1}) \Delta t \\ y_{t-1} + v_{t-1} \sin(\theta_{t-1}) \Delta t \\ \theta_{k-1} \\ v_{t-1} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ w_\theta \\ w_v \end{pmatrix}. \tag{4.5}
\]

\(w_\theta\) and \(w_v\) are the noise components for orientation and velocity, respectively.

**Cars.** For cars, which move non-holonomically due to mechanical constraints, we employ the Ackermann steering model, c.f. e.g. [Cameron and Proberdt, 1994]. This incorporates two so-called driving processes, which are the steering angle \(\phi_t\) and the tangential acceleration \(a_t\), see Fig. 4.4 (b). The state vector therefore is

\[
s_t = [x_t, y_t, \theta_t, v_t, \phi_t, a_t]^\top. \tag{4.6}
\]

This gives rise to the update equation

\[
f(s_{t-1}, w_{t-1}) = \begin{pmatrix} x_{t-1} + v_{t-1} \cos(\theta_{t-1}) \Delta t + \frac{1}{2} a_{t-1} \cos(\theta_{t-1}) \Delta t^2 \\ y_{t-1} + v_{t-1} \sin(\theta_{t-1}) \Delta t + \frac{1}{2} a_{t-1} \sin(\theta_{t-1}) \Delta t^2 \\ \theta_{t-1} + \frac{1}{2} v_{t-1} \tan(\phi_{t-1}) \cdot \Delta t \\ v_{t-1} + a_{t-1} \cdot \Delta t \\ \phi_{k-1} \\ a_{k-1} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ w_\phi \\ w_a \end{pmatrix}. \tag{4.7}
\]

\(L\) is the distance between the axles of the car and is set to a default value of \(L = 3.2\) m in our scenario. The process noise \((w_\phi, w_a)\) is only dependent on the two driving processes.
4.3. Data Association

Note that the EKF cannot account for any constraints on the steering angle. We nevertheless chose this option for its simplicity and good performance.

Application. Given the current state’s position $s_t^{(1,2)}$, the likelihood of an object $o_i$ with position $x_i$ under the motion model is measured as follows:

$$p(o_i | M_j) = e^{-\frac{1}{2}(x_i - s_t^{(1,2)})(C_t + C_{x_i})^{-1}(x_i - s_t^{(1,2)})^\top}$$

where we account for both the uncertainty in the system $C_t$ as well as for the localization uncertainty of the detection $C_{x_i}$. The latter is especially important to handle far away objects correctly, as their localization in depth is highly inaccurate (c.f. Chapter 2). The correct modeling of these two terms proved to be of prime importance for good object tracking across a large working range.

For cars, we extend this function to also take the orientation of the detection $\hat{\theta}_i$ into account,

$$p(o_i | M_j) = e^{-\frac{1}{2}(x_i - s_t^{(1,2)})(C_t + C_{x_i})^{-1}(x_i - s_t^{(1,2)})^\top - \lambda |\hat{\theta}_i - \theta_t|}.$$  

(4.9)

$\lambda$ sets the influence of the orientation similarity on the distance, this value usually also has to accommodate for the spacing between detector orientations. $\theta$ here indicates the directional vector corresponding to the orientation $\hat{\theta}$. The employed motion models are rather simple, but proved to be effective in practice. In particular, the positional accuracy of the underlying detections is not sufficient to support more complex motion models (e.g., incorporating acceleration). A drawback of models typically used in the literature is their assumption of independence: without any measurements, a basic extrapolation is performed, disregarding possible collisions or other interactions between objects. In Chapter 6, we therefore investigate a simulation-based approach, where the motion of a single agent can be influenced by other agents, thus better constraining the search space for data association. This can be especially helpful in the case of re-finding an agent after prolonged occlusion.
4. Multi-Object Tracking

Figure 4.5: Color appearance is calculated using an ellipse fitted to the bounding box, weighted with increasing distance to the center.

4.3.2 Appearance Model

As a hypothesis $H_j$’s appearance model $A_j$, we choose an $(8 \times 8 \times 8)$-bin color histogram in HSV space. For each observation $o_i$, we compute its histogram $a_i$ over an ellipse fitted inside the detected bounding box, applying a Gaussian kernel that radially weights pixels closer to the center, see Fig. 4.5:

$$w_A = e^{-\frac{1}{2}((x-c_x)^2 + (y-c_y)^2)} .$$

(4.10)

For robustness against slight color aberrations, trilinear interpolation is used when building the histogram.

The similarity of an object and a hypothesis is then defined by the Bhattacharyya coefficient between the histograms,

$$p(o_i|A_j) = \sum_q \sqrt{a_i(q)A_j(q)} ,$$

(4.11)

where $q$ is a three-dimensional index over the histogram’s bins.

When a new observation $o_i$ is added to a trajectory $H_j$, $A_j$ is updated using an Infinite Impulse Response (IIR) filter,

$$A_j(q) = (1 - w)A_j(q) + wa_i(q) .$$

(4.12)

The mixing factor $w$ depends on the closeness of the observation and the dynamic model, and is usually thresholded to always keep a minimum
amount of model information. We thus set $w$ dependent on the spatial
closeness of the observation to the trajectory,

$$w = \min \{0.9, p(o_i|M_j)\} \quad .$$ (4.13)

Combined with the above dynamic model, such a simple appearance
model can capture enough information to allow reliable tracking with
many interacting agents. Further improvements could be obtained by
using a discriminatively learned model [Song et al., 2008], where an
object’s appearance is explicitly learned against all the others in an
online fashion. Note that this discriminative approach can however not
be used when trajectory hypotheses are generated totally independently
as in [Leibe et al., 2008b], as it requires the knowledge of objects (i.e.
not only candidates) at each time step to compare classifier confidences.
Thus, as of now, (online) learning of the appearance model is an interest-
ing option for re-identifying an object after it comes out of an occlusion,
but it adds too much overhead and processing time to warrant its ap-
plication in only moderately crowded scenes.

4.4 Hypothesis Generation

The space-time volume $O$ accumulates detection responses from the cur-
rent and past frames and serves as the basis for creating trajectory hy-
potheses.

The employed hypothesize-and-test architecture requires that the set of
hypotheses passed to the selection algorithm has to be overcomplete, i.e.
the hypothesis generation stage has to generate all correct hypotheses
for the subsequent selection to recover an object. However, to keep
the method computationally tractable, the candidate set should also
be as small as possible, as the optimization procedure is NP-complete.
Therefore, we propose a hypothesis generation method [Ess et al., 2009d]
that draws its inspiration from two approaches in the literature: on the
one hand, the “trajectory extension” (Section 4.4.2) is very similar to a
standard Markovian tracker where in each step, hypotheses compete for
evidence (e.g. [Wu and Nevatia, 2007a]). We thus expect a performance
at least as good as a Markovian tracker. On the other hand, we start
Figure 4.6: Visualization of the basic trajectory growing procedure (adapted from Leibe et al., 2008b). (a) Starting from an observation, all detections that adhere to the dynamic model in the adjoining time steps are collected and evaluated under the trajectory model. (b) The trajectory is adapted based on inlier points, and the process iterated both forward and backward in time. (c) This results in a set of candidate trajectories, which are passed to the hypothesis selection stage. (d) For efficiency reasons, trajectories are not built up from scratch at each time step, but are grown incrementally.

searches backwards in time to initialize hypotheses and generate possible additional explanations (Section 4.4.1), the latter of which could be interpreted as an instance of the “observe-and-explain” approach [Ryoo and Aggarwal, 2008].

4.4.1 Independent Generation

In their original implementation, [Leibe et al., 2007a] suggest to start hypothesis generation from each detection in a given time window by applying a motion model and use it to guide the trajectory search up and down in time. This algorithm is visualized in Fig. 4.6 (a–c). In typical scenes, this creates a candidate set with many duplicates, thus requiring an effective pruning mechanism to keep the optimization tractable.

As observed in [Leibe et al., 2008b], the hypothesis set can be constructed in an incremental fashion by (1) starting independent searches backwards in time from all detections of the current time step, and (2) extending hypotheses from the last time step to the current one. We also use the first method, both to detect hypotheses for new objects, as well as to create new, retrospective explanations. Such an “observe-and-explain”
Algorithm 2 Independent hypothesis generation.

// Start search from all detections at time $t$

for all $o_i \in \mathcal{O}_t$ do

// Set up hypothesis $H_j$

$S_j = \{o_i\}$

init $M_j$ using $x_i$

init $A_j$ using $a_i$

// Follow the trajectory backwards in time

for $t' = t$ downto $t_0$ do

// Find best candidate

predict $M_j$

$k = \arg \max_{o_{k'} \in \mathcal{O}_{t'}} p(o_{k'}|H_j)$

if $p(o_k|H_j) > \alpha$ then

$S_j = S_j \cup \{o_k\}$

update $M_j$

update $A_j$

end if

end for

end for

approach [Ryoo and Aggarwal, 2008] limits the combinatorial explosion in comparison to standard multi-hypothesis tracking [Reid, 1979]. In contrast to [Leibe et al., 2008b] however, we use a different extension methodology, described in the following section.

Another change to their implementation is to use a nearest neighbor matching strategy instead of the weighted inclusion of all inliers. When generating/extending the candidate trajectories independently of each other, they cannot compete for measurements—the competition is left to the final selection algorithm. In difficult crowded cases, candidates will therefore include wrong measurements of other nearby objects. To remedy this behavior, we rely on the facts that image-based non-maximum suppression only yields one detection per object and camera, and that the conservative clustering procedure (Section 3.7) merges the measurements from multiple cameras to a single one. Hence, only the detection closest to the EKF’s predicted location is used to update the state, rather than using all nearby detections weighted by the distance. Algorithm 2 summarizes the independent hypothesis generation.
4.4.2 Trajectory Extension

To sustain hypotheses, which were selected in the previous frame and thus believed to be correct, we extend them in a similar way as Markovian trackers do. Specifically, we have trajectories compete for detections, ensuring that each trajectory can be updated with at most one observation.

In order to resolve conflicts which arise when a measurement is the closest one for two or more candidate trajectories, the extension step is carried out simultaneously for all existing candidates, greedily assigning each detection to the trajectory candidate with the closest prediction. Candidates which do not manage to claim any detection during this process are merely extended through extrapolation.

The effect of the competitive hard assignment of detections is twofold. Firstly, it avoids unwanted attraction between candidates and better separates closely interacting pedestrians (when using soft assignment as in the original work, the same measurement can influence several nearby trajectory candidates, pulling them closer together). An example would be two closeby persons that are being tracked: if in any given frame, only one detection is available, both trajectories are potentially influenced by this single detection, which could erroneously pull one trajectory in the wrong direction. Having this competitive assignment makes sure that only the better explanation gets updated. Secondly, the set of candidates tends to be more compact, because each measurement can only support a single candidate in a crowded region, making weak candidates more prone to attrition.

Note that alternative explanations are still available due to the independent generation described before. Algorithm 3 describes the trajectory extension. In a practical implementation, the similarity matrix can be replaced by priority queues.

4.4.3 Occlusion Reasoning

The limited height of most vehicles enforces rather low camera placement, such that pedestrians and/or cars are frequently occluded by each other, or by other scene objects. We therefore opt to explicitly model occlusion, rather than treat it as yet another case of missing detections.
Algorithm 3 Trajectory extension.

// Set up distance matrix
Q = ∅
for all $H_j \in \mathcal{H}_\text{sel}^{t-1}$ do
    predict $\mathcal{M}_j$
    for all $o_i \in \mathcal{O}_t$ do
        $Q_{ij} = p(o_i | H_j)$
    end for
end for

// Greedy selection of best matches
while $Q \neq ∅$ do
    $\{i, j\} = \arg\max_{\{i', j'\}} Q_{i'j'}$
    if $p(o_i | H_j) > \alpha$ then
        $S_j = S_j \cup \{o_i\}$
        update $\mathcal{M}_j$
        update $\mathcal{A}_j$
    end if
    $Q = Q(1:n)\setminus i, (1:m)\setminus j$
end while

Figure 4.7: From the image data (left) we infer occlusion regions (right) due to both static obstacles (black, casting blue umbra) and the previous frame’s object predictions (red umbra). This information is used to correctly treat occluded candidate tracks.
To this end, we generate an occlusion map on the ground plane, again discretized to a polar grid like the occupancy map in Section 2.4.4. An example is shown in Fig. 4.7. The map contains the regions occluded by both pedestrians and static obstacles. To compute the map, object locations are estimated by extrapolating the previous tracker state to the current frame, whereas static obstacles are read out of the occupancy map.

As long as a candidate trajectory remains in an occluded region, it is kept alive and its state is extrapolated. Here the uncertainty modeling of the EKF becomes important: prolonged extrapolation without measurements leads to progressively larger location uncertainty and hence a larger search region for supporting detections. This increases the chances of finding the object once it becomes visible again. The greedy assignment described in Section 4.4.2 meanwhile makes sure that such a candidate does not steal detections from less uncertain competitors. As a result, we obtain longer people tracks, and also keep track of people in occlusion, which better supports path planning [Ess et al., 2009a].

### 4.5 Interactions and Model Selection

Based on the previous steps, a set of possible explanations, some of them mutually exclusive, is generated. The selection of the optimal explanation is described next.

To select the jointly optimal subset of trajectories, we assign each trajectory \( H_j \) a support \( S \), which is composed of the strength of its supporting detections \( \{o_i\} \), weighted by their goodness-of-fit with respect to the dynamic model \( \mathcal{M} \) and the appearance model \( \mathcal{A} \).

\[
S(H_j | \mathcal{I}_{t_0:t}) = \sum_i S(o_{i,t_i} | H_j, \mathcal{I}_{t_i}) = \sum_i p(o_{i,t_i} | \mathcal{A}_{j_i}^t)p(o_{i,t_i} | \mathcal{M}_{j_i}^t)p(o_{i,t_i} | \mathcal{I}_{t_i}), \tag{4.14}
\]

where \( p(o_{i,t_i} | \mathcal{I}_{t_i}) \) is the probability of a detection as defined in Chapter 3, and each detection is evaluated under the trajectory’s appearance and dynamic model at that time.
4.5. Interactions and Model Selection

Choosing the best subset \( \{H_j\} \) is now a model selection task. If we only take into account pairwise interactions\(^2\) it translates to the quadratic binary problem

\[
\mathcal{D}(\mathbf{m}) = \mathbf{m}^\top \mathbf{Q} \mathbf{m} ,
\]

\[
\max_{\mathbf{m}} \mathcal{D}(\mathbf{m}) , \quad \mathbf{m} \in \{0, 1\}^N ,
\]

where \( \mathbf{m} \) is an index vector (length \( N \)), specifying which candidates to use \( (m_i = 1) \) and which to discard \( (m_i = 0) \). The diagonal elements \( q_{ii} \) contain the individual likelihoods of candidate trajectory \( H_i \), reduced by the “model penalty”, a prior which favors solutions with few trajectories. The off-diagonal elements \( q_{ij} \) model the interaction between candidates \( i \) and \( j \) and contain the correction for double-counting detections consistent with both candidates, as well as a penalty proportional to the overlap of the two trajectories’ footprints on the ground plane:

\[
q_{ii} = -\epsilon_1 + \sum_{o_k, t_k \in H_i} e^{-\lambda(t-t_i)} \left( (1 - \epsilon_2) + \epsilon_2 S(o_k,t_k|H_i,I_{t_k}) \right) ,
\]

\[
q_{ij} = -\frac{1}{2} \epsilon_3 O(H_i, H_j) - \frac{1}{2} \sum_{o_k, t_k \in H_i \cap H_j} e^{-\lambda(t-t_k)} \left( (1 - \epsilon_2) + \epsilon_2 S(o_k,t_k|H_\ell,I_{t_k}) \right) ,
\]

\( (4.17) \)

where \( H_\ell \in \{H_i, H_j\} \) denotes the weaker of the two trajectory hypotheses, whose evidence is subtracted to avoid double counting; \( O(H_i, H_j) \) measures the physical overlap between the footprints of \( H_i \) and \( H_j \) given average object dimensions. \( \epsilon_1 \) is the base cost for each new trajectory, required to prevent over-fitting, and should be chosen such that it suppresses trajectories with less than \( \approx 2 \) good detections, in order to weed out erratic false detections; \( \epsilon_2 \) is a regularization parameter, which ensures a minimal support for each explained object detection and compensates for model inaccuracies—smaller \( \epsilon_2 \) puts less weight on the goodness-of-fit in terms of appearance and dynamics, and more weight on the fact that a detection could be associated with the trajectory at all; \( \epsilon_3 \) is the influ-

\( ^2 \)Disregarding higher-order interactions results in too high penalties in cases where more than two trajectories compete for the space and/or detections; if interaction penalties are high enough to enforce complete exclusion, this will not alter the result.
ence weight of the overlap penalty, and should be chosen large enough
to prevent selecting any two trajectories with significant overlap.

Thus, two overlapping trajectory hypotheses compete both for support-
ing observations and for the physical space they occupy during their
lifetime. This makes it possible to model complex object-object interac-
tions, such that two objects cannot occupy the same space-time region.

4.5.1 Optimization

The maximization Eq. (4.16) is NP-hard, but there are several methods
which find strong local maxima, e.g. the multi-branch method of [Schind-
ler et al., 2006], or QBPO-I [Rother et al., 2007]. The solution is a locally
optimal set of object candidates for the current frame: most false de-
tections are weeded out, since they usually do not have a supporting
trajectory in the past (this is the main source of improvement), whereas
missed detections are filled in by extrapolating those trajectories which
have strong enough support in the previous frames.

We use an extended version of the multi-branch method of [Schindler et
al., 2006], summarized in Algorithm 4. The basic algorithm selects at
every level the set of most promising hypotheses and tries adding each
one recursively. At each level $R$, at most $B_R$ hypotheses are tested.
An important insight of [Schindler et al., 2006] is that the function $D$
is submodular ($q_{ii} > 0$ and $q_{ij} \leq 0$ ($\forall i \neq j$)). Due to this property,
the path to the optimum never contains any descent steps. Formally,
given a current choice $m'$, the next step in the search $m''$ must always
fulfill $D(m'') > D(m')$, otherwise, the search along this path can be
terminated. In the above algorithm, this is reflected in the constraint
$s_{k, i} > 0$.

Here, we employ an additional bound. Given a solution $m'$, let $L'$ be
the still available hypotheses (i.e., $\{l_i \in L' : m'_l = 0\}$), and denote by
$1_{l_i}$ a vector that has all 0s except at entry $l_i$. Starting from a current
point $m$, the maximally reachable score never exceeds

\[
s = D(m) + \max \left\{ 0, \sum_{l_i \in L'} \left( D(m + 1_{l_i}) - D(m) \right) \right\}.
\]  

(4.18)
Algorithm 4 Multi-ascent optimization with additional bounds.

<table>
<thead>
<tr>
<th>Level 1:</th>
<th>Set $s_{\text{max}} = 0$ and $m = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level $R$:</td>
<td>Current solution $m$, hypotheses $i = R \ldots N$ are left</td>
</tr>
</tbody>
</table>

// Record best solution

if $D(m) > s_{\text{max}}$ then
    $s_{\text{max}} = D(m)$
    $\hat{m} = m$
end if

if $R = N$ then
    return
end if

// Calculate merit $s_i$ of hypothesis $i$ under current selection $m$

$s_i = D(m + 1_i) - D(m) \quad \forall i = R \ldots N$

if $D(m) + \sum_{i=R}^{N} \max\{0, s_i\} < s_{\text{max}}$ then
    return
end if

// Add hypotheses

Let $k_i$ be the indices of the sort according to $s_i$

for all $i = R \ldots R + B_R$ do
    if $s_{k_i} > 0$ then
        Recursively invoke algorithm with $R = R + 1$ and $m = m + 1_{k_i}$
    end if
end for

That is, the maximal additional score is always bounded by the sum of adding all positive hypotheses, accounting for their interaction with the current solution, but ignoring their interactions amongst each other. Formally, $D(m + 1_i + 1_j) + D(m) \leq D(m + 1_i) + D(m + 1_j)$. As a consequence of this bound, the optimization can quit a branch of the search tree as soon as the maximally reachable score is smaller than the current maximum $s_{\text{max}}$. This early stopping reduces the number of invocations of the search recursion to 63% of the original algorithm. Furthermore, when operating on a video sequence, the last solution can be used to constrain the optimization: to do so, we set $s_{\text{max}} = D(\hat{m}(t-1))$ before invoking the algorithm, where $\hat{m}(t-1)$ are the trajectories deemed successful in the last time step evaluated under the current interaction.

---

3This is in fact the definition of submodularity, c.f. [Boros and Hammer, 2002]
matrix. Doing so ensures that \( s_{\text{max}} \) is always a reachable solution. Due to temporal consistency, this is usually a good starting value for \( s_{\text{max}} \), reducing the number of invocations by another 2% to 61% of the original number (which is only a small improvement, but comes at no extra cost).

**Discussion.** As the maximization is performed on a per-frame basis, there is no guarantee that the current explanation is consistent with the one obtained for the previous frame. Still, when it is selected it not only explains the current frame \( t \), but also offers the most likely explanation for the past, in the light of the entire evidence up to time \( t \). We can thus follow a trajectory back in time to determine where a pedestrian came from when he first stepped into view, even if back then no trajectory was selected for that particular object.

Typically, the model selection step keeps between 25% and 35% of the candidate trajectories. In extreme cases, this figure extends to 8% and 100%, respectively. The ratio is strongly dependent on the momentary complexity of the scene: the closer together the pedestrians move, the more candidates will be created. These are also the cases where a greedy maximization of Eq. (4.16) fails. When using our optimization method, we however did not notice any problems with weak maxima. The limiting factor seems to be model inaccuracy, rather than optimization failures.

### 4.6 Implementation Details

In this section, we review some important details of the practical implementation. Although these details are mainly straightforward engineering considerations, we discuss them in some detail, in the hope that they may be useful for other researchers.

#### 4.6.1 Hypothesis Pruning

Continually extending the existing hypotheses and at the same time generating new ones leads to an ever-growing hypothesis set, which would quickly become intractable. A conservative pruning procedure is used to control the number of hypotheses to be evaluated: (1) hypotheses older
4.6. Implementation Details

than the time window under observation \((t_0 - t)\) are removed, (2) candidates which have been extrapolated through time for too long without finding any new evidence are removed, and (3) candidates which have been in the hypothesis set for too long without having ever been selected are discontinued (these are mostly weaker hypotheses, which are always outmatched by others in the competition for space).

Importantly, the pruning step only removes hypotheses which have been unsuccessful over a long period of time. All other hypotheses, including those not selected in recent frames, are still propagated and are thus given a chance to find new support at a later point in time. This allows the tracker to recover from failure and retrospectively correct tracking errors.

4.6.2 Identity Management

While the hypothesis selection framework uses the available information from several time steps in a global view, its explanations are independent at each time step. Thus, object identities are not automatically preserved. If this is desired (e.g., for surveillance scenarios), an additional step is needed to propagate these identities.

In the case that a trajectory gets selected which was generated by the extension step (Section 4.4.2), identity preservation is trivial. In the other case, we compare the selected candidate with the winners from past frames (that already have an ID). If the sets of explained detections \(S_{H_k}\) overlap sufficiently, then the ID is transferred. If however the new trajectory does not match any known trajectory, a new ID is instantiated.

As a criterion for trajectory support, we use:

\[
\frac{|S_H \cap S_{H_k}|}{\min\{|S_H|, |S_{H_k}|\}} > \eta \quad \text{and} \quad k = \arg \max_j |S_H \cap S_{H_j}|. \tag{4.19}
\]

By itself this may seem like a crude heuristic. However, in the context of the presented system, we can choose a very conservative threshold \(\eta\) (say, 50%), because the physical exclusion constraints during trajectory generation and selection ensure that any two trajectories selected at time \(t\) have zero overlap (and hence only one trajectory at any time \(t\) can significantly overlap a reference trajectory from a previous time step).
4.6.3 Trajectory Initialization and Termination

Tracking is started automatically after a few frames as soon as the benefit of a correct trajectory exceeds its cost. The initialization is not constrained to a specific screen region, since such “entry regions” cannot be defined for general scenarios, even less so if the camera is moving. Although several frames are required as evidence for a new track (in our application, 2–3 usually suffice due to information from two cameras), the trajectory is in hindsight recovered from its beginning.

The flipside of automatic initialization is that trajectory termination needs to be handled explicitly. If an object leaves the observed scene, the past detections along its track still exist and may prompt unwanted re-initializations. To avoid this behavior, exit zones are defined in 3D space along the image borders and are constantly monitored.\(^4\) When an object’s trajectory enters the exit zone from inside the image, the object is labeled as terminated, and its final trajectory is stored in a list of terminated tracks. To keep the tracker from re-using the underlying data, all such trajectories are always selected, thus preventing re-initializations based on the same detections through their interaction costs.

4.7 Results

In the following, we apply the tracker to a set of sequences (Linthescher, Bahnhofstrasse, Loewenplatz, Bellevue, City) introduced in Chapter 2.

For testing, all system parameters are kept the same throughout all sequences, except for setup-specific parameters such as camera calibration and height. Another exception is the ground plane prior for the car platform, which we assume to be Gaussian around the measured camera height. Following the discussion of Chapter 3, we will restrict ourselves to the HOG detector due to its superior performance in our application scenario.

\(^4\)The exit zones are automatically shifted for a moving camera setup such that they always correspond to the image borders
Figure 4.8: Single-frame performance comparison between original detector, scene analysis system, and tracking output on 4 sequences.

Quantitative Results. In Fig. 4.8 (a), we evaluate single-frame performance on Seq. BAHNHOFSTRASSE, and compare the described method with its predecessors, as well as alternative approaches. As before with the scene analysis, this is done by comparing generated and annotated bounding boxes and plotting recall over false positives per image (FPPI), Eq. (3.17). A tracker-generated bounding box is counted as correct if the intersection-over-union of the two bounding boxes’ areas is $> 50\%$.

The HOG detector alone, without any scene knowledge, already performs reasonably well (“Raw detector”). Adding the scene analysis stage im-
proves performance by 5–10% (“Bayesian Net”). Adding tracking further improves the reachable recall, but loses performance in the high-precision regime [Ess et al., 2009a]. This is partly an effect of per-frame evaluation: the tracker requires 2–3 detections to initialize a trajectory (losing recall), and it does report people while they are occluded and hence not annotated (losing precision). To compensate the latter effect, we also plot the detection rates after removing pedestrians who are occluded according to the estimated 3D state (“Tracker”). The performance of the monocular system [Leibe et al., 2007b] using ISM and no depth information, is very poor, mainly because of the high number of false alarms produced by the ISM pedestrian detector. Scene knowledge cleans up many of the false alarms, hence our previous ISM-based system with depth reasoning [Ess et al., 2008] reaches 55% at 1 FPPI. To assess the benefit of the multi-hypothesis approach, we also reduce our system to a first-order Markov tracker, by running only the extension step without any model selection (on the HOG detections), and initializing new trajectories from unassigned detections. This emulation, similar to the raw detector, reaches 63% recall, clearly showing the advantage of explicit multi-frame space-time reasoning.

On the same sequence [Zhang et al., 2008] report 70% recall at 1 FPPI. While they do not use stereo data, their approach is a batch process (requiring the detections of the entire video sequence) and can thus use future observations to correctly handle occlusions. Our online system performs comparably with 73% recall at 1 FPPI. Also using stereo data, [Bajracharya et al., 2009] report 58% recall on this sequence at 1 FPPI (and 42% recall on Seq. LINTHESCHER, see below). In Fig. 4.8 (b–c), we compare single-frame performance of the tracker with the raw detections and the scene analysis system on three further sequences. Again, tracking suffers from the latency of trajectory initialization (this effect is more pronounced for Seq. LOEVENPLATZ, which contains many briefly visible pedestrians), and from “false” detections when keeping track of objects during occlusion. However, only the tracking stage can provide the necessary temporal information for motion prediction and dynamic path planning. The blue curves in Fig. 4.8 (b–c) show the performance on all annotated pedestrians. When only considering the near and mid range up to 10/15/30 m distance (depending on the platform and driving speed), performance is considerably better, as indicated by the red curves.
### 4.7. Results

In Tab. 4.1, we also compare the effect of using different methods for depth-map generation, introduced in Chapter 2. This is of special interest, since nowadays a plethora of stereo algorithms of varying quality and runtime are available. Specifically, we compare a fast GPU-based plane sweeping technique [Cornelis and Gool, 2005] (GPU) with the originally used belief-propagation-based stereo algorithm [Felzenszwalb and Huttenlocher, 2006] (BP), and a high-quality global-optimization approach [Zach et al., 2009] (Zach). On the one hand, computationally intensive algorithms indeed yield an improvement in both scene analysis and tracking performance, but come at the cost of considerably higher runtime (30 ms for GPU vs. 30 s for the others). On the other hand, we are using robust statistics on the estimated depth values, hence top-of-the-line stereo matching does not yield noticeable improvements in system performance, despite producing visually better depth maps.

**Table 4.1:** Detection rates for Seq. Bahnhofstrasse with different stereo matching methods. Better depth maps improve localization, and hence also tracking, in the near field. Fast GPU methods come at the expense of slightly worse performance. Since we use robust statistics on depth, elaborate stereo algorithms bring little improvement.

<table>
<thead>
<tr>
<th></th>
<th>No depth</th>
<th>GPU</th>
<th>BP</th>
<th>Zach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Bayesian Net</td>
<td>-</td>
<td>-</td>
<td>0.63</td>
<td>0.68</td>
</tr>
<tr>
<td>Tracker</td>
<td>0.19</td>
<td>0.29</td>
<td>0.60</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Restricted to 15 m

<table>
<thead>
<tr>
<th></th>
<th>No depth</th>
<th>GPU</th>
<th>BP</th>
<th>Zach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Bayesian Net</td>
<td>-</td>
<td>-</td>
<td>0.66</td>
<td>0.67</td>
</tr>
<tr>
<td>Tracker</td>
<td>0.32</td>
<td>0.47</td>
<td>0.66</td>
<td>0.74</td>
</tr>
</tbody>
</table>

**Track-level Evaluation.** For a more fine-grained analysis, it is instructive to look at the tracking result at trajectory level. Automatic track-level evaluation of complex scenes is still an unsolved problem. For static scenarios with moderate interaction between agents, publicly available software exists [Bernardin and Stiefelhagen, 2008] that operates on a per-frame basis and in 2D. However, in busy scenarios with frequent occlusions and changing cameras, this will fail as the assignment problem quickly becomes non-trivial. We therefore resort to an
4. Multi-Object Tracking

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Bahnhofstrasse</th>
<th>89</th>
<th>125</th>
<th>0.55</th>
<th>0.15</th>
<th>0.30</th>
<th>0.62</th>
<th>16</th>
<th>9.9/1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq.</td>
<td>Loewenplatz</td>
<td>107</td>
<td>126</td>
<td>0.48</td>
<td>0.25</td>
<td>0.27</td>
<td>1.09</td>
<td>6</td>
<td>9.3/2</td>
</tr>
</tbody>
</table>

**Table 4.2:** Trajectory-based evaluation on Seq. Bahnhofstrasse and Seq. Loewenplatz.

An interactive solution, which first does the obvious assignments between tracking output and ground-truth automatically and then polls the user in case ambiguities arise. In a first step, trajectories are trimmed such that the first element corresponds to the minimum size of 60 px, as used in the previous evaluations. This trimming is only done at a trajectory’s head, as this accounts for the most frequent case of far away people slowly moving towards the camera. The distance between two trajectories $T_i = \{x_t^{(i)}, t\}$ is defined as the distance between their corresponding object positions, robustly\(^5\) averaged over the two trajectories’ lifetime. Formally,

$$d_{ij} = \frac{1}{|T_j \cap T_k|} \sum_{t \in T_i \cap T_j} \min \left\{ \| x_t^{(i)} - x_t^{(j)} \|, d_{max} \right\} . \quad (4.20)$$

Trajectories with no overlap or $d_{ij} \geq \min\{|T_i|, |T_j|\}$ are not matched. Occlusion is not accounted for here, as the annotation does not contain this information.

Using a Hungarian matching on the obtained distance matrix would only yield the best solution, and could not account for possible many-to-many mappings between ground truth and tracker output (on the one hand, a ground truth trajectory can be split between multiple tracklets, especially if an occlusion is inbetween; on the other hand, the tracker output can switch from one ground truth to another). Therefore, we opt to first automatically assign the obvious candidates (i.e., only one possible match) and then get user input for the ambiguous cases, allowing many-to-many relations. The metrics themselves then resemble the ones used in other trajectory-level evaluations ([Wu and Nevatia, 2007a; Li et al., 2009]): as background information, we report the number of ground

\(^5\)The gating at distance $d_{max} = 1 \text{ m}$ is required to be robust against inaccuracies of the ground truth—objects are annotated in 2D, their depth has to be estimated from the bounding box.
truth trajectories (GT) and the number of output tracklets (OT). Then
the fraction of mostly tracked (resp. missed) subjects is reported: each
ground truth subject is classified as either mostly tracked (MT, best
output trajectory covers > 80% of the ground truth), partially tracked
(PT, 1 − MT − ML), or mostly missed (MM, estimate covers < 20% of
the ground truth). Furthermore, we report the average number of false
alarms per-frame (FA), the number of identity switches (ID), as well as
the latency, i.e., the mean and median number of frames it usually takes
to start tracking (LA).

Tab. 4.2 gives the results for Seq. BAHNHOFSTRASSE and Seq. LOEWEN-
PLATZ. In both cases, few false alarms occur. Also, the fraction of severe
failures (“mostly missed”) is relatively low. Still, the fraction of “mostly
tracked” trajectories as well as the mean latency are high. This is mostly
due to the strict annotation: it often happens that a distant pedestrian is
visible for a few frames, then becomes occluded for a long time before be-
coming visible at a much smaller distance, whence he is picked up by the
tracker. Even in the best case, such a pedestrian will produce an identity
switch, since the occlusion lasts too long to associate the two trajectories
before and after it. In the worst case, however, the subject will only be
picked up after leaving the occlusion, and hence be reported as “mostly
missed” or “partially tracked”, with a long initialization latency. This
is confirmed by the mean latency, which is significantly higher than the
median, because the entire track of these subjects before being detected
counts as latency. In Seq. BAHNHOFSTRASSE, 9 out of 76 (MT+PT) ob-
jects suffer from the aforementioned annotation problem and thus have
latencies > 30, severly biasing the mean latency. Furthermore, most
of the “partially tracked” subjects are quite well covered—reducing the
threshold for “mostly tracked” to 70% would increase the corresponding
fraction to 0.72.

For better analysis, a more advanced tool is needed, supporting partial
occlusion in both annotation and evaluation, as well as providing more
automatic feedback. A learned distance function, as used for tracking
in [Li et al., 2009], could help in decreasing the interactivity needed.
Especially for car applications, a differentiation between scales or zones
(e.g. “on the vehicle’s future path”, “off the street”, . . . ) would also be
interesting.
**Figure 4.9:** Exemplary pedestrian tracking results on Seq. Bahnhofstrasse.

**Figure 4.10:** Exemplary pedestrian tracking results on Seq. Linthescher.

**Examples Images.** Fig. 4.9 shows two scenarios from Seq. Bahnhofstrasse. In the first row, a pedestrian gets successfully tracked on his way around a few standing people, and a pedestrian is detected at a distance of 20 m, which is at the limit for the detector and the depth map, *c.f.* Section 2.4.3.

Example tracking results for Seq. Linthescher are shown in Fig. 4.10. Our system’s ability to track through occlusion is demonstrated in the top row: please note how the woman entering from the left temporarily occludes almost every part of the image. Still, the tracker manages to pick up the trajectory of the woman on the right again (in red).
Fig. 4.11 demonstrates the vision system in a car application. Compared to the previous sequences, the viewpoint is quite different, and faster scene changes result in fewer data points for creating trajectories. Still, stable tracking performance can be obtained, also for quite distant pedestrians.
Figure 4.13: Exemplary pedestrian and car tracking results on Seq. Bellevue.

Figure 4.14: Distance distribution for pedestrians (a) and cars (b) on Seq. Bellevue.

Finally, combined pedestrian and car tracking is demonstrated in Fig. 4.12 for Seq. City and in Fig. 4.13 for Seq. Bellevue. Tracking cars is more complicated due to the higher distances, and less reliable car detections, especially with respect to viewpoint. The red backlight detector (Section 2.3.3) helps to a certain extent, but cannot handle side views. In
Seq. CITY, one additional detector was trained on rear views of vans in order to be able to track vans like the one in the top row of Fig. 4.12.

Fig. 4.14 shows the depth distribution of tracked pedestrians and cars for Seq. BELLEVUE. For both classes, a few objects are tracked well outside the working range given in Chapter 2. Especially in the case of cars, many are tracked at 30–50 m, where localization accuracy is between 2–5 m in depth, making the construction of proper trajectories difficult. Still, the system manages to track a fair number of cars in inner city scenarios, properly reasoning about spatial occupancy against pedestrians.

**Computational Efficiency.** For applications in robotics or autonomous driving it is of prime importance to have a computationally efficient tracker. As reported in [Ess et al., 2009a], it is possible to get the proposed method to frame-rate performance, even though it involves an extensive sampling of solutions as well as a model selection step.

The computational cost of the approach can be divided into three factors: the cost of finding trajectories, to build the interaction matrix $Q$, and to solve the actual optimization problem. In Section 4.4 we already showed that finding trajectories can be done in an incremental fashion without losing tracking performance. This incremental generation also allows the caching of the respective entries in $Q$: for extended trajectories, the existing values merely need to be multiplied by a temporal decay factor and summed with the probability of the newly added detection. For newly generated trajectories, new columns and rows have to be added.

The resulting runtimes, averaged over the 1,000 frames of Seq. BAHNHOFSTRASSE, can be seen in Tab. 4.3. On the average, the timing figures are very attractive, allowing tracking at rather high frame rates. For complicated situations, the model selection needs considerably more time. The additional bounds introduced in this work give a considerable improvement and the algorithm could be further controlled as to operate only in a given time window. Such a method might yield slightly worse solutions, but can help to fulfill timing constraints. On the other hand, the tracker is implemented to work in actual time units and is thus able to cope with slight jitter or frame drops.
### 4. Multi-Object Tracking

<table>
<thead>
<tr>
<th>Component</th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histogram calculation</td>
<td>0</td>
<td>30</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Hypothesis generation</td>
<td>0</td>
<td>12</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Matrix building</td>
<td>0</td>
<td>44</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Optimization (original)</td>
<td>0</td>
<td>276</td>
<td>39</td>
<td>25</td>
</tr>
<tr>
<td>Optimization (bound)</td>
<td>0</td>
<td>188</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>Tracking Total</td>
<td>0</td>
<td>274</td>
<td>41</td>
<td>27</td>
</tr>
</tbody>
</table>

**Table 4.3**: Average runtimes for tracking system on Seq. Bahnhofstrasse.

### 4.8 Conclusion

In this chapter, we have shown how to employ a hypothesize-and-test tracking framework for mobile tracking applications. For each frame of a video sequence, the proposed architecture first generates an overcomplete set of hypotheses by starting EKF-based single-target trackers in a space-time volume that encompasses all detections of the current and past frames. To this end, two suitable motion models are employed to support tracking of both pedestrians and cars. In a second stage, the obtained set is pruned to a mutually consistent explanation using model selection. The tracking system is demonstrated on four challenging sequences where it manages to track most of the individuals over long periods of time. Even though complex, the system has a rather small computational cost and is thus amenable to an implementation on a moving observer where computational power is limited due to constraints on space and power.

The experiments corroborate the fact that reliable tracking from a mobile observer in busy urban scenarios is possible. As demonstrated, reasoning over longer time spans and inclusion of physically motivated space-time violations allow for a tracking system that can also handle complex interactions between multiple objects, including prolonged occlusion. Even with such a non-Markovian system, competitive computation times can be achieved. While the system was only demonstrated on footage obtained with moving cameras at a relatively low viewpoint, it is general enough to be used for surveillance or other...
4.8. Conclusion

Applications. Operating in an online, but global fashion should always give at least as good results as a Markovian tracker. If offline application is possible however, other approaches [Zhang et al., 2008; Li et al., 2009] might be more suitable as they also incorporate future information.

Various improvements should allow for even more robust tracking: the motion models are rather simple and might be unable to handle erratic motions, but more importantly, they do not take any information about the scene in account. To this end, we suggest a simulation-based motion model in Chapter 6. For the appearance model, the most obvious extension would be the use of a discriminatively trained classifier to keep objects better apart, especially after longer occlusions. Besides better object detectors, the measurement model could be improved by a closer coupling with the tracker (i.e., going back into voting space such as [Leibe et al., 2007b; Breitenstein et al., 2009]) and incorporating more scene information. Lastly, the model selection is mostly dependent on a good and compact set of hypotheses. Here, ideas from other global trackers, such as [Kaucic et al., 2005; Huang et al., 2008], could be implemented to, e.g., link secure tracklets or prevent unnecessary hypotheses generation by applying methods from information theory that spend more time sampling in complicated areas rather than the obvious ones. Another challenge is the inclusion of occlusion reasoning directly into the model selection.

Together with other scene understanding components that are constantly improving, such a tracking system could become a good basis for applications in autonomous driving or robotics.
Integration & Extensions

So far, we have introduced the basic components used in our mobile vision system: depth estimation, detection, and visual odometry (Chapter 2); scene analysis (Chapter 3); and multi-body tracking (Chapter 4). This chapter deals with two aspects of the resulting vision system: the tight integration of the different modules; and the extension with further modules.

In the first part, the integration of these components is studied in more detail. To ensure robust performance of our final system, we opt for a close interplay between the different vision components, rather than employing them in a “feed-forward” fashion only. As suggested in [Ess et al., 2008; 2009b], we therefore couple the tracking system with both the scene analysis and the visual odometry system. The integration naturally leads to several cognitive feedback loops between the modules. Among others, we propose a feedback connection from the object detector to visual odometry which utilizes the semantic knowledge of detection to stabilize localization. Feedback loops always carry the danger that erroneous feedback from one module is amplified and causes the entire system to become instable. Therefore, the incorporation of automatic failure detection and recovery is important, allowing the system to continue when a module becomes unreliable.

In the second part, the resulting system, i.e. a multi-body tracker that can operate reliably from a moving platform in busy urban scenarios, will be regarded as a building block of its own. This module can then be combined with further components, hence extending the system. In this chapter, we will first explore the extension of the static occupancy maps as introduced in Chapter 2. The obtained augmented occupancy maps
form a possible input to path planning algorithms. In a second extension, we use the multi-body tracker to effectively factorize the state space of an articulated tracking problem, thereby arriving at an articulated multi-body tracker.

This chapter is structured as follows. Related work is presented in Section 5.1. In Section 5.2, we describe a holistic view of the system that is then used in Section 5.3 to describe the actual cognitive feedback channels between the components. Failure detection is then addressed in Section 5.4, wrapping up the integration part of the chapter. Then, two extensions to the coupled system are presented: in Section 5.5, we show how to extend the occupancy map generation by the obtained tracking information, hence making them usable for path planning in dynamic scenarios. In Section 5.6, the articulated multi-body tracking system is introduced.

5.1 Related Work

Integration of multiple modules for the purpose of robust tracking is especially done in robotics, where several middleware platforms have been established over the last few years, such as TheMOOS [Newman, 2007] or the navigation toolkit CARMEN [Montemerlo et al., 2003]. These form the technical basis for inter-component communication, fostering research results such as the ones obtained in the DARPA Grand Challenge or the DARPA Urban Challenge [DARPA, 2007]. With the multitude of sensors and actuators, incorporating modules for localization, mapping, navigation, and tracking, such systems go well beyond what is done in this thesis, where we focus on localization and tracking only. Already for these two tasks, researchers in robotics use more than one sensor. E.g., [Spinello et al., 2009] combine laser with vision for detection and tracking of pedestrians and cars, with self-localization handled by a combination of GPS, inertial measurement unit, and laser. Clearly, with more sensors available, one can make up for the deficiencies of each individual sensor. Thus, a close interplay is often not necessary and sometimes even cumbersome due to the number and size of messages that has to be sent through the middleware between multiple computers.
When using a single sensor such as vision only, however, care has to be taken to account for any sensor peculiarities.

In vision, a considerable share of recent research has focused on independently improving sub-problems such as detection or tracking. Some notable exceptions include work fusing pedestrian detection and tracking, such as [Gavrila and Munder, 2007; Giebel et al., 2004; Wu and Nevatia, 2007a]. Few authors have extended this with visual odometry [Bajracharya et al., 2009; Ess et al., 2008; Leibe et al., 2007a], but mostly as an additional component without strong coupling to the rest of the system. Another strand of research has investigated the combination of scene analysis and detection [Hoiem et al., 2008; Wojek and Schiele, 2008] (see also Chapters 3 and 7). This is however often not directly applicable to autonomous driving scenarios.

With this work, we explore how far one can get with vision alone in the above-mentioned problem of localization and object tracking. This work can later on be combined with more sensor modalities and play an important role in an autonomous system. The integration presented here fuses the modules from the previous chapters, as described in [Ess et al., 2009b].
5. Integration & Extensions

<table>
<thead>
<tr>
<th>Component</th>
<th>Model</th>
<th>Solution</th>
<th>Ch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw detector</td>
<td>Various</td>
<td>ISM, HOG, part-based</td>
<td>Ch. 2</td>
</tr>
<tr>
<td>Object detection</td>
<td>Bayesian network</td>
<td>Belief propagation</td>
<td>Ch. 3</td>
</tr>
<tr>
<td>Object tracking</td>
<td>Minimum description</td>
<td>Multi-branch ascent</td>
<td>Ch. 4</td>
</tr>
<tr>
<td></td>
<td>length (MDL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual odometry</td>
<td>Projective geometry</td>
<td>Structure from motion</td>
<td>Ch. 2</td>
</tr>
</tbody>
</table>

*Table 5.1: Mathematical models associated with the components.*

5.2 Integration

Fig. 5.1 gives a schematic overview of the basic mobile vision system. This figure can be seen as the engineering view of the system. For each frame, the blocks are executed as indicated: first, a depth map is calculated and the new frame’s camera pose is predicted. Then objects are detected. This step also encompasses the Bayesian network that performs single-frame reasoning based on basic detector input, depth, and scene geometry (*c.f.* Chapter 3). The obtained information is then used for stabilizing visual odometry, which then updates the pose estimate for the platform and the detections. Next, the multi-object tracker is run on these updated detections. The whole system is held entirely causal, *i.e.*, at any point in time, it only uses information from the past and present frame pairs. Tab. 5.1 summarizes the components introduced in the previous chapters, along with their employed model and solution strategy.

In the first part of this chapter, we will focus on the actual interaction between these different modules, both in terms of feedback (Section 5.3) as well as failure detection (Section 5.4).

5.2.1 Runtime

The combined system was implemented in C/C++, with some algorithms (preprocessing, visual odometry) operating on the graphics card. The total runtime of the entire system for a single frame, disregarding the detector, is shown in Tab. 5.2. As before, this is calculated on Seq. BAHNHOFSTRASSE, with an average of 7 pedestrians per frame, topping at 13. Preprocessing involves loading the data from disc, as well as debayering
5.3 Cognitive Loops

A key component for a robust system is the close interplay between the different modules. This is examined in the following.

5.3.1 Scene Analysis

For detection and tracking, the Bayesian network is updated as shown in Fig. 5.2, implementing cognitive loops between the two components. The new part regarded in this chapter is a temporal feedback channel, which so far has been ignored in the design of the scene analysis system. Specifically, temporal context is introduced for two variables: the ground plane and the objects. For the ground plane $\pi$, a new dependency on

<table>
<thead>
<tr>
<th>Component</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>22</td>
</tr>
<tr>
<td>Visual odometry</td>
<td>21</td>
</tr>
<tr>
<td>Depth map</td>
<td>83</td>
</tr>
<tr>
<td>Scene analysis</td>
<td>43</td>
</tr>
<tr>
<td>Tracker</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>116</strong></td>
</tr>
</tbody>
</table>

*Table 5.2: Runtime of the entire vision system.*

and undistortion. Most of the time is taken by the probabilistic scene analysis, which could clearly be optimized further (see also Chapter 3). The depth map generation adds another rather constant overhead, depending on the number of selected iterations. In total, the system can reach about 3–4 $\text{fps}$ depending on scene complexity, peaking at 10 $\text{fps}$. Parallelization to multiple machines using an appropriate middleware is possible. The framerate would then be controlled by the slowest part and latency would correspond to the total time given above plus any communication overhead. In our experiments, the detector is run offline. However, current GPU-based implementations (*e.g.* [Wojek et al., 2008; Prisacariu and Reid, 2009]) can reach up to 10 $\text{fps}$.
its previous state $\pi_{t-1}$ is introduced. Specifically, let $\hat{\pi}$ be the result of the previous frame's estimation, transformed into the new camera's coordinate frame. As in Eq. (2.28), let $\mathbf{R} = \mathbf{R}_t (\mathbf{R}_{t-1})^\top$ and $\mathbf{t} = \mathbf{t}_t - \mathbf{R}_t \mathbf{R}_{t-1}^\top \mathbf{t}_{t-1}$ be the corresponding transformation for a point. Then, with $\pi = (\mathbf{n}, \pi^{(4)})$,

$$\hat{\mathbf{n}} = \mathbf{R} \mathbf{n}_{t-1} \quad \quad (5.1)$$

$$\hat{\pi}^{(4)} = \pi^{(4)}_{t-1} - \mathbf{n}^\top \mathbf{t} \quad . \quad (5.2)$$

The transformed ground plane $\hat{\pi}$ is then used directly as a temporal prior,

$$P(\pi | \pi_{t-1}) = \alpha P(\pi) + (1 - \alpha) P(\hat{\pi}) \quad . \quad (5.3)$$

For the objects, we introduce a temporal prior, favoring object hypotheses that are close to trajectories selected in the last frame, predicted to the new timestep. That is, trajectory hypotheses from past frames $H_{t_0:t}$ exert a spatial prior on certain object locations $o_{i,t+1}$, which raises the chance of finding detections there above a uniform background level $\mathcal{U}$. We model this prior as a Gaussian around the predicted object position using the trajectory’s dynamic model $\mathcal{M}$. Thus,

$$p(o_{i,t+1} | \{H_{j,t_0:t}\}) \propto \max \left\{ \mathcal{U}, \max_j P(x_{j,t+1}) \right\} \quad , \quad (5.4)$$

**Figure 5.2:** Bayesian network from Chapter 3, updated with temporal information from tracking and the previous ground plane.
5.3. Cognitive Loops

Figure 5.3: Trajectory estimation of our system with and without cognitive feedback. (Top) A few frames of a difficult sequence, Seq. Crossing 1. (Bottom) Trajectory estimates. As can be seen, the proposed feedback greatly stabilizes the egomotion estimation and leads to improved tracking performance.

where $P(x_{j,t+1})$ is the normal distribution obtained from applying $\mathcal{M}$ to a hypothesis $H_{j,t_0:t}$. In our experiments, the uniform background level is set to 0.75.

In most cases, the result of this integration is minor, mostly producing a smoother ground plane for the visual output of the tracker. For Seq. Bahnhofstrasse, the output of the Bayesian network shows about 1% more recall over most of the detection performance curve when setting $\alpha = 0.3$ as opposed to $\alpha = 0$. As the maximum performance of the input detections cannot be surpassed, both curves eventually merge again towards higher false positive rates, reaching the same recall. Setting $\alpha = 1$ however results in a curve that has consistently $\approx 3\%$ less recall, also proving again that allowing a set of ground planes is beneficial.

5.3.2 Visual Odometry

Even though the visual odometry is not directly part of the model seen in Fig. 5.2, it can be integrated into the loop. In the following, we propose a feedback channel from detection-by-tracking to the visual odometry
system that increases the latter’s robustness by introducing semantic information from the scene.

Standard algorithms for visual odometry assume a predominantly static scene, treating moving objects just the same as incorrect correspondences (Chapter 2). Most systems use robust hypothesize-and-test frameworks such as RANSAC or Least-Median-of-Squares for removing such outliers. Recently, some multi-body Structure-from-Motion systems have been demonstrated on realistic video scenes [Li et al., 2007; Ozden et al., 2007]. However, those remain constrained to rigidly moving bodies such as cars and require a sufficient number of interest points for each model. We show that the use of basic scene understanding can effectively stabilize visual odometry by constraining localization efforts to regions that are likely to be part of the rigid scene.

In order to underline the importance of the proposed integration, consider the scene shown in Fig. 5.3, taken from one of our recordings using CharioBot Mk. II (Seq. Crossing 1). Here, our mobile platform arrives at a pedestrian crossing and waits for oncoming traffic to pass. Several other people are standing still in its field of view, allowing standard VO to lock onto features on their bodies. When the traffic light turns green, everybody starts to move at the same time, resulting in extreme clutter and blotting out most of the static background. Since most of the scene motion is consistent, VO fails catastrophically (as shown in the red curve). This is of course a worst-case scenario, but it is by no means an exotic case—on the contrary, situations like this will often occur in practical outdoor applications.

Spatial binning for feature selection (as promoted in [Nistér et al., 2004; Zhu et al., 2006b]) improves the result in two respects: firstly, spatially better distributed features per se improve geometry estimation. Secondly, binning ensures that points are also sampled from less dominant background regions not covered by pedestrians. Still, the resulting path (shown in blue) contains several physically impossible jumps. Note here that a spike in the trajectory does not necessarily have to stem from that very frame. If many features on moving objects are associated successfully (e.g., on a person’s torso), RANSAC can easily be misled by those a few frames later. Failure detection as described in the upcoming Section 5.4 (in green) reduces spiking further, but is missing the semantic information that can prevent VO from attaching itself to moving bodies.
5.3. Cognitive Loops

![Figure 5.4](image_url)

**Figure 5.4**: Tracking result on Seq. CROSSING 1 with cognitive feedback from tracking to visual odometry.

<table>
<thead>
<tr>
<th>Seq.</th>
<th># Frames</th>
<th>Dist</th>
<th>Standard</th>
<th>w/ Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROSSING 1</td>
<td>220</td>
<td>12 m</td>
<td>30%</td>
<td>40%</td>
</tr>
<tr>
<td>LINTHESCHER</td>
<td>1,208</td>
<td>120 m</td>
<td>27%</td>
<td>33%</td>
</tr>
<tr>
<td>BAHNHOFSTRASSE</td>
<td>999</td>
<td>110 m</td>
<td>39%</td>
<td>41%</td>
</tr>
<tr>
<td>JELMOLI</td>
<td>950</td>
<td>82 m</td>
<td>40%</td>
<td>45%</td>
</tr>
<tr>
<td>CROSSING</td>
<td>840</td>
<td>43 m</td>
<td>12%</td>
<td>32%</td>
</tr>
</tbody>
</table>

**Table 5.3**: Overview of used test sequences (frames, approx. travelled distance), along with average percentage of visual odometry inliers. The cognitive feedback consistently improves the inlier ratio, especially in highly dynamic scenes (CROSSING).

Finally, the magenta line shows the result using our complete system, which succeeds in recovering a smooth trajectory.

The tracking result is shown in Fig. 5.4. Detection performance improves as well: when measuring recall over false positives per image on single detections, recall increases by 6% at 0.5 FPPI when using the cognitive feedback.

While this pedestrian crossing example represents a worst-case scenario for VO, the beneficial effect of the proposed cognitive feedback can also be seen in less extreme cases. For instance, for Seq. LINTHESCHER (see Tab. 5.3), estimated walking speed only spikes 15 instead of 39 times (in 1,200 frames) above a practical upper limit of 3 m/s when using cognitive feedback. This means that the fallback options are used less frequently, and in turn that dead reckoning and hence introduction of drift are reduced.
5. Integration & Extensions

Figure 5.5: Visual odometry and occupancy maps are only based on image parts not explained by tracked objects, i.e. the parts we believe to be static. (a) Original image with detected features. (b) Image when features on moving objects (green) are ignored.

The intuition behind our proposed feedback procedure is to remove features on pedestrians using the output of the object tracker. For each tracked person, we mask out her/his projection in the image. If a detection is available for the person in the current frame, we use the confidence region returned by the object detector. If this region contains too large holes or if the person is not detected, we substitute an axis-aligned ellipse at the person’s predicted position (this procedure is also employed for detectors that do not provide confidence maps). A few example masks are shown in Fig. 5.5.

Given this object mask for a frame, we now adapt the sampling of corners. In order to ensure a constant number of features, we adapt the number of corners to look for in bin $i$ as follows

$$N_i = \frac{N_{\text{org}}(1 - p_o^{(i)})}{1 - \sum_i p_o^{(i)} / n_{\text{bins}}}$$

(5.5)

with $N_{\text{org}}$ the originally desired number of corners per bin, $p_o^{(i)}$ the percentage of occluded pixels in bin $i$, and $n_{\text{bins}}$ the number of bins (in our case $n_{\text{bins}} = 100$). Corners are only sampled from unmasked pixels. If no pixel is occluded, the normal number will be sampled. As soon as one bin contains a number of occluded pixels, the possibly missed corners are
redistributed among the other bins. Even with imperfect segmentations, this approach improves localization by sampling the same number of feature points from regions where one is more likely to find rigid structure. By optimizing the sampling locations, the feedback generally improves the feature distribution and thus also the number of inliers. This can be seen in Tab. 5.3 for several test sequences. The sampling also has practical consequences regarding speed: for RANSAC, the number of iterations $M = \log(1 - p) / \log(1 - (1 - \eta)^3)$ is in our case only controlled by the percentage of expected outliers $\eta$ [Hartley and Zisserman, 2004]. The desired probability $p = 0.99$ of an outlier-free solution and the number of points needed to produce a hypothesis is constant for our problem. In most of our examples, the increased number of inliers translates to about half the necessary samples to be tested.

5.4 Failure Detection

For systems to be deployed in real-life scenarios, failure detection is an often overlooked, but critical component. In our case, ignoring odometry failures can lead to erratic tracking behavior, since tracking relies on correct 3D world coordinates. As tracking is in turn used to constrain visual odometry, errors are potentially amplified further. Similarly, the feedback from object tracking as a spatial prior to detection can potentially lead to resonance effects if false detections are integrated into an increasing number of incorrect tracks. Finally, reliance on the ground plane to constrain object detection may lead to incorrect or missed detections if the ground plane is wrongly estimated. The proposed system relies on the close interplay between all components, so each of these failure modes could in the worst case lead to system instability and must be addressed.

5.4.1 Visual Odometry

To detect visual odometry failures, we consider two measures: firstly the deviation of the calculated camera position from a Kalman filter estimate and secondly the uncertainty (covariance) of the camera position after pose estimation. The former is obtained by running a Kalman filter with
5. Integration & Extensions

(a)
(b)
(c)

**Figure 5.6:** (a,b) For estimating the uncertainty of the camera’s position, moving objects have to be masked out, as otherwise, an erroneously small covariance matrix ensues. (c) Flow diagram for vision system including failure detection and Kalman filter smoothing.

A constant velocity model in parallel to the visual odometry system, with measurements coming from the camera position. The latter is obtained from the pose optimization described in Chapter 2. Thresholds can be set for both values according to the physical properties of the moving platform, i.e., its maximum speed and turn rate. Note that an evaluation of the covariance is only meaningful if based on rigid structures. Moving bodies with well-distributed points could yield an equally small covariance, but for an incorrect position (Fig 5.6 (a)). When dynamic objects are disregarded, the covariance gives a reliable quality estimate for the feature distribution (Fig. 5.6 (b)).

In case of failure, the Kalman filter prediction is used instead of the measurement, all scene points are cleared, and the visual odometry is restarted from scratch. This allows us to keep the tracker running without resetting it. While such a procedure may introduce a small drift, a locally
smooth trajectory is more important for our application. The adapted visual odometry system is shown in Fig. 5.6 (c).

### 5.4.2 Object Tracking

The employed tracking method (Chapter 4) by construction accommodates failure detection and correction. Instead of taking a final decision at each time step and propagating only that decision to the next step, the approach builds upon a model selection framework to optimize tracks over a large temporal window. At each time instant, the tracking module explores a large number of concurrent track hypotheses in parallel and selects the most promising subset. This means that it can compensate for tracking errors and recover temporarily lost tracks.

### 5.4.3 Object Detection and Ground Plane Estimation

These two components are kept stable by the continuous use of additional information from stereo depth as an independent source of information. Depth measurements are employed both to support the ground plane estimate and to verify object detections. Thus, false predictions from the tracking system are corrected. Additionally, environments in which moving platforms can safely travel allow for a strong temporal prior $P(\pi_{t-1})$ to smooth measurement noise.

### 5.5 Dynamic Occupancy Maps

The mobile vision system, as described in the last sections, can be used a building block for further extensions. As a first such extension, we will investigate the creation of dynamic occupancy maps.

For successful path planning in scenarios where multiple independent motions and partial occlusions abound, it is vital to extract semantic information about individual scene objects. Consider for example the scene depicted in the top left corner of Fig. 5.7. When just using depth
Figure 5.7: A static occupancy map (bottom left) can erroneously suggest no free space for navigation, even though space is actually freed up a second later (top right). By using the semantic information from an appearance-based multi-person tracker, we can cast predictions about each tracked person’s future motion. The resulting dynamic obstacle map (bottom right) correctly shows sufficient free space, as the persons walk on along their paths. Colored circles indicate standing persons, cones the area occupied by a walking person within the next second.

Information from stereo or LIDAR, an occupancy map would suggest little free space for driving (bottom left). However, as can be seen in the top right image (taken one second later), the pedestrians free up their occupied space soon after, which would thus allow a robotic platform to pass through without unnecessary and possibly expensive replanning. The difficulty is to correctly assess such situations in complex real-world settings, detect each individual scene object, predict its motion, and infer a dynamic obstacle map from the estimation results (bottom right). This task is made challenging by the extreme degree of clutter, appearance
Figure 5.8: Extended flow diagram including components to allow for the construction of dynamic occupancy maps.

variability, abrupt motion changes, and the large number of independent actors in such scenarios.

However, the mobile vision system introduced in this chapter takes care of the major part of this problem: the actual motion analysis of the platform and the objects in the scene. What thus remains is the application of this component to construct more informative, dynamic occupancy maps. The corresponding extended flow diagram is shown in Fig. 5.8 and is explained below.

5.5.1 Method

The method for the construction of the occupancy map introduced in Section 2.4.4 integrates entire depth maps (including any dynamic objects). For a more informative map, we filter out these dynamic parts, such that only the static part of the scenery will be considered in the integration. As in the cognitive feedback with visual odometry, we use the tracker prediction as well as the current frame’s detections to mask out any non-static parts (Fig. 5.5). The reasons for this are twofold: first, integrating non-static objects can result in a smeared occupancy map. Second, we are not only interested in the current position of the dynamic parts, but also in their future locations. For this, we can use accurate and category-specific motion models inferred from the tracker.
**Figure 5.9:** Precision of the tracker prediction for increasing prediction horizon. Data was recorded at 12–14 fps.

As each object selected by the tracker is modeled by an independent EKF, we can predict its future position at any point in time and obtain the corresponding uncertainty $C$. Choosing a bound on the positional uncertainty then yields an ellipse where the object will reside with a given probability. In our experiments, a value of $99\% \ (3\sigma)$ resulted in a good compromise between safety from collision and the need to leave a navigable path for the robot to follow. For the actual occupancy map, we also have to take into consideration the object’s dimensions and, in case of an anisotropic “footprint”, the bounds for its rotation. We assume pedestrians to have a circular footprint, so the final occupancy cone can be constructed by adding the respective radius to the uncertainty ellipse. In our visualization, we show the entire occupancy cone for the next second, *i.e.*, the volume the pedestrian is likely to occupy within that time. The cone is constructed by drawing a line from the object’s current center to the apexes of the predicted ellipse.

Based on this predicted occupancy map, free space for driving can be computed with the same algorithm as in [Badino *et al.*, 2007], but using an appropriate prediction horizon. Note that in case a person was not tracked successfully, it will still occur in the static occupancy map, as a sort of graceful degradation of the system.
5.5. Dynamic Occupancy Maps

5.5.2 Results

Quantitative Prediction. To assess the suitability of our system for path planning, we investigate the precision of the motion prediction for increasing time horizons. This precision is especially interesting, since it allows to quantify the possible advantage over system modeling only static obstacles. Specifically, we compare the bounding boxes obtained from the tracker’s prediction with the actual annotations in the frame using the intersection-over-union criterion (Eq. (3.17)). The result, plotting \(1 - \text{prec}\) over an increasing prediction horizon, can be seen in Fig. 5.9. As expected, precision drops with increasing lookahead time, but stays within acceptable limits for a prediction horizon \(\leq 1\) s (12 frames). Note that this plot should only be taken qualitatively: a precision of 0.9 does not imply an erroneous replanning every 10th frame, as many of the predicted locations do not affect the planned path. Rather, this experiment shows that for reasonable prediction horizons, the precision does not drop considerably. Some more experiments regarding prediction capabilities are given in Chapter 6.

In future work, such an analysis should be conducted on actual 3D trajectories. This however requires appropriate ground-truth, \textit{e.g.}, from lasers or a static top-down camera that provides better localization of the traffic agents.

Qualitative Results. Example tracking results for Seq. PARADEPLATZ are shown in Fig. 5.10. The operating point for generating those results was the same as the one used in Fig. 5.9 (right). Recorded on a busy city square, many people interact in this scene, moving in all directions, stopping abruptly (\textit{e.g.}, the lady in the first row with an orange box), and frequently occluding each other (see, \textit{e.g.}, the lady in the second row with an orange box). The bounding boxes are color coded to show the tracked identities (due to the limited palette, some color labels repeat). Below each image, we show the inferred dynamic obstacle map in an overhead view. Static obstacles are marked in black; each tracked pedestrian is entered with its current position and the predicted occupancy cone for the next second (for standing pedestrians, this cone reduces to a circle). As can be seen, the integrated system is able to track most of
Figure 5.10: Example tracking results for Seq. PARADEPLATZ. For each image, we show the actual tracking results as well as an overhead view of the dynamic occupancy map.

the visible pedestrians correctly and to accurately predict their future motion.

Fig. 5.11 shows more results for Seq. BAHNHOFSTRASSE. Note that both adults and children are identified and tracked correctly even though they differ considerably in their appearance. In the bottom row of the figure, a man (pink box) walks diagonally towards the camera. Without motion prediction, a following navigation module might issue an unnecessary stop here. However, our system correctly determines that he presents no danger of collision and resolves this situation. Also note how the standing
Figure 5.11: Example tracking results with corresponding dynamic occupancy maps for Seq. BAHNHOFSTRASSE. Note the long trajectories and the tracker’s ability to handle temporary occlusions in complex scenarios.

woman in the white coat gets integrated into the static occupancy map as soon as she is too large to be detected (second row, middle). This is a safe fallback in the design of our system—when no detections are available, its results simply revert to those of a depth-integration based occupancy map.
5.6 Articulated Tracking

As a second extension to the system, we will explore articulated multi-body tracking, i.e., instead of only considering the pedestrians as a mass point, we will try to infer the current body pose. In this form, articulated multi-body tracking was first suggested by [Gammeter et al., 2008].

Again, the extension introduced in this section will be based on our mobile vision system. As a detector, the ISM detector is used, as it provides a rough “top-down segmentation” of the pedestrian. The multi-body tracker first analyzes the scene and recovers individual pedestrian trajectories, bridging sensor gaps and resolving temporary occlusions. As an extension, a specialized articulated tracker is then applied to each recovered pedestrian trajectory in parallel to estimate the tracked person’s body pose over time. The articulated tracker is implemented in a Gaussian Process framework and operates on global pedestrian silhouettes using a learned statistical representation of human body dynamics. The two tracking levels are interfaced through a guided segmentation stage, which combines traditional bottom-up cues with top-down information from a human detector and the articulated tracker’s shape prediction.

While there has been a long history of research in articulated tracking (a good overview can be found in [Forsyth et al., 2006]), a vast majority of those papers focuses on recovering body poses of single persons in simpler environments [Carranza et al., 2003; Deutscher et al., 2000; Urtasun et al., 2005; Ren et al., 2005a; Sidenbladh et al., 2000; Sigal et al., 2004] (notable exceptions include [Andriluka et al., 2008; Lee and Nevatia, 2006; Mitchelson and Hilton, 2003; Ramanan and Forsyth, 2003; Ramanan et al., 2005; Zhao and Nevatia, 2004a]). Although several approaches have demonstrated body pose estimation in static surveillance scenarios [Lee and Nevatia, 2006; Zhao and Nevatia, 2004a], none of those systems addresses the more challenging task of articulated tracking in unconstrained, busy street scenes, where many people overlap and partially occlude each other and where the camera itself may undergo egomotion.

This is by no means a coincidence. Articulated tracking under such conditions is extremely hard, and many factors contribute to this difficulty. Even when only tracking a single person, pose estimation and data association between frames contain significant challenges. Several
state-of-the-art approaches build up articulated models from local parts in a bottom-up fashion [Ramanan and Forsyth, 2003; Ren et al., 2005b]. Those approaches can easily get confused by the abundance of human limbs that are visible in busy street scenes. Other approaches rely on global shape, which is difficult to extract in crowded situations due to clutter, overlap and occlusion, especially when the camera itself is moving [Jaeggli et al., 2007].

When trying to track the articulations of multiple persons at the same time, additional difficulties arise from those persons’ interactions. While algorithms that support multiple hypotheses can in principle deal with several people (see, e.g., [Ramanan and Forsyth, 2003]), they typically do not explicitly distinguish between competing pose hypotheses for a single person in the image and different persons that are simultaneously visible. Also, relations between different subjects, such as temporary occlusion, cannot be modeled that way. A straightforward extension of a probabilistic inference algorithm to multiple subjects with occlusion reasoning requires a joint representation for the state space of multiple subjects [Hue et al., 2002], leading to an exponential increase in computational complexity.

In this section, we propose an approach to overcome those difficulties in a system’s context. The key insight behind this work is that it is not necessary to handle the complexity of multi-person interactions at the level of articulated tracking. Instead, we propose to carry out the global occlusion and multi-object reasoning on a coarser level and to only perform a more detailed articulated analysis on the output trajectories of the higher-level multi-body tracker. This allows us to also impart the articulated tracker with important information from trajectory analysis, such as a person’s 3D walking direction, speed, and the knowledge when a trajectory is occluded. However, even a sophisticated multi-body tracker cannot solve the entire problem. Data association remains a challenging task: especially when multiple persons are walking close to each other, their limbs are often hard to distinguish. We address this issue by providing the articulated trackers with a guided segmentation that incorporates top-down knowledge from human detection. Together with a dynamic shape prediction from tracking, this observation model provides sufficiently precise measurements to support articulated multi-body tracking in very challenging street scenes.
The next section discusses related work. Section 5.6.2 gives an overview of our proposed system. The following two sections then present its different components in detail: the Gaussian Process articulated tracking approach (Section 5.6.3), and the guided top-down/bottom-up segmentation (Section 5.6.4). Finally, Section 5.6.5 presents experimental results.

5.6.1 Related Work

The main challenges of 3D articulated tracking are the high-dimensional search spaces of body poses, multi-modal posterior distributions, and the fact that the images do not provide all the necessary information due to their 2D nature, noise, or (self-)occlusions. Using multiple cameras and a controlled environment, ambiguities can be limited, and accurate 3D tracking results can be obtained [Ren et al., 2005a; Carranza et al., 2003; Sigal et al., 2004]. We focus on realistic scenarios with noise and occlusions, where the scene is observed by a single camera or a small-baseline stereo setup.

Many existing articulated tracking approaches can either be described as model-based generative top-down methods [Sidenbladh et al., 2000; Deutscher et al., 2000], or part-based bottom-up approaches [Ramanan and Forsyth, 2003; Ren et al., 2005b] (see [Forsyth et al., 2006; Moeslund et al., 2006] for a comprehensive survey). The latter typically only allow for pairwise constraints between neighboring body parts in a graphical model of the human body. In order to infer 3D body poses from monocular or binocular image sequences, more powerful holistic prior models of possible 3D poses have been learned in [Sidenbladh et al., 2000; Urtasun et al., 2005].

More recently, several approaches have been proposed that learn the statistical properties of human body motion and the relationship between body poses and their image appearance. They rely on machine learning techniques such as kernel regressors or dimensionality reduction and can be divided into discriminative (e.g. [Agarwal and Triggs, 2006; Sminchisescu et al., 2005]) and generative (e.g. [Lee and Elgammal, 2007; Jaeggli et al., 2007; Navaratnam et al., 2007]) methods. While discriminative approaches lead to more direct inference algorithms, they have
5.6. Articulated Tracking

to deal explicitly with ambiguities of the one-to-many discriminative mapping. Furthermore, they assume that the subject’s 2D image location is known beforehand, which is not a trivial task for challenging multi-person scenarios such as the ones considered here. Generative approaches, on the other hand, suffer from the high dimensionality of the body pose space, which is a problem for both the learning and the generative tracking algorithms. Their performance can however be improved by a suitable dimensionality reduction. [Lee and Elgammal, 2007; Jaeggli et al., 2007] first learn such a low-dimensional pose representation and then model the mappings into the pose and appearance spaces, as well as the pose dynamics, using kernel regressors. [Navaratnam et al., 2007] propose an integrated formulation that obtains a dimensionality reduction in a Gaussian Process framework by estimating a low-dimensional latent space which simultaneously maps into the pose and appearance spaces. In our work, we follow a similar line, but explicitly take into account also the dynamics, which prove to be very important for our application.

While most articulated tracking approaches consider only single persons, several methods have also been proposed for multi-person scenarios. In [Mitchelson and Hilton, 2003], multiple independent articulated trackers are initialized manually on different persons. [Ramanan et al., 2005] also automates this initialization stage by detecting stylized poses for 2D body pose estimation. Several approaches have demonstrated 3D body pose estimation in static surveillance scenarios [Lee and Nevatia, 2006; Zhao and Nevatia, 2004a]. Most directly related to our approach, [Zhao and Nevatia, 2004a] also apply a multi-object tracker to identify individual trajectories and estimate each tracked person’s body poses over time. However, their tracking approach relies on background modeling, and their pose estimation process is based on a coarse discretization of the pose space. In recent work, [Andriluka et al., 2008] propose an articulated pedestrian detector as basis for articulated multi-person tracking. While in this approach the articulation can help solve the data association problem, it is currently restricted to side views and performs tracking only in 2D. In contrast, 3D articulated multi-body tracking from a moving, human-level perspective still remains an open issue.
5.6.2 Overview

Fig. 5.12 shows the schematic layout of our multi-body articulated tracking system. A small-baseline calibrated stereo rig mounted on a mobile platform captures two image streams and passes them on to a human detection module. Based on the obtained bounding boxes and rough stereo depth information, the multi-body tracker finds consistent object trajectories in 3D. Each trajectory is then passed to a single-person articulated tracker (Section 5.6.3), which estimates the person’s 3D articulation based on a learned statistical representation. The estimation is made robust by a guided segmentation stage (Section 5.6.4) that combines the pedestrian detector’s top-down segmentation with bottom-up image cues and a shape prediction inferred from the current state of the articulated tracker. This results, for every frame of the sequence, in one body pose estimate per tracked person, located in 3D world coordinates.

While in our approach, stereo-based depth computation supports the multi-body tracker and contributes to finding the subject’s silhouette (see Section 5.6.4) by setting it apart from the background, the accuracy of the depth information is limited by the small baseline between the cameras and does not allow for further disambiguation of the pose estimates (as would be possible in a true multi-camera setup [Carranza et al., 2003; Sigal et al., 2004; Ren et al., 2005a]). The articulated pose estimation algorithm thus relies on image descriptors that are computed from the subject’s silhouette. We do however take into account both image streams of the binocular sequences, which helps to alleviate problems that are caused by image noise; i.e., when one camera stream is
temporarily corrupted by noise or occlusion, the algorithm can base its pose estimate on the second camera.

The output of the multi-body tracker module is a trajectory for each pedestrian in 3D world coordinates (including the person’s 3D orientation, velocity, and bounding box), as well as the information when the person was occluded. As the articulated tracker is currently only trained on walking people, objects below and above a certain speed threshold are discarded. The unoccluded parts of each remaining trajectory (the “tracklets”) can be processed independently by the subsequent articulated tracking module, which would otherwise become intractable. We want to point out, however, that data association between those tracklets still remains a challenging problem, as the limbs of adjacent persons may easily get confused. Section 5.6.4 therefore introduces a guided segmentation, which combines top-down information from the human detector with bottom-up image cues and which considerably improves the observation process.

5.6.3 3D Articulated Tracking

The employed articulated tracking approach\(^1\) operates on the output of the multi-body tracker and is provided with 3D trajectories and walking directions of the individual pedestrians, where many ambiguities and temporary occlusions are already resolved and accounted for by the previous stage.

The articulated tracking algorithm is based on a learned statistical model of body motions and their appearance. This model follows a generative approach to capture the relationship between body pose and image appearance and is conceptually similar to [Navaratnam et al., 2007]; here we additionally learn a dynamical model and propose extensions that make learning tractable for sizable training sets.

The training data consists of corresponding pairs of body articulations and shape descriptors (silhouettes). The body articulations are represented as a list of spatial 3D body part locations from 20 joints of the human skeleton, as shown in Fig. 5.21. The matching shape descriptors

\(^1\)This is joint work with Stephan Gammeter and Tobias Jäggli, as described in [Gammeter et al., 2008].
Figure 5.13: Overview of the learned model. (a) Two slices of the temporal Markov chain. The arrows show the learned relationships between the variables. The low-dimensional pose representation (“Latent Space”) is learned using LLE. (b) The prediction of the shape $y$ depends on the low-dimensional body pose variable $x$ and the orientation $\omega$, while the body articulation $p$ is only modeled as a function of $x$.

are vectors computed from a detected person’s bounding box, where each entry indicates whether a certain pixel lies on the foreground or on the background. We currently use bounding boxes with a resolution of $45 \times 50$ pixels and apply PCA dimensionality reduction on the resulting 2250-dimensional shape descriptor.

The tracking algorithm operates in a low-dimensional representation of the body poses that is obtained by applying Locally Linear Embedding (LLE, [Roweis and Saul, 2000]) on the data set of body articulations. We then model the reconstruction of the original representation of the articulations, the prediction of the corresponding human shape in image space, and the temporal evolution (dynamics) of the body poses over time using Gaussian Process regression [Lawrence, 2005]. This model is illustrated in Fig. 5.13.

Pose reconstruction. Gaussian Processes (GP) define probability distributions over functions and can be used to model the regression between two variables, in our case the reconstruction from the low-
dimensional pose space $X$ to the original articulation representation $P$. $x \in X$ denotes a specific pose in the low-dimensional pose space. Given a covariance function $k_{rec}(x_i, x_j)$ and a set of training pairs, a posterior pdf over expected reconstructions $P^*$ can be computed for any point $x^*$ in the low-dimensional pose space. Training a GP regression model entails finding good parameters $\beta_{rec}$ of the covariance function (model selection). This can be done by maximizing the marginal likelihood of $P$ with respect to the covariance parameters $\beta_{rec}$

$$P(P|X, \beta_{rec}) = \frac{1}{Z} \exp \left( -\frac{1}{2} \text{tr}(K_{rec}^{-1}PP^T) \right),$$

(5.6)

Here, $X \in \mathbb{R}^{N \times d_x}$ and $P \in \mathbb{R}^{N \times d_p}$ are matrices containing the training data, $N$ is the number of observations, and $d_x$ and $d_p$ are the respective dimensionalities of the appearance and pose data. The covariance matrix $K_{rec} \in \mathbb{R}^{N \times N}$ is a function of the data $X$ and the parameters $\beta_{rec}$ of the covariance function, with elements $K_{rec}^{i,j} = k_{rec}(x_i, x_j)$. We use standard squared exponential covariance functions with independent noise.

$$k_{rec}(x_i, x_j) = \beta_1^{rec} \exp \left( -\frac{\beta_2^{rec}}{2} ||x_i - x_j||^2 \right) + \beta_3^{rec} \delta_{x_i, x_j},$$

(5.7)

where $\beta^{rec} = \{\beta_1^{rec}, \beta_2^{rec}, \beta_3^{rec}\}$. The marginal likelihood of Eq. (5.6) can then be optimized using numerical optimization methods such as scaled conjugate gradient.

**Dynamics.** In addition, our model is able to temporally predict future body poses according to a transition model $p(X_{t+1}|X_t)$. Similarly to Eq. (5.6), the marginal likelihood $P(X|\beta^{dyn})$ is derived for the regression from $X_t$ to $X_{t+1}$ (see [Wang et al., 2006]), and optimized with respect to the parameters of the dynamics covariance function $\beta^{dyn}$.

**Shape prediction.** In contrast to the pose reconstruction, the shape prediction additionally depends on the orientation $\omega$ of the subject with respect to the observing camera (see Fig. 5.13 (b)). In our training data, every body pose has a number of corresponding silhouettes, each viewed from a different angle. This results in $NM$ training examples, where $N$ is the number of poses and $M$ the number of viewing directions in our training database. For the regression model, we thus have to optimize
the marginal likelihood $P(Y|\Omega, X, \beta^{app})$, where $\Omega$ contains the viewing angles of the training shapes. Using a straightforward implementation, the complexity of the GP training algorithm scales with $(NM)^3$, since it involves the inversion of the covariance matrix $K_{app} \in \mathbb{R}^{NM \times NM}$. This is impractical for the large datasets we use. We thus propose a covariance function that allows the covariance matrix to be written as a Kronecker tensor product, reducing the complexity to $O(N^3 + M^3)$ instead of the original $O((NM)^3)$. This can be done by defining the appearance covariance function as a product of a pose covariance function $k_{\text{pose}}(x_i, x_j)$ (e.g., squared exponential) and an orientation covariance function $k_{\text{ori}}(\omega_i, \omega_j)$,

$$k_{\text{app}}(x_i, \omega_i; x_j, \omega_j) = k_{\text{pose}}(x_i, x_j)k_{\text{ori}}(\omega_i, \omega_j) .$$

(5.8)

If for every pose $x \in X = \{x_1 \ldots x_N\}$ there are silhouettes for all possible viewing directions $\omega \in \Omega = \{\omega_1 \ldots \omega_M\}$, then the appearance covariance matrix can be written as

$$K_{\text{app}} = K_{\text{pose}} \otimes K_{\text{ori}} .$$

(5.9)

Complexity can be further reduced by replacing the orientation covariance function with a delta function $k_{\text{ori}}(\omega_i, \omega_j) = \delta_{\omega_i, \omega_j}$. This makes sense during training of the GP regression, because the training samples only involve a number of discrete viewing directions $\omega \in \Omega$. Once the regression parameters have been learned with this additional approximation, the orientation covariance function can then be replaced by one with a larger support (e.g., a Von Mises distribution), in order to allow for interpolations between the discrete viewing directions $\omega \in \Omega$.

**Learning the embedding.** The marginal likelihood of the entire learned model can now be written as

$$P(P, Y, X|\Omega, \beta^{rec}, \beta^{app}, \beta^{dyn}) = P(P|X, \beta^{rec})P(Y|\Omega, X, \beta^{app})P(X|\beta^{dyn}) .$$

(5.10)

Rather than just optimizing the regressors, as done here, Eq. (5.10) could be optimized with respect to the latent positions $X$ as well, where the LLE coordinates serve as an initialization, similarly to [Navaratnam et al., 2007]. This would lead to a multi-set extension of the Gaussian Process Latent Variable Model [Lawrence, 2005] with separate covariance
functions for each of the mappings. However, our experiments suggest
that this does not improve the tracking results; they thus do not justify
the increase in the number of parameters to be optimized from less than
ten to several thousand.

**Articulated tracking.** The articulated tracking algorithm operates
on the output trajectories of the multi-body pedestrian tracker, which de-

135

livers 2D image locations, scales, and orientations of the tracked persons.
Its observations are automatically estimated pedestrian silhouettes, ob-
tained through the guided segmentation procedure of Section 5.6.4. A
particle filter serves as an overall framework, where at time $t$ the body
pose hypotheses $x^i_t$ are propagated in the low-dimensional pose space
according to the learned dynamical model. For each particle $x^i_t$, a shape
$y^i_t$ can be predicted by taking into account the 3D track orientation $\omega_t$,
estimated by the multi-body tracker. The particles are then weighted
with their image likelihoods, obtained by comparing the predicted shape
to the actually observed shape $y^s$,

$$w^i \propto p(y^s_t | \omega_t, x^i_t) = \mathcal{N}(y^s, \mu^i_t, \Sigma^i_t) \quad (5.11)$$

where $\mu^i_t$ and $\Sigma^i_t$ are the mean and covariance matrix of the predicted
shape.

Finally, once the particle filter has been run on all images of a tracklet,
a Viterbi algorithm extracts a smooth and consistent trajectory through
the particle set (note that this can in practice be approximated with a
fixed temporal look-ahead). Again, the transition costs between neigh-
boring states are based on the learned dynamical model. As shown in
Fig. 5.14, manually tuning the variance of the dynamics can easily result
in an overreliance on either dynamics or appearance. Thus, in order to
account for variations in the framerate of the sequence and the walking
speed of the subjects, this step additionally chooses between different
scaling factors of the predicted velocities, *i.e.*, accelerated and slowed-
down variants of the dynamical model. Therefore, graphs with different
transition probabilities are created. The Viterbi algorithm is run on all
these graphs independently, using *unnormalized* values of the weights
to ensure comparability. The dynamic model with the highest corres-
ponding Viterbi output is then chosen. This procedure allows a strict
Figure 5.14: Depending on the chosen variance, dynamics will be enforced more or less. In case of a bad silhouette (middle column), strong dynamics still lead to smooth tracks (top), while weak dynamics tend to produce rather jittery output (bottom). However, even in case of a good observation (right column), strong dynamics might override the data and get out of sync with the actual walking cycle (top), while for weak dynamics, this can allow proper inference. As we cannot be sure of the data quality, we use a slightly modified version of the Viterbi algorithm to overcome this problem.

dynamic model (with low variance), while staying in sync with the actual observations.

5.6.4 Guided Adaptive Segmentation

As an interface between the multi-body tracker and the articulated tracker, we are using a set of automatically estimated figure-ground segmentations for each tracked person. In the majority of previous works [Navaratnam et al., 2007; Sminchisescu et al., 2005; Zhao and Nevatia, 2004a], silhouettes are assumed to be available, and are in practice often obtained using background modeling. Since we are dealing with a moving camera setup, we cannot use this option. Instead, we propose
Figure 5.15: In order to segment a person from the background, an MRF is used to combine top-down cues from the detector’s confidence masks as well as feedback from the articulated tracker; with bottom-up cues from color and depth.

to obtain the segmentations by fusing top-down cues (from the detector and the articulated tracker) with bottom-up image information (from color and stereo depth). Keeping in line with previous work by several authors [Boykov and Lea, 2006; Cremers et al., 2007], the segmentation is formulated as an energy function that is minimized with respect to the foreground/background labeling $C = \{c_0, \ldots\}$ of all pixels.

$$E(C) = \sum_i R(c_i) + \lambda \sum_{i,j \in \mathcal{N}} B(c_i, c_j) .$$

(5.12)

In the above equation, $R(c_i)$ denotes the region term for a base element (either a pixel or a superpixel) with index $i$. This models the cost for assigning element $p_i$ to a specific class $c_i$, in this case either figure or ground. $B(p_i, p_j)$ is the boundary (or regularization) term defined on the neighboring elements of $p_i$, using a given neighborhood $\mathcal{N}_i$. This term is used to encourage a smooth segmentation. Joint probabilities can be transformed into an appropriate energy function by working in log-space. $\lambda$ is the weighting factor between region and boundary terms. As a binary labeling problem, the above function can be minimized efficiently using graph cuts [Boykov and Lea, 2006].

In short, the energy function is transferred to a graph structure, with each base element represented by a node and two additional nodes representing source and sink. The edges connecting source and sink with image
element nodes encode the region term in their weights; the inter-element edges are weighted with the boundary term. The minimal energy of Eq. (5.12) corresponds to the cut through the graph with minimum edge weight, which can in turn be efficiently calculated using the max-flow algorithm (e.g. [Cormen et al., 1990; Kolmogorov and Zabih, 2004]). The minimization yields the binary foreground mask \( y_t^{obs} \). Together with the bounding box position and motion direction from the multi-body tracker, this mask serves as the input for inference in the articulated tracker.

**Cues.** \( R(c_i) \) is based on the top-down segmentation map \( f \) of the detector and the shape prediction map \( \xi = \sum_j w_j^t \mu_j^t \) of the articulated tracker, where \( w_j^t \) is the weight of sample \( j \) and \( \mu_j^t \) is its predicted shape from Eq. (5.11).

\[
R(c_i) = -\log(P\xi(c_i) P_f(c_i)) .
\]

(5.13)

Here, \( P\xi \) and \( P_f \) are the probabilities of a certain label given the segmentation maps \( \xi \) and \( f \) from the articulated tracker and from the detector respectively:

\[
P\xi(c_i) = \begin{cases} 
\xi_i & \text{if } c_i = 1 \\
1 - \xi_i & \text{if } c_i = 0
\end{cases} ,
\]

(5.14)

where \( \xi_i \) is the part of the segmentation map corresponding to base element \( i \). \( P_f \) is defined analogously, based on the detector’s top-down segmentation.

The boundary term \( B(c_i, c_j) \) encodes the belief that region boundaries typically coincide with intensity and depth discontinuities. It is defined on the neighborhood \( N \) and penalizes neighboring pixels with different labels but similar colors \( I_i \) and depths \( D_i \). Due to the delta function \( \delta_{c_i \neq c_j} \) this term vanishes when the neighboring pixels are assigned the same label \( c_i = c_j \).

\[
B(c_i, c_j) = e^{-\frac{|I_i - I_j|^2}{2\sigma_i^2}} e^{-\frac{|D_i - D_j|^2}{2\sigma_d^2}} \delta_{c_i \neq c_j} .
\]

(5.15)

**Weighting factor.** The weighting factor \( \lambda \) between region and boundary term is usually a free parameter in the segmentation problem. However, the region term grows with the area of the segmentation, while
5.6. Articulated Tracking

Figure 5.16: Original images (top) and sample segmentations obtained using our guided segmentation (bottom). If no detection is available, the articulated tracker’s prediction is used in conjunction with bottom-up cues (red silhouettes).

the boundary term grows with the boundary length. To avoid scale dependence, it is thus beneficial to multiply $\lambda$ by the height of the selected object bounding box $h_{BB}$, i.e., $\lambda = h_{BB} \cdot \text{const.}$

Shape prior. By ways of the expected appearance $\xi$, we can incorporate prior knowledge about human shapes from the articulated tracker into the segmentation task, which can effectively complete partial segmentation maps $f$. We can furthermore bridge missing detections by feeding back the tracker’s expectation and combining it with bottom-up information. This is demonstrated in the sequence shown in Fig. 5.16: at first, the segmentation works well, giving a good initialization to the particle filter. In later frames, the detector fails due to missing contrast, but the prediction, along with the depth map, is good enough to obtain a usable segmentation.

Of course, care has to be taken not to reinforce erroneous feedback, which might lead to hallucinated walking cycles. Therefore, the influence of $\xi$ is
kept low as long as a detection is present and its weight is only increased when the trajectory contains holes. Even in those cases, however, the boundary terms usually restrict the segmentation well enough.

5.6.5 Results

Training. For training the articulated tracker, we recorded motion capture data (at 30 Hz) of 6 different people walking at speeds between 3 and 6 km/h. The resulting data set consists of slightly more than 2,000 different body poses, each represented by 20 joint locations (i.e., 60 dimensions). For every body pose, silhouettes of a synthetic 3D person model were rendered for 36 different viewing directions. A three-dimensional LLE of the body pose data serves as the low-dimensional pose space for the GP regression model. The marginal likelihood was optimized with scaled conjugate gradients using the FITC sparse approximation with 200 inducing variables [Snelson and Ghahramani, 2006; Lawrence, 2007].

Segmentation. Before applying the segmentation stage to articulated tracking, we explore its parameters separately on a test set of 236 pedestrian image pairs, for which a ground truth segmentation was obtained manually. We compare recall and precision on a per-pixel basis, averaged over all examples. Note that the ISM pedestrian detector was trained on a separate set. In all experiments, we fix the parameters $\sigma_d = 1$, $\sigma_c = 300$, $\lambda = 0.44$. The prior $\xi$ (Eq. (5.14)) is first chosen uniformly and its value swept between in the range of [0, 1] for the experiments.

In a first systematic experiment, we use superpixels, obtained with the algorithm of [Comaniciu and Meer, 2002], as base elements and compare the resulting segmentation procedure with two baselines. The thresholded top-down segmentation of the ISM detector gives the expected lower limit for the segmentation performance. As a second baseline, we consider an ellipse fit into the bounding box (the sweep is over its size). Both reach an equal-error rate (EER) performance of 75%, but the ellipse reaches a higher recall, as it can cover more pixels. Next, we sum the top-down information over superpixels and consider the region term by itself. As the superpixels are semantically largely correct entities,
5.6. Articulated Tracking

Figure 5.17: (a) Performance plot comparing several baselines when using superpixels as base elements. Note that the range of the plot is from 0.5 to 1. (b) Scatter plot for the EER operating point on the best curve.

Figure 5.18: Examples for incorrect segmentations when using superpixels as base elements. Note that similar fore- and background colors can lead to big superpixels that semantically do not belong together, and hence result in incorrect segmentations.

This experiment delivers a considerable boost in performance to 83%. The maximally reachable performance using the employed superpixel algorithm is indicated by the star. Completing the energy functional by adding the boundary term encourages smooth segmentations and slightly increases performance to 85%. Finally, adding depth information is expected to yield a further improvement, which is however only partially the case. The integration of depth estimates over entire superpixels seems unreliable, presumably since depth discontinuities do not always coincide with segmentation borders. So far, we have only considered
Figure 5.19: (a,b) Different flavors when using pixels as base elements, compared to the best result obtained using the superpixel-based segmentation. Note the changed scale from 0.75 to 1.

By using pixels as base elements, we expect to overcome this limitation. In Fig. 5.19 (a), we compare the previously best superpixel segmentation with a first set of pixel-based approaches: first, we try to combine both worlds by using pixels as base elements, but modeling color features based on the superpixels (“hybrid”). This allows cutting between superpixels, and performs roughly similar as the standard pixel-based segmentation (85.1% EER). Including depth information gives a slight but consistent increase to 85.6% EER. We take this as baseline and conduct further experiments (Fig. 5.19 (b)). When introducing very limited prior knowledge in form of an elliptical prior (replacing the uniform prior in Eq. 5.13), we achieve 86%. One thing we note here is, that while the boundary term encourages smooth segmentations, there might still be a few disconnected regions. A simple means to increase precision is hence to discard outlier regions (regions smaller than 2% of the biggest region’s size) in a postprocessing step, which improves EER performance to 87%.
5.6. Articulated Tracking

Figure 5.20: Examples for the segmentations obtained when using pixels as base elements. For the example images on the left, we also show the original top-down segmentation obtained from the ISM detector. Note how the combination of cues can deal with adverse lighting conditions, textures (striped sweatshirt), and scale changes. Some typical failure cases are shown in the last row.

Finally, we try to increase smoothness by considering a neighborhood of 20 instead of 4 pixels. This leads to a final EER of 88%, a 13% improvement over the baseline. A few sample segmentations, shown in Fig. 5.20, underline this result qualitatively.

In the experiments for articulated tracking, we will therefore use the pixel-based approach with a neighborhood size of 20, information from both color and depth, and the feedback from the articulated tracking substituting the simplistic elliptical prior shown here.
### Table 5.4: Sequences used for evaluating the articulated multi-body tracking system. We report the number of (walking) persons and the ones actually found by the multi-body tracker (MBT).

<table>
<thead>
<tr>
<th>Seq.</th>
<th># Frames used</th>
<th>Pedestrians</th>
<th>Found by MBT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>454</td>
<td>23</td>
<td>20</td>
</tr>
<tr>
<td>Bahnhofstrasse</td>
<td>173</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>Linthescher</td>
<td>242</td>
<td>21</td>
<td>17</td>
</tr>
</tbody>
</table>

**Articulated tracking.** We demonstrate our approach on 3 challenging video sequences showing real-world inner-city scenes (Seq. Static, Seq. Bahnhofstrasse, and Seq. Linthescher). These videos were captured using CharioBot and CharioBot Mk. II. Seq. Static is an additional sequence recorded using CharioBot Mk. II. Note that the low framerate used for recording complicates the articulation reasoning considerably. Tab. 5.4 gives an overview over the sequences used.

Seq. Static (Fig. 5.21) was taken on a busy sidewalk. Even though the camera itself is standing still, traditional background subtraction would be difficult due to small camera shake, as well as passing trams and cars. While most people move sideways, they still appear at different depths and often have a slightly tilted trajectory, which we can account for by tracking directly in 3D. In the sequence’s 454 frames, our system tracks 20 out of 23 people successfully with the multi-body tracker. One of the missed pedestrians runs too fast, and another one is at all times occluded by other persons. In addition to the 20 correct tracks, the system yields two additional tracks that contain errors due to wrongly estimated orientation or scale. Counting each person individually, this amounts to a total track length of 932 frames, where a detection is available in 86% of the cases. We visually inspected all the resulting segmentations and found that 55% of these are well-defined (meaning the entire person is covered) in at least one camera. For the individual cameras, only 41% were well-defined. This underlines the usefulness of a stereo system in such real-world scenarios with frequent occlusions. While these numbers might seem low, we would like to note that the articulated tracker can also operate if only parts of the body are segmented correctly (most importantly, the legs). Based on these segmentations, the system tracks 74 walking cycles, 54 out of which were entirely correct. The remaining 20 cases mainly occur at the end of longer trajectories and are mostly due
Figure 5.21: Articulated multi-person tracking results for Seq. Static. The last row shows a 3D visualization of the estimated world state in the three images of the second row.

to multiple, consecutive bad segmentations or occlusions. Note, however, that the silhouettes generally did not contain enough information to unambiguously recover the arm positions, which additionally differed from our training examples since many people were carrying shopping bags or similar accessories. Example pose estimates of our system are shown in Fig. 5.21.

For the remaining sequences, we show qualitative results in Figs. 5.22 and 5.23. As the multi-body tracker takes care of the mapping between
the world coordinate frame and the local articulated trackers, we can apply our system to scenes captured under significant egomotion. Fig. 5.22 shows an example of such a case, where people enter the scene from several directions and undergo large scale changes. The multi-body tracker restricts the sampling for orientations, we can thus still get acceptable results on such data. A more challenging case is shown in Fig. 5.23. Here, the system has to cope with more extreme scale changes and people moving in many different directions, while following one person through a busy pedestrian zone. In particular movement parallel to the viewing direction is highly ambiguous. Still, the articulation is identified in most cases.
Figure 5.23: Articulated tracking results for Seq. LINthescher. This sequence shows a very challenging scenario with considerable egomotion and many pedestrians entering the visible scene at various distances and from different directions.

AWEAR platform. The AWEAR platform (Chapter 2, [Havlena et al., 2009]) is a wearable platform that is able to record video at 14 fps, with the intention of supporting the user in order to give her/him feedback on possibly dangerous or unknown situations. Equipped with wide-angle lenses, the system has a large field of view that is transformed into a pseudo-perspective image using a non-central cylindrical projection ([Havlena et al., 2009]). Based on the known ground-plane from the first frame, the image is also stabilized. Example articulated tracking using one camera only are shown in Fig. 5.24. We only report the articulation of the closest person in a separate window for better visibility. The three bars next to it denote, from left to right, the multi-body tracker’s confidence (i.e., its score), the articulated tracker’s confidence (from the Viterbi algorithm’s output), and their agreement. Towards
Figure 5.24: Sample scene from the AWEAR platform, with a horizontal field of view of 150°. The articulated tracker’s output for the person in front is shown in the lower right of the figure. The three bars indicate the multi-body tracker’s and articulated tracker’s confidence (blue), as well as their disagreement (red). A red bar indicates low agreement, e.g., low confidence in the articulated tracker due to a bad segmentation (in the first few frames, due to the distance) or an unknown pose (the startling pose towards the end of the sequence).

The end of the sequence, there is a rare event, the pedestrian startling due to an almost-collision with the cyclist. The resulting body pose is unknown to the articulated tracker, which is reflected in its confidence,
denoted by the middle bar. When comparing this confidence with the output of the multi-body tracker, incongruencies can be detected. I.e., in this case, the multi-body tracker is still able to track the person, while the articulated tracker fails to find a suitable explanation. This information could be used for signaling an anomaly to the user. Ideally, if such events occur more often, the system might be even able to learn such behavior and adapt accordingly.

5.7 Conclusion

This chapter explored the integration of the vision modules introduced in the previous chapters, as well as the extension of the resulting mobile vision system.

To obtain the integrated system, the different modules (appearance-based object detection, depth estimation, tracking, and visual odometry) were combined in a graphical model, where information is exchanged using a set of feedback channels. This close coupling proves to be a key factor in improving system performance in crowded urban scenarios. We show that special care has to be taken to prevent system instabilities caused by erroneous feedback. Therefore, a set of failure prevention, detection, and recovery mechanisms was proposed. In future work, other sensor modalities, such as laser or GPS, should be integrated, and the system should be taken to actual test platforms where it has to communicate with other modules such as navigation and mapping. Then it will be interesting to investigate the various new feedback channels that ensue from such a combination.

In a first extension, we then explored how to couple the static occupancy map with the tracking information to obtain a dynamic occupancy map. The unified framework helped in obtaining reliable tracking information for motion prediction, which could then be readily used for augmenting occupancy maps. The resulting maps should provide valuable input for actual path planning algorithms [Macek et al., 2008]. While first image-based experiments showed that the precision of the obtained prediction stays in acceptable limits for planning horizons up to 1–2 s, the applicability eventually has to be tested on an online platform, where both the platform itself and the agents in the scene can interact as opposed to
pre-recorded video streams. Before such tests can be done however, the system needs to be optimized further. Even parallelization of the current system onto multiple machines could yield more than 8 fps (depending on the slowest component, as of now, the scene analysis) as opposed to the current 3–4 fps.

The second extension was an articulated multi-body tracking system. Here, good results were achieved by factorizing the problem into separate tasks of multi-body tracking under occlusion and articulated body pose estimation for individual trajectories. This formulation allowed the articulated tracker to benefit from trajectory-level information about the tracked person’s speed and walking direction, which considerably simplified inference and rendered the problem tractable. We have further presented a way to implement this idea with an articulated tracker based on Gaussian Processes and have shown how the framework can be applied under egomotion with the help of a guided top-down/bottom-up segmentation module. Experimental results confirm the viability of the proposed approach.

Currently, the articulated multi-body tracker is restricted to learned articulations from known actions such as walking and running; in contrast to bottom-up approaches, it cannot recover arbitrary body poses. One way to mitigate this issue would be the use of semi-global models instead of global ones. In addition, the results of our estimation could be used to learn specialized color models for different body parts, which then support more general pose recovery, as, e.g., [Ramanan et al., 2005]. The method for pose inference by [Andriluka et al., 2008] could also be a promising extension, since it is based on local appearance and may enable a more direct interplay between detector and tracker. As an alternative to the employed silhouettes, the use of other general descriptors such as HOG or edgelets seem to be another promising direction of research. In addition, the training set of learned gait dynamics should be enlarged in order to more densely cover the range of different walking styles, including gait modifications when, e.g., carrying luggage items. Finally, a feedback from body pose estimation to the multi-body tracker could be added in order to also improve the detection model by incorporating gait dynamics, similar to [Giebel et al., 2004].
Object tracking has seen considerable progress in recent years, with current systems able to handle long and challenging sequences automatically with high precision. One such system was introduced in the previous chapters of this dissertation. The progress is mostly due to improved object models—either generic appearance models or detectors for specific kinds of objects—and better optimization strategies. One aspect that was hardly explored so far however is the dynamic model, another key component of every tracking approach. Typically, a standard first-order model is used, which does not account for the real complexity of human behavior.

In particular, physical exclusion in space is often modeled only indirectly, by allowing at most one detection to be assigned to a trajectory, while at the same time making sure that detections are sufficiently far apart from each other. In practice this amounts to non-maximum suppression in 2D image space. In situations where full occlusions are common (e.g., in street scenes seen by a street-level observer), such an image-based approach fails to adequately differentiate collisions from occlusions.

We believe that one main problem in this context is the dynamic model, typically a first- or second-order approximation applied independently to each subject, e.g., using an Extended Kalman Filter (EKF). Inspired by work on crowd simulation, we propose a more elaborate dynamic model in this chapter, which takes into account the social interactions between objects (here, pedestrians) as well as their orientation towards a destina-
Figure 6.1: While walking among other people, several factors influence short-term path planning. Smoothness of motion, intended destination, and interactions with others limit one’s choice of direction and speed. In the example (overhead view of the same scene, two pedestrians’ perspectives), blue indicates good choices for velocity, red signals “no-go”s. The white cross shows the actually chosen velocity. In this chapter, we propose a dynamic model that takes these factors into account.

The fact that people anticipate future states of their environment during path planning, rather than only react to others once a collision is imminent, has largely been ignored in the literature. This goes to the extent that standard motion models do not even take into account the elementary fact that people have a destination and hence steer back to their desired direction after deviating around an obstacle.

The proposed model, termed Linear Trajectory Avoidance (LTA), is designed for walking people with short-term prediction in mind. Due to the complexity of human motion patterns, longer prediction horizons become unreliable; very short ones do not require sophisticated models, since displacements are so small that linear extrapolation is sufficient. Hence, the effect of LTA is best seen in busy scenarios with frequent short-term occlusions, or when framerate is low and the data association procedure is less reliable.

The model (Section 6.2) operates in physical world coordinates and can be applied to any tracker which operates in a metric frame, such as the tracker of our mobile vision system. We show how the model parameters

---

1This is joint work with Stefano Pellegrini, as described in [Pellegrini et al., 2009].
can be learned from bird's eye view data (Section 6.3), and apply it both in a simple patch-based tracker operating on oblique views, and in a mobile vision system operating on footage from a moving camera (Section 6.4).

6.1 Related Work

Multi-target tracking. In recent years, object tracking has been successfully extended to scenarios with multiple objects [Ess et al., 2009b; Huang et al., 2008; Leibe et al., 2008b; Okuma et al., 2004]. Modern systems can track through long and challenging sequences with high precision. To this end, researchers have focused on improving the appearance model [Grabner and Bischof, 2006; Song et al., 2008], the object detector [Andriluka et al., 2008; Dalal and Triggs, 2005; Felzenszwalb et al., 2008; Wu and Nevatia, 2007a], and/or the optimization strategy [Kaucic et al., 2005; Leibe et al., 2008b; Li et al., 2009; Zhang et al., 2008]. Others have developed approaches specifically for crowded scenes [Ali and Shah, 2008; Brostow and Cipolla, 2006; Zhao and Nevatia, 2004b].

The dynamics and interaction between targets is much less explored. Several models include the requirement that the tracked objects should not collide in any frame. The condition is met by assigning every object detection to at most one tracked object [Huang et al., 2008; Okuma et al., 2004; Wu and Nevatia, 2007a]. Note that the unique assignment alone does not solve the problem for finite object size and finite framerate: detections are not guaranteed to be far enough apart to prevent collisions—one has to rely on non-maximum suppression in image space. Furthermore, there are valid assignments which give rise to crossing paths with a collision between adjacent frames.

In our multi-body tracker (Chapter 4), we explicitly model physical exclusion between subjects in world coordinates. However, this is restricted to the selection of the best trajectory hypotheses only—the important step of creating these hypotheses is done independently and does not take into account interactions.

Besides interactions, one important factor in our model is the desired direction of a subject by the way of goal points. Such points have been
used to influence tracking [Ali and Shah, 2008; Huang et al., 2008; Kaucic et al., 2005]. We directly include target points in our dynamic model.

**Social behavior models.** Modeling the behavior of pedestrians has been an important area of research mainly in evacuation dynamics and traffic analysis. Pedestrian behaviors have been studied from a crowd perspective, with *macroscopic* models for pedestrian density and velocity. On the other end of the spectrum, *microscopic* models deal with individual pedestrians. One example for the latter is the *social force* model [Helbing and Molnár, 1995], where pedestrians react to energy potentials caused by other pedestrians and static obstacles through a repulsive force, while trying to keep a desired speed and motion direction. The model has been shown to well reproduce collective effects encountered empirically [Helbing and Molnár, 1995]. Another branch of microscopic models assumes *agents* that interact autonomously through a basic form of intelligence represented by a rule set [Klügl and Rindsfüsser, 2007; Penn and Turner, 2002]. In yet another branch, cellular automata are used, which discretize the space and select the next desired direction from a preference matrix, *e.g.*, [Schadschneider, 2001].

All these models have been designed and used for simulation purposes. This is also the case for the example-based model of [Lerner et al., 2007], although in this work the simulation is used for synthesizing computer graphics videos.

We are only aware of three works which use a pedestrian model in computer vision applications. [Ali and Shah, 2008] use the cellular automaton model atop a set of scene-specific “floor fields” to make tracking in extremely crowded situations tractable. In contrast, we model single pedestrians in world coordinates, which decouples the approach from the camera setup. [Antonini et al., 2006] propose a variant of the Discrete Choice Model to build a probability distribution over pedestrian positions in the next time step, assuming that all subjects perform a global optimization for the next step. Very recently, [Mehran et al., 2009] use the social force model to detect abnormal behavior in crowded scenes.

Our LTA model shares some characteristics with the *social force* model [Helbing and Molnár, 1995], but differs in two crucial ways: first, rather than modeling the pedestrians at their current location as energy poten-
tials, we predict their expected point of closest approach, and use that point as the driving force for decisions. Second, when simulating a subject, we make it move in the optimal direction instead of just applying a gradient-dependent force. Hence, in LTA pedestrians exhibit decisive behavior and choose their path such as to minimize collisions, rather than just being reactive particles.

The proposed model is aimed at medium density environments where agents can still be detected by their local appearance or by a state-of-the-art pedestrian detector, but are still in frequent interaction, including occlusions over longer periods of time.

6.2 Modeling Social Behavior

Given a current configuration $S = \{s_i\}$ of subjects ($i = 1 \ldots n$), our model estimates the velocity of each $s_i$ in the next time step, based on current positions and velocities for all the subjects. Specifically, we model a subject as $s_i = (p_i^t, v_i^t)$, where $p_i^t$ denotes its 2D position on the ground plane and $v_i^t$ its velocity vector at time $t$. For brevity’s sake, we define the current time step as $t = 0$ and drop the corresponding superscript, e.g., $p_i = p_i^0$. In the following, we will first concentrate on the basic case of two subjects before generalizing to an arbitrary number.

We assume a first-order model jointly for all pedestrians in the scene: every pedestrian knows the current positions and velocities of all subjects.\(^2\) It is thus reasonable to think that each pedestrian will predict the movement of the other pedestrians following a constant velocity model. Therefore, if subject $s_i$ proceeds with the velocity $\tilde{v}_i$, it expects to have the squared distance $d_{ij}^2(t)$ from $s_j$ at time $t$:

$$d_{ij}^2(t, \tilde{v}_i) = \|p_i + t\tilde{v}_i - p_j - tv_j\|^2,$$  \hspace{1cm} (6.1)

where we have made explicit the dependence of the $d_{ij}$ to $\tilde{v}_i$ to highlight that we are taking the perspective of $s_i$ (without loss of generality). Defining $k_{ij}^t = p_i^t - p_j^t$ and $q_{ij}^t = \tilde{v}_i - v_j^t$ we can rewrite Eq. (6.1) as

$$d_{ij}^2(t, \tilde{v}_i) = \|k + tq\|^2,$$  \hspace{1cm} (6.2)

\(^2\)In the actual formulation, this is then restricted to a maximum distance and a viewing cone.
Figure 6.2: Two subjects, with their current directions (black) and velocities (magenta). $s_1$ feels the repulsion from $s_2$’s expected point of closest approach $c_2$, and vice versa. Colors denote energies for different velocities, white dots mark the respective minima. Note how $s_2$ accelerates and turns right in order to avoid $s_1$, while $s_2$ slows down and turns to his right.

We assume that pedestrians try to steer clear of collisions. As $s_i$ has an estimate for $s_j$’s velocity from the last time step, it will adapt its own velocity $\tilde{v}_i$ such that the minimum distance $d_{ij}^2$ from $s_j$ is greater than a certain value that $s_i$ considers comfortable. The minimum distance occurs at the time of closest approach $t^*$, where

$$t^* = \arg \min_{t>0} d_{ij}^2(t, \tilde{v}_i),$$

(6.3)

and we constrain the search to future time steps. Relaxing this constraint for a moment, the time at which the distance is minimized is found by setting the derivative of $d_{ij}^2$ with respect to $t$ to zero,

$$\frac{\partial d_{ij}^2(t, \tilde{v}_i)}{\partial t} = 2q^\top (k + tq) = 0$$

(6.4)

$$\Rightarrow t^* = -\frac{k \cdot q}{\|q\|^2}.$$  

(6.5)

In Eq. (6.4), the distance $d_{ij}^2$ decreases for $t < t^*$ and increases for $t > t^*$. We can therefore reintroduce the constraint, saying that if $t^*$ is
smaller than zero, then the minimum of \( d_{ij}^2 \) for \( t \geq 0 \) will be at \( t = 0 \). Substituting Eq. (6.4) into Eq. (6.2) then yields the minimum distance

\[
d_{ij}^*(\tilde{v}_i) = \| k - \frac{k \cdot q}{\|q\|^2} q \|^2 . \tag{6.6}
\]

Note that Eq. (6.6) does not depend on time anymore. In order to make sure that \( s_i \) avoids \( s_j \), one could set Eq. (6.6) equal to some preferred distance. However, this does not extend well to the case of multiple pedestrians. We therefore propose to build an energy functional for the interaction between \( s_i \) and \( s_j \) as a function of \( d_{ij}^* \),

\[
E_{ij}(\tilde{v}_i) = e^{-\frac{d_{ij}^*(\tilde{v}_i)}{2\sigma_d^2}} , \tag{6.7}
\]

where \( \sigma_d \) controls the distance to the subject to be avoided. \( E_{ij} \) is maximal when the linear trajectories would lead to a collision, and is minimal as \( d_{ij}^* \) goes to infinity.

Based on Eq. (6.7), the influence of multiple subjects can now be modeled as a weighted sum, where each subject \( s_r \ (r \neq i) \) gets assigned a weight \( w_r(i) \) depending on its current distance and angular displacement \( \phi \) from \( s_i \). We set

\[
w_r(i) = w_r^d(i)w_r^\phi(i) \quad (6.8)
\]

\[
w_r^d(i) = e^{-\frac{\| k_{ir} \|^2}{2\sigma_w^2}} \quad (6.9)
\]

\[
w_r^\phi(i) = \left( (1 + \cos(\phi))/2 \right)^\beta \quad . \tag{6.10}
\]

\( \sigma_w \) defines the radius of influence of other objects, \( \beta \) controls the “peakiness” of the weighting function used for the field-of-view. The overall interaction energy for subject \( s_i \), \( I_i(\tilde{v}_i) \), is then given by

\[
I_i(\tilde{v}_i) = \sum_{r \neq i} w_r(i)E_{ir}(\tilde{v}_i) \quad . \tag{6.11}
\]

These interactions alone, however, do not bound the minimization appropriately because scene knowledge is ignored. Like in other works [Ali and Shah, 2008; Johansson et al., 2007], we assume that each pedestrian walks towards a destination \( z_i \), and in doing so tries to maintain a de-
sired speed $u_i$. These two components can be represented by two further energy potentials,

$$ S_i(\tilde{v}_i) = (u_i - ||\tilde{v}_i||)^2 \quad (6.12) $$
$$ D_i(\tilde{v}_i) = -\frac{(z_i - p_i) \cdot \tilde{v}_i}{||z_i - p_i|| ||\tilde{v}_i||} \quad (6.13) $$

The overall energy for subject $s_i$ can hence be written

$$ E_i(\tilde{v}_i) = I_i(\tilde{v}_i) + \lambda_1 S_i(\tilde{v}_i) + \lambda_2 D_i(\tilde{v}_i), \quad (6.14) $$

with $\lambda_1$ and $\lambda_2$ controlling the influence of the two regularizers. See Fig. 6.1 and Fig. 6.2 for a visualization of the obtained energies. Minimizing this distance with respect to the velocity $\tilde{v}_i$ cannot be done in a closed form. In our experiments we employ gradient descent with line search.

Given the situation of a pedestrian facing a group of people, an interesting outcome emerges from Eq. (6.11) and Eq. (6.14). Fig. 6.3 shows the energy that a subject $s_1$ sees when trying to avoid two oncoming pedestrians, $s_2$ and $s_3$. Each column of the figure describes the energy for a different direction of the velocity vector (keeping the speed fixed), while each row indicates a different distance between $s_2$ and $s_3$. One can see that as a consequence of the Gaussian shape, a local minimum in the middle exists only when the gap between the two oncoming subjects is sufficiently large. As the gap narrows, the two people form a local maximum that $s_1$ will try to avoid.

The minimization of the energy functional allows for the calculation of the next desired velocity $\tilde{v}_i^\ast$. However, due to inertial constraints, the subject has to undertake a transition from the current velocity to the desired one. This is modeled through a simple filtering approach. The subject’s position is updated according to

$$ p_i^{t+N} = p_i + (\alpha_N v_i + (1 - \alpha_N)\tilde{v}_i^\ast) t_N \quad , \quad (6.15) $$

where the prediction interval $N (t_N, \alpha_N)$ is made explicit to allow for the adaptation to different frame rates, and $\alpha$ is a mixture coefficient. Naturally, as $N$ grows the prediction becomes more linear. We keep the time interval $N$ at the frame rate of the respective sequence and recompute the desired velocity at each time step.
6.2. Modeling Social Behavior

Figure 6.3: Energy seen by subject $s_1$ when making a choice of changing its heading (horizontal axis) as it approaches two subjects moving in opposite direction. The wider the gap between $s_2$ and $s_3$ (vertical axis), the easier it is to pass between them (bottom of graph, minimum in middle) instead of steering around the pair (top, minima on the side).

6.2.1 Static Obstacles

So far, we only took dynamic obstacles in the form of pedestrians into account. In most common scenes however, people will also try to avoid static obstacles. Following other authors [Johansson et al., 2007] we model such obstacles as subjects with zero velocity. The obstacle’s position is approximated at every time step by the point closest to the pedestrian [Ali and Shah, 2008; Johansson et al., 2007]. While being a coarse approximation, this works well except for highly non-convex obstacles.

6.2.2 Application of the Model

Given the current configuration of dynamic and static obstacles at time $t$, we infer the optimal velocity at time $t + 1$ for each subject in turn by minimizing Eq. (6.14) and then applying Eq. (6.15). Once these velocities have been identified for each subject, they are updated in parallel. In the case of tracking, if an observation is available, it is merged with the simulation’s estimate at this point. Note that we do not iterate the simulation in the current time step, assuming that pedestrians base their
immediate path planning only on the past. Also, iterative optimization within one frame may in certain cases oscillate rather than converge (note, oscillations over time can still occur when people walk towards each other—a well-known situation from everyday life).

6.3 Training

The model as defined in the previous section has six free parameters, which need to be learned from training sequences: the standard deviations defining the comfortable distance $\sigma_d$ and the radius of interest $\sigma_w$, the “peakiness” $\beta$ of the subject’s field of view, the importance weights $\lambda_1$ and $\lambda_2$ of the desired speed and velocity, and the update rate $\alpha$. We fix the prediction time step to 0.4 seconds, which is a reasonable horizon for the model to operate.

To train our model, we have recorded two data sets from bird’s eye view and annotated them manually. This gave a total of 650 tracks over 25 minutes. A sample image including annotation can be seen in Fig. 6.4.

In both scenes, goal points were labeled and the desired direction for each subject was set towards the closest goal. The desired speed was set...
6.4. Results

<table>
<thead>
<tr>
<th>$\sigma_d$</th>
<th>$\sigma_w$</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$\beta$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.361</td>
<td>2.088</td>
<td>2.33</td>
<td>2.073</td>
<td>1.462</td>
<td>0.730</td>
</tr>
</tbody>
</table>

Table 6.1: Model parameters obtained from training sequences.

to the mode of the speed histogram over all trajectories. People standing or strolling aimlessly were not used for finding the correct parameters. However, they were still incorporated when simulating other pedestrians.

To find an optimal set of parameters we have experimented with two optimization strategies, namely gradient descent starting from multiple random initializations, and a variant of genetic algorithms (GA). We found that among the returned local optima of the parameter vector, several performed equally well. For the following experiments, we always use the local optimum with the lowest energy (which resulted from the GA optimization). In every iteration, the energy is calculated by simulating each subject in turn, holding the others fixed at the ground truth. The simulation is started every 1.2 seconds along the subject’s path, and continues for 4.8 seconds, similar to [Johansson et al., 2007].

We obtained the parameters given in Tab. 6.1. At first glance, $\sigma_d = 0.36$ looks reasonable, stating that people will not feel uncomfortable with a person more than $3\sigma_d \approx 1$ meter away; $\sigma_w = 2.1$ means that people further away than $\approx 6$ meters do not influence path planning. Note also that $\beta$, being significantly greater than zero, suggests a relevant peak of attention in the center of the field of view.

6.4 Results

To experimentally evaluate the trained dynamic model, we test it in three different settings. First, we measure its mere quality as a predictor, which is, e.g., of interest for path planning in robotics. Then, we apply it inside two tracking methods; a simple patch-based tracker, as well as our mobile vision system.
6.4.1 Prediction

To test the prediction performance of our model, we use annotated data provided by the authors of [Lerner et al., 2007]. The video shows part of a shopping street from an oblique view. We evaluate on a subsequence of about 3 minutes at 2.5 fps containing 86 trajectories annotated with splines. With the same simulation setting used during training (see Section 6.3) this yields \( \approx 300 \) simulations. A homography from image to ground plane was estimated from four manually clicked points on the footpath to transfer image to world coordinates. As destinations we chose two points far outside the left and right image borders, which holds for most subjects.

We compare our model with a simple baseline ("LIN"), that merely extrapolates using the previous velocity, and with a re-implementation of the social force model ("SF") with elliptical potentials [Johansson et al., 2007]. Parameters for the latter are learned using the procedure discussed in Section 6.3. For our LTA model, we explore two possible parameter sets: the first one was trained without interaction term, adding only the drive towards a destination ("DEST"), whereas the other one ("LTA") also caters for interaction among subjects.

**Figure 6.5:** Performance of the LTA model (solid red) against a trained model that uses destinations but no interactions (crossed green), the social force model (dashed blue) and simple linear extrapolation (dashdot black).
As error measure, the average Euclidean distance between predictions and ground truth is measured in each simulation step. The experiments show an improvement of 6% in prediction error for the LTA model compared to SF and DEST, and of 24% compared to the LIN model. A closer look at the distribution of the errors sheds more light on the differences between models. For this purpose, we define a trajectory as correctly predicted when for each timestep of its simulation, the distance from prediction to ground truth lies within a threshold $T$. The curve in Fig. 6.5 shows the result of this analysis, plotting the percentage of the correctly predicted trajectories over varying $T$. At a threshold of 1 meter, $\approx 50\%$ of the trajectories are already correctly predicted using linear extrapolation (LIN). Adding goal-direction (DES) increases the
correctly predicted trajectories to $\approx 63\%$. The SF model performs only slightly better than the DES model. Another $\approx 7\%$ boost is achieved using LTA, reaching a total of $\approx 70\%$.

There are two issues to note here. Firstly, the scene is only moderately crowded and a large part of the trajectories are almost straight. For these, all models give satisfactory results, which washes out the average difference. Secondly, the error distribution of LTA has a light but long tail with a small number of very large errors. These happen when the model in its present deterministic form avoids other pedestrians by walking around the wrong side, see Fig. 6.6 (c). Although from a tracking perspective, bumping into an obstacle is a no less severe failure than passing it on the wrong side, the latter adds twice as large errors and thereby distorts the comparison. A stochastic variant of our model using our dynamic model as a proposal distribution in a particle filter could help here.

### 6.4.2 Patch-based Tracking

To highlight the effect of the dynamic model and compare it to the LIN model, we have implemented a simple patch-based tracker, using the normalized cross-correlation (NCC) as similarity measure. In the first frame a rectangular patch is manually initialized at each person’s location $p_0^i$ as appearance model, and the speed of all targets is initialized to $\|v_i\| = 0$. At each new time step $t$, the target location $p^i_t$ is predicted with the dynamic model, and a Gaussian centered at the prediction gives the location prior $P_{pred}(p) = \frac{1}{Z} \exp \left( - \left( \frac{\|p - p^i_t\|}{2\sigma_{pred}} \right)^2 \right)$. In the surroundings of the predicted location, the squared exponential $P_{data}(p) = \frac{1}{Y} \exp \left( - \left( NCC(p, p_0^i) - 1 \right)^2 \right)$ is employed as data likelihood, and the maximum of the posterior $P_{pred} \cdot P_{data}$ gives the new target location.

This simple tracker was applied to short, interesting sub-sequences of the footpath sequence (non-overlapping with the ones used above). For the dynamic model, we plug in either the LIN (constant velocity) model or our LTA model, leaving the other parameters unchanged. For the LTA model, the desired direction (standing, left-to-right, or right-to-left) is set for each person according to their last displacement, and the desired speed is set to a constant value for all people.
6.4. Results

Figure 6.7: LTA model vs. constant velocity (LIN) model. Selected frames from two tests with the patch-based tracker. Top: When using the LTA model, the pedestrian marked in red is constrained by people walking nearby. The LIN model overshoots when he manoeuvres around an oncoming person and loses track. Bottom: the LIN model for the person marked in red makes a significantly wrong prediction and loses track, whereas the LTA model tries to avoid oncoming people and predicts correctly. Note also how in both examples the persons marked in cyan drift away at the end, because they are not steering towards a target direction.

Tracking was performed at 2.5 fps, leaving 0.4 seconds between consecutive frames. In this scenario with low framerate, multiple interactions, and low data quality, a strong dynamic prior is important to enable tracking at all. As can be seen in the examples of Fig. 6.7, the simple constant-velocity model loses track of several targets, when they pass others and have to adjust their speed and direction. The examples also show how the trajectories fail to swing back without a target direction. On the other hand, LTA successfully tracks all people in the two examples.
6.4.3 Tracking with a Moving Observer

To further demonstrate the versatility of the approach, we apply the LTA model (as learned from bird’s eye view) to tracking from a moving observer. We use the mobile vision system introduced in Chapters 2–5, and plug in both the LIN (*i.e.*, the original EKF-based model) and the LTA models for modeling pedestrian dynamics. Both versions are then evaluated on Seq. BAHNHOFSTRASSE and Seq. LINTHESCHER.

The tracking system generates a set of trajectory hypotheses based on the object detections and a dynamic model, and prunes that set to a minimal consistent explanation with model selection. This pruning relies on the assumption that all actual trajectories are present in the set of hypotheses, thus requiring correct tracking even when no data is available to immediately correct the motion model, mainly during occlusions. Here the LTA model comes into play.

LTA is introduced in the concurrent trajectory extension step (Section 4.4): there, we apply the LTA model for each hypothesis in turn, making them anticipate the other subjects’ movements in order to avoid them. Especially during occlusion, this ensures that blind trajectory extrapolation takes into account other subjects, and increases the chance that a subject’s trajectory leaves the occlusion at the right position, so that tracking can continue correctly. In addition, we can try to match hypotheses that come out of occlusions by accurately simulating their behavior, now given *all* the recorded data for the rest of the scene. This furthermore decreases unmatched hypotheses and therefore ID switches. Static obstacles are added based on the dynamic occupancy map (Section 5.5).

LTA requires a desired orientation and velocity. Assuming very little scene knowledge, we set the desired orientation parallel to the road, pointing in the respective pedestrian’s previous direction. The desired velocity is set to the last measured speed of the hypothesis.

As the tracker builds on a quite reliable set of pedestrian detections, we expect an advantage of the LTA model mainly in case of occlusion. The improvement is therefore bounded by the frequency of occlusion events. Then, LTA’s extrapolation which is constrained by other agents should outperform a standard linear model, thus preventing possible data association problems when the occlusion is over.
Table 6.2: Comparison of the dynamic models for differing data association thresholds based on the CLEAR evaluation metrics.

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Bahnhofstrasse</th>
<th>LIN</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTA</td>
<td>48</td>
<td>42</td>
<td>45</td>
<td>41</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>LIN</td>
<td>55</td>
<td>55</td>
<td>51</td>
<td>48</td>
<td>0.29</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Linthescher</th>
<th>LIN</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTA</td>
<td>31</td>
<td>30</td>
<td>26</td>
<td>25</td>
<td>0.21</td>
<td>0.21</td>
<td>0.20</td>
<td>0.20</td>
<td>0.08</td>
<td>0.09</td>
<td>0.08</td>
<td>0.09</td>
<td>0.08</td>
<td>0.09</td>
</tr>
<tr>
<td>LIN</td>
<td>35</td>
<td>33</td>
<td>31</td>
<td>31</td>
<td>0.21</td>
<td>0.21</td>
<td>0.21</td>
<td>0.21</td>
<td>0.08</td>
<td>0.08</td>
<td>0.09</td>
<td>0.09</td>
<td>0.08</td>
<td>0.09</td>
</tr>
</tbody>
</table>

To quantitatively relate the two approaches with each other, we compare tracking output with annotated ground-truth using the CLEAR evaluation metrics [Bernardin and Stiefelhagen, 2008], which measure ID switches and the percentage of false negative / false positive bounding boxes. In Tab. 6.2, we compare the two dynamic models by varying the threshold on the Mahalanobis distance $d$ used in the data association. The reasoning behind this procedure is the intuition that a larger search area could possibly compensate for the disadvantages of a less accurate prediction. When using LTA, the number of ID switches is constantly lower, while the number of misses and false positives stays about the same. While consistent, the automatic evaluation tends to over-estimate the number of ID-switches with increasing number of occlusion events. For $d = 3$, we thus manually re-counted the ID switches for the two sequences. In the first sequence, using LTA yields 31 as opposed to 36 ID switches with LIN. In the second sequence, these figures are 18 (LTA) and 26 (LIN). Here, many people leave the field of view and enter again, which is always flagged as a new ID by the tracker. Leaving out these “unrecoverable” cases, the last comparison gets down to 10 (LTA) vs. 18 (LIN), a 44% improvement.

Several interesting situations from the two sequences are shown in Fig. 6.8–6.10. The first three columns show the sequence including the occlusion event as tracked by LTA, then two plots in bird’s eye view contrast the results for LTA with those for LIN. Note the ID switches (red arrows), and the missing track in the third example. This last example is especially interesting, because the person in the very front is only detected as
Figure 6.8: Example where LTA improves the performance of multi-body tracking in Seq. Bahnhofstrasse. Bird’s eye views are for the middle frame. Black areas are static obstacles, red arrows mark ID switches, dotted lines show the pre-switch trajectories still being extrapolated—these dissappear after $\approx 5$ frames as they fail to find supporting detections. The man on the left is successfully recovered from occlusion.

a static obstacle in the occupancy map (Section 2.4.4). Nevertheless it influences the man in the striped sweater, who successfully steers around it, whereas LIN looses track.

6.5 Conclusion

In this chapter, we have explored a new, more powerful dynamic model for tracking multiple people in complex scenarios. The LTA model is not dependent on any specific tracker or scene, it merely needs the subjects to reside in a space that allows one to calculate metric distances.

The LTA model takes into account both simple scene information in the form of destinations or desired directions, and interactions between
6.5. Conclusion

(a) Sequence tracked with LTA

(b) LTA (bird’s eye view)

(c) LIN (bird’s eye view)

Figure 6.9: Example where LTA improves the performance of multi-body tracking for Seq. BAHNHOFSTRASSE. Constrained by the oncoming person, both ladies and the oncoming man are picked up again.

(a) Sequence tracked with LTA

(b) LTA (bird’s eye view)

(c) LIN (bird’s eye view)

Figure 6.10: Example where LTA improves the performance of multi-body tracking in Seq. LINTHESCHER. While the man in the front is not detected, he is integrated into the obstacle map, thus constraining the man in the red-black sweater.
different targets. As it operates in world coordinates, the model can be trained offline on training sequences, and then applied elsewhere. We have shown experimentally that the model yields better predictions, and consistently improves tracking performance compared to dynamic models which disregard social interaction. The improvement comes at negligible computational cost (less than 10 ms for a frame with 15 subjects).

When included into our mobile vision system, the number of ID switches could be reduced. This was achieved by integrating the dynamic model into the trajectory extension step, with additional information about static obstacles gained from the occupancy map generation.

We draw attention to an additional lesson learned from the study: a person’s destination is valuable information and should always be used. While this finding is by no means new, e.g., [Kaucic et al., 2005; Huang et al., 2008], we emphasize that it is true even when the destinations are incomplete or inaccurate. We have shown that even roughly guessed target directions help to make more meaningful predictions. This is particularly interesting for the case of mobile cameras, where the destination cannot be learned from continuous observation. In future work, it will still be interesting to use information as, e.g., gained from the algorithms presented in Chapter 7 to further influence the simulation with the knowledge of static obstacles, pedestrian zones (i.e., sidewalk, pedestrian crossings, etc.), as well as entrance and exit points.

In the present state, we do not model groups of people walking together. This would be possible by an extension to the energy potential. A further interesting direction is the stochastic application of the proposed energy functional.
A reliable vision system for robotics and autonomous vehicles has to go well beyond the mere task of object tracking. In order to be able to cast better predictions for path planning or other attentive mechanisms, a more holistic understanding of the scene where the objects are moving is often desirable. So far, this scene analysis was constrained to a simple ground-plane reasoning and an occupancy map inferred from depth map information. Both these parts have been largely missing a semantic notion: which part of the ground plane is the actual road, which part pedestrian crossing or sidewalk? Are the obstacles buildings, poles, trees, or moving objects that the detector could not account for?

In this chapter, we will take a first step in the direction of more advanced scene understanding. For that purpose, we first explore the use of texture classification to assign each image patch a class label. Spatial coherency is obtained by the way of a globally trained conditional random field that can be further extended with information from object detection. As the labeling is still missing more global semantic knowledge, we will refer to it as *intermediate representation*. In a second step, we use this representation as a basis for an image-based system that is able to recognize both the road type (straight, left/right curve, crossing, . . .) as well as a set of often encountered objects (car, pedestrian, pedestrian crossing).

Particularly the latter stage is novel and could complement other modalities, such as GPS-based maps. While such maps abound and have reached an incredible level of accuracy, they can still profit from additional, image-based information. Especially in urban scenarios, GPS reception can be shaky, or the map might not contain the latest detours due to constructions, demonstrations, *etc*. Furthermore, such maps are
static and cannot account for other dynamic traffic agents, such as cars or pedestrians. The obtained image-based information could thus be fused with existing maps and either assist the driver directly (e.g., a pedestrian crossing is ahead: slow down) or help in improving object tracking (e.g., where are possible entrance/exit points for pedestrians or cars?).

This chapter is structured as follows. After reviewing related work in Section 7.1, the basic patch classifier is introduced in Section 7.2. Next, Section 7.3 discusses the random field used for obtaining spatial coherence, also including object detections. Based on either the original or the spatially smoothed labeling, the traffic scene is analyzed using the method presented in Section 7.4. Following that, Section 7.5 introduces the dataset and annotation methodologies used. Section 7.6 showcases the results for the different stages, before the chapter is concluded in Section 7.7.

7.1 Related Work

Scene categorization has been a very active research field over the past years, we are however not aware of any direct application to autonomous driving in urban scenarios. The related work can be roughly split into three areas: local segmentation or texture classification, global image categorization, or the specialized analysis of traffic scenes, as, e.g., in lane finding.

Segmentation. By segmenting an incoming image into meaningful classes, the employed intermediate representation can be thought of as a first level of scene categorization. Several researchers have proposed systems that first divide the image into a set of superpixels (either based on an oversegmentation [Hoiem et al., 2005] or a regular grid [Schroff et al., 2008]) and then use a set of appearance and geometry features to obtain a class label for each image patch [Brostow et al., 2008; Hoiem et al., 2005; Posner et al., 2007; Schroff et al., 2008; Shotton et al., 2008; Wojek and Schiele, 2008]. Most of these works focus on the improvement of object detection [Hoiem et al., 2006; Wojek and Schiele, 2008] or single-image 3D reconstruction [Hoiem et al., 2005; Saxena et al., 2007].
Higher-order MRFs [Kohli et al., 2008; Sturgess et al., 2009; Wojek and Schiele, 2008] can be used to include more global constraints on the labeling, inferred from, e.g., color segmentations or detectors. In all cases, the result of such a procedure is a local labeling. This labeling fails to capture higher-level relationships: we know which pixels belong to the road or might be a building, but it is not immediately evident how these relate with each other, i.e., what the scene in front of the observer actually is.

**Categorization.** Another branch of work is interested in a more holistic interpretation of the image. Fostered especially by data sets such as CALTECH-101 [Fei-Fei et al., 2004] or the PASCAL VOC challenge [Everingham et al., 2008], researchers have proposed several methods to classify an image into categories. The most successful and popular underlying approach is the bag-of-words representation [Fergus et al., 2003; Lazebnik et al., 2006], with some approaches going into concurrent object segmentation and classification [Cao and Fei-Fei, 2007]. It is often based on a visual vocabulary that would need to be retrained for new image sets and does not readily encode spatial relationships. In the proposed two-stage approach, a retraining of the intermediate patch classifier should suffice to account for different lighting and weather conditions; only different street topologies or road marking conventions (e.g., UK vs. rest of Europe) would require retraining of the entire pipeline.

Also aimed at a global image classification, [Oliva and Torralba, 2001] suggest a “spatial envelope” (GIST) that can reliably distinguish between a wide variety of different scene classes such as “coast” or “countryside”, and even between streets and highways. To some extent, GIST also manages to distinguish subclasses of street, as we show in the results section. However, it has difficulties with more local object classes, such as pedestrian crossings.

**Traffic scenes.** Understanding traffic scenarios per se is currently often limited to analyzing low-level features [Wedel et al., 2008] or typical objects, e.g., by object tracking [Ess et al., 2008; Gavrila and Munder, 2007], lane finding [McCall and Trivedi, 2006], free-space computation [Badino et al., 2007], and traffic sign detection [Broggi et al., 2007; Timofte et al., 2009] algorithms.
7. Patch-based Scene Analysis

(a) (b) (c)

Figure 7.1: Given an input image (a), we calculate an intermediate representation based on a patch-wise scene classification (b) into a set of urban texture classes (c).

We are, in contrast, mainly interested in the type of the upcoming road section as well as the presence of a typical set of objects. This requires more higher-level information than a patch-wise segmentation, and needs more specialized information than standard classification approaches in order to deal with the highly similar subclasses and their difficult appearances.

7.2 Patch Classification

As a basis for all other algorithms proposed in this chapter, we will use a local representation that divides the image into a set of patches and assigns each image patch a label corresponding to an urban class (Fig. 7.1). The actual labeling is obtained by applying machine learning algorithms to a given set of texture features. Operating on a local level restricts the number of possible labels to a set of often-encountered textures and thus makes supervised learning tractable.

The method proposed here is largely based on the work of [Wojek and Schiele, 2008], but we further investigate its application to urban scenarios, adding depth features, a global learning algorithm, and support for pedestrians.
7.2. Patch Classification

7.2.1 Preprocessing

The visual features for the patch classifier are computed in CIE $L^*a^*b^*$ color space. $L^*a^*b^*$ is a device-independent color space, where distances in color space are approximately equal to perceptual color distances. It is hence better suited for classification than the standard RGB space with its arbitrary metric distances. As an additional advantage, the luminance information $L$ is separated from the color information $a^*$ and $b^*$.

To ensure a certain invariance to changing illumination conditions, the image is corrected according to a gray-world assumption: operating in urban scenarios, one can assume a largely gray world, hence, the mean color in an image is supposed to be a gray value. In $L^*a^*b^*$ color space, this translates to the requirement that the sum over both the $a^*$ and $b^*$ channels must be equal to 0, which can be obtained by subtracting the mean $a^*$ and $b^*$ value,

\begin{align*}
    a^*_{GW}(x, y) &= a^*(x, y) - m_{a^*} \quad (7.1) \\
    b^*_{GW}(x, y) &= b^*(x, y) - m_{b^*} \quad (7.2)
\end{align*}

As shown in Fig. 7.2, two images from a similar scene but different daytimes have a much more similar color appearance after the correction.
Patch-based Scene Analysis

(a) $8 \times 8$ pixel patches

(b) $16 \times 16$ pixel patches

Figure 7.3: Comparison of $8 \times 8$ vs. overlapping $16 \times 16$ pixel patches.

We adopt this preprocessing step, even though it might fail in scenarios where non-gray colors, e.g., from vegetation, dominate the image.

7.2.2 Features

To facilitate texture analysis, the image is divided into a set of image patches. Traditionally, there are two possibilities for this: while super-pixels obtained by using oversegmentation [Comaniciu and Meer, 2002; Felzenszwalb and Huttenlocher, 2004] can yield smoother boundaries and semantically more meaningful regions, they tend to spill, take some time to compute, and do not ensure temporal consistency.

We therefore choose to divide the image into a regular grid, consisting of patches of size $8 \times 8$ pixels (Fig. 7.3 (a)). This choice comes naturally, as on the one hand, a patch of this size contains enough texture to be classified, and on the other hand, the segmentation is still quite detailed. Furthermore, the Walsh-Hadamard transform, introduced below, requires the patches to be of size $2^n \times 2^n$. If more context is desired, features corresponding to overlapping $16 \times 16$ patches can be added, see Fig. 7.3 (b).

Texture analysis. In order to analyze the texture inside an image patch, we employ the Walsh-Hadamard transform (e.g., [Hel-Or and Hel-Or, 2005]). Similar to the discrete cosine transform, it decomposes an incoming signal into square waves by transforming a vector of length $2^n$ into a vector of the same size.
7.2. Patch Classification

The transformation is achieved by multiplying the vector $v$ with a so-called Hadamard matrix $H$ and a normalization factor,

$$\hat{v} = \frac{1}{2^{n/2}} H_{2^n} \cdot v \quad (7.3)$$

A Hadamard matrix $H$ is a square matrix that satisfies the following two conditions:

- The entries of the matrix are either +1 or −1.
- The rows of the matrix are mutually orthogonal.

For a given size, there exists more than one Hadamard matrix. One example is the so-called Walsh matrix or natural ordered Hadamard matrix $H_N$, which is constructed recursively as follows:

$$H_{N,1} = \begin{bmatrix} 1 \end{bmatrix} \quad (7.4)$$

$$H_{N,2^k} = \begin{bmatrix} H_{N,2^{k-1}} & H_{N,2^{k-1}} \\ H_{N,2^{k-1}} & -H_{N,2^{k-1}} \end{bmatrix} \quad (7.5)$$

In the two-dimensional case, the transformation is first applied to each row and then to each column of the matrix, or vice versa. This leads to the following formula for a matrix $M$ of size $2^n \times 2^n$:

$$\hat{M} = \frac{1}{2^n} H_{2^n} \cdot M \cdot H_{2^n} \quad (7.6)$$

Fig. 7.4 (a) shows the basis of the two-dimensional Walsh-Hadamard transform. Due to an algorithm by [Fino and Algazi, 1976], the naturally ordered Walsh-Hadamard transform can be calculated in time $O(N \log N)$ instead of $O(N^2)$ (Fig. 7.4 (b)).

To arrive at the actual features, the transformation is applied to every color channel. In the experiments presented here, we use the features from both the $8 \times 8$ as well as the overlapping $16 \times 16$ neighborhoods.

**Geometric features.** A simple, yet powerful feature that has been utilized by many authors [Hoiem et al., 2005; Shotton et al., 2006; Wojek
7. Patch-based Scene Analysis

Figure 7.4: (a) Bases of the $8 \times 8$ sequency-ordered Walsh-Hadamard transform. (b) Diagram for the fast computation of the transformation for a vector of length 8. Each node on the left represents a input vector entry, each node on the right its corresponding output.

and Schiele, 2008] is the normalized position of the patch in the image. Especially the $y$-coordinate can serve as a very strong prior for classes like ground or sky. In our work, we do not consider the $x$-coordinate as a feature, as it turned out to be too strong a prior, producing artifacts in situations where the road is not exactly centered in front of the camera.

**Depth features.** As described in Chapter 2, the setups used for testing our system are all equipped with a pair of cameras (in this Chapter, we will only consider the SmartTer platform). Hence, the depth information can be used to add further, complementary cues. We will use two extra features, the median depth of each patch

$$d_{med} = \text{med}_{\text{pixel } p \in \text{patch}} D(p)^{(3)},$$

and the median height of the obtained point over the ground plane $\pi = (n, \pi^{(4)})$,

$$d_h = \text{med}_{\text{pixel } p \in \text{patch}} n^\top D(p) - \pi^{(4)}.$$
As shown in later in the results section, these two features can already help with identifying vertical categories, such as cars or walls. In the future, more 3D features, as also promoted by [Brostow et al., 2008], could be employed.

### 7.2.3 Classifier

We assume that each patch belongs to either one of $C_p = 13$ classes (street, car, . . . , see Fig. 7.1 (c)). The classes were chosen as to reflect the most common textures found in an urban scene. Based on a set of fully annotated images, a discriminate classifier is trained independently in a one-versus-all manner for each class.

Learning is performed using discrete AdaBoost [Freund and Schapire, 1995] for feature selection [Tieu and Viola, 2000]. In general, boosting forms a strong classifier by a linear combination of weak classifiers. The weak classifiers are trained sequentially on a reweighed set of the labeled training data (i.e., weights of misclassified examples are increased and thus the algorithm focuses on the hard-to-learn examples). For feature selection, each weak classifier corresponds to a feature (defined above) and the best performing feature (the one with the lowest error) is chosen and added to the strong classifier. As weak classifier, a decision stump is used,

$$h(y) = \begin{cases} (-1)^a & \text{if } y^{(d)} < t \\ (-1)^{a+1} & \text{else} \end{cases}.$$  \hspace{1cm} (7.9)

The learning algorithm automatically decides for the threshold $t \in \mathbb{R}$, the sorting order $a \in \{0, 1\}$, and the dimension/index $d$. For the patch classifier, $K = 500$ features from the available pool are selected. Given the boosting value $H(y) = \sum_{k=1}^{K} \alpha_k h_k(y)$, a posterior probability can be obtained using the formula

$$P(x|y) = \frac{e^{H(y)}}{e^{H(y)} + e^{-H(y)}}.$$  \hspace{1cm} (7.10)

### 7.2.4 Application

In order to label an image, it is first preprocessed, as described in Section 7.2.1. Then, each patch is processed independently, first calculating
the features and then applying all $C_p$ classifiers. The responses can be interpreted as the probability $P(x|y)$ that the patch corresponds to the class $x$, given the features $y$. These probabilities are used as input for further processing stages. For hard decisions, *e.g.* when showing images, the label of the classifier with the maximum response is chosen.

### 7.3 Spatial Smoothing

An independent classification of each image patch can result in a noisy labeling. It is hence desirable to introduce more context by modeling the dependency between neighboring patches. To do so, we employ a Conditional Random Field (CRF).

A CRF is a stochastical model that directly models the posterior probability as a product of functions over a small subset of random variables $x$. In this case, the subsets are single nodes and pairs of neighboring nodes. The corresponding factors are called *node functions* $\phi$ and *edge functions* $\psi$ respectively,

$$
\Upsilon(x|y) = \prod_i \phi_i(x_i|y_i) \prod_{i,j} \psi_{i,j}(x_i, x_j|y_i, y_j).
$$

(7.11)

The model is called conditional, because these functions may be implicitly dependent on the corresponding feature vectors $y$. When the CRF is instantiated, the features $y$ get fixed and the model is reduced to a Markov Random Field (MRF). The function value $\Upsilon(x|y)$ should then be approximately proportional to the real posterior probability

$$
\frac{1}{Z(y)} \Upsilon(x|y) \approx P(x|y),
$$

(7.12)

where $Z(y)$ is unknown, but only dependent on the features $y$ and thus constant for a specific image.

In our case, a node in the CRF represents an image patch, with edges between neighboring patches. The variables $x$ stand for the class of the patches, while $y$ represents the features defined in Section 7.2. Since the CRF is defined as a product of subfunctions, it can be represented as a factor graph, as depicted in Fig. 7.5. It consists of state nodes
7.3. Spatial Smoothing

Figure 7.5: Factor graph representation of conditional random field: State nodes (black circles) represent the variables, in our case, the label for each patch. The single connected function nodes (blue) are evidence from the patch classifier as described in Section 7.2. The connecting function nodes (red) model the neighborhood between patches. The latter are trained globally.

that encapsulate a random variable to be solved for (in our case, the true label of a patch), and function nodes that encode the relationship between states. Single connected function nodes correspond to the unary potentials and only influence the state of their respective state node. The connecting function nodes correspond to the edge potentials and thus encode neighborhood relations between state nodes. See [Loeliger, 2004] for an introduction to factor graphs.

7.3.1 Training

In order to use the model, node and edge functions need to be defined. With the model defined, what remains is its actual training. Instead of globally training both nodes and edges at the same time, the training will be split into two separate parts.

Nodes functions. For the unary node functions, we will adopt the output of the patch classifier (Section 7.2) as it stands,

\[ \phi_i(x_i|y_i) = P(x_i|y_i) \]
While using the patch classifier from the previous step may lead to the
global optimum not being found, the use of the more sophisticated training
algorithm (AdaBoost) used for the patch classifier should make up
for this disadvantage.

**Edge functions.** The edge functions are chosen independent of the
image, and model the cooccurrence of different texture classes. In this
work, we will consider two different possibilities. The first one is the
frequently used Potts model, that favors same labels next to each other.
Alternatively, the edge functions can be made fully class-dependent to
allow a more flexible model, which however results in higher training
costs.

- **Potts model.** The Potts model rewards neighboring patches of
the same class and punishes different classes,

\[
\psi(x_i, x_j; \lambda) = \begin{cases} 
1 & \text{if } x_i = x_j \\
\exp(-\lambda) & \text{else}
\end{cases}.
\]  

(7.14)

The parameter $\lambda > 0$ is usually set by hand to a reasonable value. Due to its simplicity, the Potts model allows for specialized optimizers (e.g., [Felzenszwalb and Huttenlocher, 2006]).

- **Class-dependent model.** The Potts model does not have any
notion of what classes tend to occur next to each other. We thus
will also use a class-dependent model, where the edge function
 corresponds to a matrix encoding the neighborhood probabilities
of two classes,

\[
\psi(x_i, x_j; \theta_{\psi}) = \theta_{\psi,i,j}.
\]  

(7.15)

For better classification performance, we differentiate between ho-
izontal and vertical edges,

\[
\psi(x_i, x_j; \theta_{\psi}) = \begin{cases} 
\theta_{\psi,h,i,j} & \text{if } (i, j) \text{ is a horizontal edge} \\
\theta_{\psi,v,i,j} & \text{if } (i, j) \text{ is a vertical edge}
\end{cases}.
\]  

(7.16)

The entries of the matrices can be learned from training data, details of the learning algorithm are given in Appendix A.
Note that in contrast to [Wojek and Schiele, 2008], we do not consider the dependence on the features $y$, such as to keep the number of parameters to be optimized low.

### 7.3.2 Inclusion of Objects

While the CRF already introduces a notion of neighborhood, these dependencies tend to be rather local. More semantics can be introduced when including an object detector directly into the formulation. Similar to the scene analysis system of Chapter 3, this then allows joint reasoning about the class labels and correct object detections: one the one hand, a strong object hypothesis can help in correctly labeling the image region as “pedestrian”, on the other hand, the labeling can help in rejecting hypotheses that, e.g., do not stand on the ground.

For modeling a detection inside the CRF framework, we assume a binary variable $o_l \in \{0, 1\}$ for each object detection, indicating the object’s validity. With these new object variables $o$, the posterior probability that has to be maximized can be decomposed as

$$P(x, o|y) = P(o|y)P(x|o, y). \quad (7.17)$$

In the following, these two factors will be described.

**Detector score.** Since the individual objects are considered independent, the first factor of Eq. (7.17) can be further decomposed into the individual probabilities of a object hypothesis being valid,

$$P(o|y) = \prod_l P(o_l|y). \quad (7.18)$$

As in Chapter 3, we use logistic regression on the detector score $s(o_l)$ to arrive at the actual probability,

$$P(o_l|y) = r(s(o_l); \theta_r) \quad , \quad (7.19)$$

where $r(\cdot)$ is the regression function with parameters $\theta_r$. 
Extending the random field. The second factor of Eq. (7.17), $P(x|o, y)$, is very similar to the CRF from the last section. Under the assumption that the detector’s bounding boxes do not overlap, each patch depends only on the object hypothesis in whose bounding box it lies.

For every object hypothesis obtained from the detector, a new state node is created and connected to the state nodes of the patches lying inside the detection’s bounding box. Furthermore, a function node is introduced for every detection, representing the object probability $P(o|y)$ (Eq. (7.19), green node in Fig. 7.6). Through the CRF, the object’s state and the labeling of the patches interact.

For patches inside an object hypothesis’s bounding box, a separate boosting classifier is employed that uses the value of $o_l \in \{0, 1\}$ as additional input feature. The patch classifier can use this additional feature of an object’s validity to, e.g., discern pedestrians from cars. However, unlike the other appearance/geometric features for the patch classifier, $o_l$ cannot be determined from the image itself. Rather, its value will be decided when solving the CRF, based on the detector score and the patch labeling. Still, the function node for $P(x_i|y_i, o_l)$, shown in red in Fig. 7.6, depends on both image features and $o_l$. To build up the function, the boosting classifier is evaluated once for $o_l = 0$ and $o_l = 1$. The dependence on $o_l$ extends to the MRF’s node functions (Eq. (7.13)),

$$
\phi_i(x_i|y_i, o_l) = P(x_i|y_i, o_l)
$$

(7.20)
7.3. Spatial Smoothing

The factor graph corresponding to these functions is depicted in Fig. 7.6. Due to this integrated modeling, we not only expect a better labeling, but also improved detector results, as the model has the opportunity to reject a detection if it has insufficient support from the texture features.

[Wojek and Schiele, 2008] also include object hypotheses into their model, but integrate them into the edge functions. An inclusion into the node functions, as done here, seems more natural, since the detector output can be rather thought of as additional evidence than a relation between patches. Furthermore, thanks to the integration into the boosting classifier, our model also does not need any additional parameters to be trained.

**Additional features.** By adding $o_l$ to the feature-set of patches that lie within a detection’s bounding box, further detector-dependent features can be added to help the patch classifier to assess a detection’s validity. To this end, we add the real-world height of the bounding box to the set by backprojecting the bounding box back into the camera coordinate system at the specific depth of the patch and calculate the height of it. As in Chapter 3, the height is then evaluated under the population density $\mathcal{N}(1.7, 0.085^2)$ [m].

The boosting classifier can then learn connections between the texture features, the real-world height, and $o_l$. Ideally, for the pedestrian-versus-all classifier, the result will be “pedestrian” if $o_l = 1$ and the height is in accordance, and “non-pedestrian” otherwise.

**Training.** When including pedestrians, the patch classifiers have to be retrained accordingly. For this, patches will obtain the ground-truth labeling of $o_l$, based on manual bounding box annotations, as additional input.

**Overlapping detections.** A drawback of this modeling is its inability to handle overlapping bounding boxes. This is due to the fact that the feature vector for one patch has to be constant in size for every patch and can thus not accommodate for a varying number of appended $o_l$. In the present implementation, this problem is circumvented by
selecting the smallest of the bounding boxes for every patch. In this way, big bounding boxes cannot completely occlude smaller ones. As the employed detectors usually perform non-maximum suppression, this problem is not crucial; but it still has to be explored in future work.

7.3.3 Application

For each image of a video stream, the patch classifier is evaluated and the corresponding factor graph (Fig. 7.5) constructed. If object detections are available, they are included into the factor graph as described in Section 7.3.2 (resulting in a graph similar to Fig. 7.6). Loopy belief propagation [Pearl, 1988] with the sum-product algorithm is used for conducting inference over the model. To avoid oscillations, the message passing is done in a randomized fashion. This yields the class probabilities for each patch, as well as the object probabilities, if included.

7.4 Segmentation-Based Urban Traffic Scene Understanding

Another option for using the patch labeling obtained from the previous stage is to infer a basic understanding of the current traffic scene in front of the vehicle. We term this Segmentation-based Urban Traffic Scene Understanding (SUTSU). Specifically, we aim to distinguish 8 different types of road layouts and to detect the presence of cars, pedestrians, or pedestrian crossings in front of the vehicle. Tab. 7.1 gives an overview of the employed classes.

7.4.1 Features

We employ 3 different types of feature-sets to capture the discriminating properties of a traffic scene: rough layout, periodic structures, and orientation histograms. Let $M_c = P(x = c|y)$ be the probability map corresponding to a specific class $c$, i.e., $M_c(u,v)$ contains for every
patch \((u,v)\) the posterior probability that it belongs to class \(c\). Naturally, \(\sum_{c=1}^{C_p} M_c(u,v) = 1\) for a given image patch \((u,v)\). \(M\) and \(N\) denote the number of patches in horizontal and vertical direction, respectively.

**Rough layout.** To get the basic underlying structure of the image, we use a hierarchical representation obtained by downsampling the patch classifier’s probability maps into maps of size \(2 \times 2\), \(4 \times 4\), and \(8 \times 8\) by mean-filtering, yielding \(C_p \cdot (4+16+64)\) features. This is in a sense similar to image pyramids [Grauman and Darrell, 2005], which were shown to be very effective in image classification [Lazebnik et al., 2006].

\[
F_p(c,l,u_l,v_l) = \frac{\sum_{u_l'}(u_l-1)L_u+1 \sum_{v_l'}(v_l-1)L_v+1 M_c(u',v')} {l^2},
\]

with \(l \in \{2,4,8\}\), \(L_u = \frac{M}{l}\), \(L_v = \frac{N}{l}\), and \(u_l = 1 \ldots l, v_l = 1 \ldots l\). However, the higher levels of a hierarchy wash out the spatial information, while learning on the actual segmentation is not invariant to slight perspective changes unless considerably more training data is used. A certain invariance can be achieved by calculating the mean classifier strength of all classes for each row, respectively each column. For \(u = 1 \ldots N, v = 1 \ldots M,\) and \(c = 1 \ldots C_p\), this amounts to

\[
F_v(c) = \sum_{w'=1}^{N} M_c(u',v)/N ,
\]

\[
F_u(c) = \sum_{v'=1}^{M} M_c(u,v')/M ,
\]

yielding another \(C_p \cdot (N + M)\) features. The mean over rows is, e.g., helpful to detect whether an object is in front of the observer disregarding its \(x\)-position, whereas columns can give an idea of the road structure.

**Periodic structures.** With either of the above feature sets, it is difficult to keep a periodic structure like a pedestrian crossing apart from standard road markings. Therefore, we try to measure periodicity using
another feature set that is constructed by again subsampling the patch classifier’s probability maps into either 8 or 16 rows:

\[
M_c^{(8)}(u, v_8) = \sum_{v'=[8v/N]}^{[8(v+1)/N]} M_c(u, v') ,
\]

with \(v_8 = 1 \ldots N/8\), or similarly, with \(v_{16} = 1 \ldots N/16\). Let

\[
A_c(t, v_8) = \sum_u M_c^{(8)}(u, v_8)M_c^{(8)}(u - t, v_8)
\]

be the autocorrelation over columns. The periodicity for each class and vertical stripe is then recorded as the location and strength of the first local maximum,

\[
F_{p1}(c, v_8) = \max_t A_c(t, v_8)
\]

\[
F_{p2}(c, v_8) = \arg \max_t A_c(t, v_8).
\]

Doing the same for both 8 and 16 stripes yields another \(2N \cdot (8 + 16)\) features.

**Orientation.** Lastly, to get an idea of the road direction (straight, curve, \ldots), we measure the orientation of road markings and the curb. Specifically, we apply an orientation operator [Costa et al., 2002] to both the probability maps corresponding to road marking and curb, divide the resulting maps into \(4 \times 4\) regions and create an orientation histograms with 18 bins for each region, giving yet another 576 features.

### 7.4.2 Learning

With one probability map for each of the \(C_p = 13\) classes, and \(N = 80, M = 60\), we obtain a pool of 6,368 features. As with the patch classification, we use boosting for feature selection to select the most important features from the available pool for each class. Again, the classifiers for each road type/object are learned independently using a one-versus-all training scheme, and 200 features are selected.
7.5. Dataset

We use two challenging data sets recorded using the SmartTer platform in Zurich. We will use the left camera’s output, but in some cases also the depth maps generated from the stereo pair using the algorithm of

<table>
<thead>
<tr>
<th></th>
<th># Frames</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Day</strong></td>
<td>22,500</td>
<td>9,697</td>
<td>2,313</td>
<td>2,254</td>
<td>621</td>
<td>852</td>
<td>2,735</td>
<td>1,701</td>
</tr>
<tr>
<td><strong>Dusk</strong></td>
<td>15,000</td>
<td>7,512</td>
<td>1,373</td>
<td>1,568</td>
<td>186</td>
<td>248</td>
<td>999</td>
<td>641</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># Frames</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Day</strong></td>
<td>22,500</td>
<td>8,424</td>
<td>3,924</td>
<td>5,398</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dusk</strong></td>
<td>15,000</td>
<td>3,300</td>
<td>5,478</td>
<td>2,601</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 7.1:** Distribution of classes in the employed sequences (number of frames containing a specific category). At 13 fps, this corresponds to roughly 50 (29+19) minutes of data.

7.4.3 Temporal Smoothing

Working from video, it is sensible to use some sort of temporal smoothing for the classifiers’ output. We opt for a Hidden Markov Model (HMM, c.f., e.g., [Rabiner, 1989]). To allow an online application of the algorithm, we report only the output of the forward pass in our experiments. For the road type, a single HMM with 8 states corresponding to the respective type is used, where the transition probabilities between the different types are learned from the training set. For each object classes, one independent HMM is used, as the object’s occurrence is not directly coupled with the road type or the other objects. Again, the respective transition probabilities are learned from the training set.

7.4.4 Application

For each input image of the test set, the patch classifier is applied as described in Section 7.2.4. After calculating the scene features, the scene classifiers are applied independently, and the HMMs are run. For the road type, we report the current maximum state; for the object classes, we report their presence if the classifier’s probability is \(> 0.5\).
[Felzenszwalb and Huttenlocher, 2006]. The first data set (Seq. Day), spanning 22,500 frames, was recorded during the day and is used for training/testing the scene classifier via cross validation. The second data set (Seq. Dusk), spans 15,000 frames and was recorded later on the same day but is quite different with respect to the number of moving objects (rush hour) and color distribution of the images (dusk and read/headlights from cars).

**Patch Classifier.** For training the patch classifier, 39 images from another, similar training set were segmented manually into the texture classes (Fig. 7.1 (c)).

**Scene Classifier.** Each image of both sequences was assigned to one of eight road types. Additionally, the presence of an object (pedestrian crossing, car, pedestrian) was marked. An overview of the class distribution on the training data can be seen in Tab. 7.1. Annotation was done as follows: each image was assigned to one of the available eight road types as soon as the beginning of the road type was below the lower third of the image (this corresponds to a distance of $\approx 20\text{ m}$ and was similar to the subjective feeling of when an image is considered as a given road type). *E.g.*, as soon as the lower border of an incoming street of a junction was below the middle line, the road type was assigned to “junction”. The rather large distance makes the problem quite hard, as the spatial resolution for the discriminating parts (with the rest of the junction even farther away) is very low: there are usually only around 5 rows of patches, *c.f.* Fig. 7.1.

Additionally to the road type, flags were set to indicate an object’s presence. Again, pedestrian crossings are annotated as soon as they are closer than $\approx 20\text{ m}$ and $\approx 10\text{ m}$ for cars. Pedestrians are split into two sets: one directly in front of the car (*e.g.*, at a pedestrian crossing), and pedestrians on the side with a height of $> 20\%$ of the image height. Due to their rather different features, we trained them separately. However, we report the figures for both pedestrian classes as one. Note that for cars and pedestrians, it is not our goal to obtain state-of-the-art performance, we rather demonstrate that such classes can be added without any necessary change to the system’s architecture. The scene classifier
7.6. Results

In the following, we first present results of the patch classification, investigating the effect of spatial smoothing as well as object detector integration. After that, we explore the use of this intermediate presentation for rough traffic scene understanding.

7.6.1 Patch Classification

To assess the quality of the basic patch classification stage, we report confusion matrices for several variations of the system in Fig. 7.7. Each row reports how the classifier voted for a specific class, with entries row-normalized. The higher the values on the matrix diagonal, the better. We compare a purely appearance-based feature set (a) with one that uses additional features from a depth map (b) and one that furthermore includes neighborhood constraints using an MRF (c).

![Confusion matrices for different variations of the basic patch classifier.](image)

**Figure 7.7:** Confusion matrices for different variations of the basic patch classifier.

was trained using 5-fold cross validation on connected subsequences of Seq. Day.

---

1The label “grass” was not used in this dataset, hence there are only 12 rows/columns.
### Table 7.2: Cross-validation errors for different stages and possibilities of the system.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cross-validation error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Patch</strong></td>
<td></td>
</tr>
<tr>
<td>Texture &amp; geometry</td>
<td>0.2587</td>
</tr>
<tr>
<td>Texture (8 × 8)</td>
<td>0.2605</td>
</tr>
<tr>
<td>Decision forest</td>
<td>0.2652</td>
</tr>
<tr>
<td>Additional depth</td>
<td>0.2520</td>
</tr>
<tr>
<td><strong>MRF</strong></td>
<td></td>
</tr>
<tr>
<td>Potts</td>
<td>0.2362</td>
</tr>
<tr>
<td>Class-dependent</td>
<td>0.2220</td>
</tr>
<tr>
<td>With detector</td>
<td>0.2172</td>
</tr>
</tbody>
</table>

In general, many of the prominent classes can be distinguished rather well, as can be seen from the block-like structure of the matrices: ground structures such as street and marking are well separated from standing structures as buildings and trees, or objects. Inside these categories, however, it is often difficult to keep labels apart: persons are often misclassified as cars, and sidewalk and road are often mistaken. The latter is especially challenging as appearance-wise, there is often no difference between street and sidewalk. Including depth helps classification of upright structures: cars are better separated from the street (0.56 correctly classified as opposed to 0.48), and also other classes such as pedestrians or building benefit from the inclusion. Still, depth feature cannot help in distinguishing cars from pedestrians due to the very local nature of the classification. In fact, many pedestrian labels that were before misclassified as road are then misclassified as car, with the correct rate staying the same (0.25). The effect of including neighborhood constraints via a Markov Random Field (MRF) using class-dependent edge potentials is shown in (c). While this gives a certain performance improvement (e.g., road structures are less confused with upright structures, sidewalk is even recognized better with an accuracy of 0.41 instead of 0.14), the computational complexity does not seem to be justified by the limited gain in performance.

This effect is also reflected in the cross-validation errors for the different stages, Tab. 7.2. Here, we additionally establish two baselines for the patch classifier: when not using the overlapping 16 × 16 textures,
7.6. Results

![Image](a) Original image with HOG detections  
(b) Ground truth

![Image](c) Spatial class-dependent model  
(d) Additional object detections without object detections

**Figure 7.8:** Effect of integrating the pedestrian detector. The discrimination between cars and pedestrians is considerably improved.

Performance suffers from the reduced context. Using decision forests [Shotton et al., 2008] instead of AdaBoost for the learning does also give a slightly worse performance. We also found it to be over-reliant on some basic features in further experiments, *i.e.*, once a wrong branch is chosen, the decision is done. Again, depth improves the classification at almost no additional cost. The MRF further improves the labeling, especially when using the class-dependent model. As it mostly affects pedestrians, the inclusion of the detector gives almost no change in the cross-validation errors. Still, the obtained labeling gets better and detector output is in general improved, as described next.
Figure 7.9: The stochastical model helps to filter out detections that have not enough support from the features: (a) Example image with filtered detections in red. (b) Performance curve for Seq. LOEWENPLATZ.

Inclusion of objects. Inclusion of object detections can improve classification performance further, as can be seen in Fig. 7.8. Here, a standard HOG detector for pedestrians is used, as introduced in Chapter 2. When using the additional information, most of the upright standing structure is classified correctly as pedestrian instead of car. This also works in the other direction, as shown in Fig. 7.9: with a threshold of $P(o_l = 1) > 0.5$, wrong detections are mostly filtered out. The performance curve for Seq. LOEWENPLATZ is shown in Fig. 7.9 (b). At lower thresholds, it exhibits a worse performance compared to the detector or the scene analysis system of Chapter 3. However, at $P(o_l) = 0.5$, it has a slight advantage over both other systems. This is also reflected by an increase of the correct rate for pedestrians from 0.26 to 0.5. Due to the close entanglement, the accuracy for cars also increases from 0.69 to 0.71. Note that this test does not yet include depth features such as “flatness” or distance comparisons that proofed very helpful in Chapter 3. Together with proper handling of overlapping detections, the MRF model should gain further in performance.

Selected features. To get an idea which features are most important, we report the influence of appearance, geometry, an depth features for the respective classes, when using the classifier with depth features
7.6. Results

![Figure 7.10: Feature distribution of the learned classifier for the different classes. (a) Distribution of feature types for patch classifier. (b) Distribution of feature types for scene classifier.](image)

The graph reports how many times a specific feature type was selected by the learning algorithm (please note that grass is ignored in this data set). The distribution of the features is rather even (note that there are considerably more appearance-based features available for selection), with no overreliance on a single, weak cue, like, e.g., geometry. The vertical image position seems to play an important role in most classes, especially for classes on the ground. Classes with a wide distribution of y-locations, e.g. trees or persons, hardly use the cue.

![Figure 7.11: Influence of depth information on the patch classifier. Object classes are identified better, but cannot be kept apart by the local information only.](image)
Example images. In a first example, we underline the usefulness of depth features for the classification. In Fig. 7.11, upright structures are identified considerably better by the patch classifier when depth is included. Even without spatial smoothing, the resulting labeling looks quite good, identifying even parts of the traffic light. Still, the local information alone is insufficient to keep cars and pedestrians apart from each other.

In a second example, we compare the effect of different learning strategies for the patch classifier. As already evident from the cross-validation errors shown before, the decision forest performs slightly worse than AdaBoost. In the top row of Fig. 7.12, the decision forest yields smoother results, however, loses some details. An extreme failure case is however shown in the bottom row, where a slightly too bright image results in its total failure. Thus, as indicated above, we will use AdaBoost as our classifier of choice.
Figure 7.13: Example of basic patch classification of an image (a) using only appearance (c), additional depth map features (d); and MRF-based smoothing using either a Potts model (e) or a class-dependent model (f).
Fig. 7.13 shows an example classification, first without, then with depth information, and then using the class-dependent MRF-based smoothing. Each of the steps gives a visual improvement of the result. In Fig. 7.13 (e,f), the effect of the spatial model is highlighted: as opposed to the simpler Potts model, choosing a class-dependent model does not only induce spatial consistency, it also preserves details like the traffic signs. Difficult classes like the sidewalk are pushed more than others, which is generally desired but not always done right as can be seen on the left side of the image.

Some more examples are shown in Fig. 7.14. In the upper example (a,b), the class dependent model favors the sidewalk too much, again underlining the difficulty of this class. In the lower example (c,d), we again highlight the labeling obtained when also including object detections.

Lastly, we show a result obtained on another data set from GeoAutomation\(^2\) in Fig. 7.15. For this, the training set was extended by some examples from their imagery, and depth data ignored. The rather different image quality and horizon make generalization quite challenging. Still, a reasonable result is obtained, except for the side of the image that is erroneously labeled as “car” and “sidewalk”.

Per se, the labeling gives visually good results and can improve object detection. In the following, we will now explore its use for further scene understanding.

### 7.6.2 Traffic Scene Understanding

In a first quantitative experiment for SUTSU on Seq. DAY, we compare our two-stage classifier with a classifier directly based on GIST features [Oliva and Torralba, 2001] (Fig. 7.16). To compare multi-class performance, we again use a confusion matrix, along with its characteristic numbers of accuracy (AC), defined as the total proportion of correct classifications, and average precision over all classes (AP). For the objects, being binary classifications, we report the number of errors, as well as precision and recall.

As can be seen, both methods manage to tell the first three classes (straight and curves) apart quite well but have problems with all the

\(^2\)http://www.geoautomation.be/
Figure 7.14: More examples of the MRF output on two input images, once with and once without object detections.

Figure 7.15: Patch classification result obtained on image from GeoAutomation.
different types of junctions, which is mostly due to the poor resolution of the patch classifier at high distances. Incoming junctions are often mapped to straight streets, where junctions going to the right are recognized slightly better due to the vehicle driving in the right lane. The class “place” is also identified rather reliably, whereas crossings and T-junctions are hardly recognized, with T-junctions often assigned to a “right curve”, probably again due to the fact that our sequences were recorded when driving on the right lane, which makes recognizing the left part of the street hard. The more complicated classes are often mapped to the dominant first three. This also indicates that the classes might be too similar given the limited data: a T-junction has both similarities to a curve and a straight road segment. Thus, it might be better to allow for more than just one class and use a more generative approach in understanding the scene.

Note that the confusion matrices indicate similar problems for both methods, a feature combination would thus not bring much. In general, reasoning works comparably well on global classes (AC: 0.57 (SUTSU) vs. 0.56 (GIST), AP: 0.45 vs. 0.42), also corroborating the effectiveness of GIST as a global scene descriptor. However, SUTSU achieves considerably better performance on object classes, c.f. Fig. 7.16 (c). This is due to the fact that its feature set is largely invariant to the positioning of an object class and also has a direct notion of periodic structures.
7.6. Results

Figure 7.17: Example images from Seq. DAY. For each image, the bottom left shows the patch classification output, as well as the scene classification (road type, present objects).

such as pedestrian crossings. Note again that our goal is not to train an object detector that can localize pedestrians or cars, our system merely detects the presence of the class.

Selected features. In Fig. 7.10 (b), the selected features for the different scene classes are shown. In all cases, the rough layout plays a central role in identifying the class or object presence. For road types, the orientation features also play an important role, with some support from periodicity features. As expected, the detection of pedestrian crossings heavily depends on these periodic features, whereas the other objects much more depend on the rough layout.
Figure 7.18: Example images from Seq. DUSK.

Figure 7.19: Typical failure cases (from left to right, top to bottom): incoming junctions are ignored due to low resolution; patch classifier mistakes car for pedestrian; headlight’s reflectance confuses the patch classifier; sidewalk is difficult to distinguish from road; road geometry not in set of classes; too complicated/ambiguous road geometry.

Example images. A few example images from both sequences are shown in Fig. 7.17 and Fig. 7.18. For each image, we plot the patch classification as well as the scene classification in the lower left corner. The
images also show a few results that were obtained by applying SUTSU without retraining either classifier to Seq. Dusk. As can be seen, it performs qualitatively similar. One typical failure case is due to the headlights’ reflectance on the car in front, causing false “street marking” patches and hence issuing the flag “pedestrian crossing”. Training the patch classifier on dusk conditions should alleviate such a problem. The failure can be seen, along with some other typical ones, in Fig. 7.19. Apart from some obvious mistakes, it is often even difficult for humans to select the right class for an image.

**Runtime.** The mixed C/C++ and Matlab implementation currently takes about 1 s for the patch classifier (C/C++) and another 1–2 s for the scene classification (Matlab). Most of the system is parallelizable, and should thus be amenable for real-time implementations.

### 7.7 Conclusion

We presented a two-stage method for inner-city street scene classification. First, we explored the use of appearance- and depth-based classification for assigning each patch of an image to one of 13 urban texture classes. Using neighborhood constraints in an MRF framework, the labeling can be smoothed and difficult cases such as the detection of the sidewalk can be improved. The method also allows to directly include object detections into the formulation, allowing a mutual improvement of both labeling and object detector confidence. In practice, the full modeling produces nice visual results, but is currently too slow for an application on a vehicle and does not give as good results for object detection as the specialized model of Chapter 3.

Alternatively, we showed the application of the basic patch classifier to more global scene analysis. Based on the obtained probability maps, we construct a pool of intermediate features that are then used to classify both road typologies, as well as detect the presence of relevant objects. The approach was tested on two challenging sequences and shows that while a state-of-the-art scene classifier can keep global classes such as road types similarly well apart, a manually crafted feature set based on a segmentation clearly outperforms a global classifier on object classes.
This system offers exciting possibilities for future work. On the one hand, the components of the system can clearly be improved: the texture classifier could benefit from more features, e.g. based on 3D points (c.f. [Brostow et al., 2008]) or optic flow. The integrated modeling could benefit from the handling of overlapping detections, as well as the inclusion of more categories. Inclusion of more higher-order information (e.g., super-pixel segmentation, [Kohli et al., 2008; Sturgess et al., 2009]) could help in obtaining a smoother labeling. The scene classifier also could directly include the vehicle’s ego-motion as well as trajectory information from the tracking system to reason about the scene. Going beyond vision-based sensors, the fusion with GPS map data is another challenge. On the other hand, the system’s actual application to autonomous driving can be investigated. Here, the set of classes can be expanded, and the interplay with tracking, path planning, or other attentive mechanisms be explored.
Conclusions and Outlook

In this thesis, a mobile vision system was presented that performs self-localization, scene analysis, and object tracking based on visual input only. For each of these problems, one or more possible solutions was described, mostly based on the input from a pair of forward-looking cameras that are mounted on a movable platform.

The self-localization is based on state-of-the-art research in visual odometry and was augmented with ideas from SLAM, as well as semantic information from other modules. For the analysis of the scene, various algorithms and subproblems were investigated. On the one hand, we explored methods for constructing static and dynamic occupancy maps with possible applications in path planning, as well as an algorithm that could infer the road type and the presence of a certain set of objects in front of the observer. On the other hand, we introduced two algorithms that are able to find the mutually best explanation of the scene, including both valid object detections and the ground plane or a texture labeling of the scene. In both cases, we have shown that their coupled solution results in both better scene understanding and a more reliable set of object detections.

This set of detections is then used as input to a multi-object tracking algorithm that operates in a hypothesis selection framework. First, it generates an overcomplete set of possible trajectories based on information from the past and present image pairs. In the second stage, these hypotheses compete against each other for physical space, resulting in a more accurate set of valid trajectories due to the joint reasoning. The corresponding motion models for the two most common urban object categories, pedestrians and cars, were implemented.
Most of the components presented in this thesis were combined into a mobile vision system that can per se function as an input component to other tasks. We demonstrated this for dynamic occupancy map generation and articulated tracking. Furthermore, we developed an advanced, simulation-based motion model that jointly estimates the motion of pedestrians and proved its applicability to multi-object tracking. All methods were tested on an extensive data set of busy urban scenarios, recorded from a variety of platforms such as child strollers or intelligent cars.

A first key insight was that robust multi-object tracking in almost real-time is possible in busy urban scenarios. Firstly, this was achieved by postponing decisions in a hypothesize-and-test paradigm, where we always allowed more hypotheses in the beginning and then only pruned them in the presence of more information (i.e. scene geometry, other hypotheses). Secondly, the main components of the systems were designed in an integrated fashion, with cognitive feedback channels between the components and separate failure detection mechanisms for each. Thirdly, in most of the above problems, we demonstrated that the principled use of depth information can improve the results of most algorithms. Our experiments underline the importance of these design steps in a system to be deployed in real-world scenarios.

The combined system was implemented in C/C++, with some algorithms operating on the graphics card. For relatively busy scenes, frame rates of up to 8 fps could be reached, which in the future can be further optimized, also by parallelization onto multiple machines. Starting from this, it will be interesting to actually deploy the system on a car or robot.

8.1 Contributions

In detail, the contributions of this dissertation are as follows.

Chapter 3 proposed a probabilistic combination of object detection, depth estimation, and scene geometry that is able to jointly estimate the ground plane of the scene while meaningfully rescoring detections, irrespective of the used basic detector. Specifically, improvements for ISM, HOG, and a part-based detector were shown on a set of challenging
urban sequences, underlining the idea that the introduction of further context can indeed help in improving object recognition.

In Chapter 4, we used these filtered detections along with visual odometry to provide stable tracking of both pedestrians and cars. As the employed tracker works in a two-stage process that first creates a set of hypotheses, we proposed an efficient method for their creation which combined the advantages of Markovian trackers with an additional observe-and-explain stage. The motion models for both classes were designed such that they correctly take measurement uncertainty into account, thereby enabling object tracking across a wide range of scales. The second stage of optimization was further analyzed to allow for occlusions and runtime was optimized by exploiting the submodularity of the ensuing optimization problem. Again, the resulting tracker was tested on a set of urban sequences, delivering stable tracking performance in a variety of different scenarios.

In Chapter 5, we then further analyzed the interplay between the components, pointing out the importance of failure detection and cognitive feedbacks for robust system performance, especially in the case of visual odometry. Furthermore, we showed how the resulting mobile vision system can be used for further applications in dynamic occupancy map generation and articulated tracking. Especially the latter showed that by factorizing the state space of motions using a detection-based tracker, articulated multi-body tracking can be made tractable. We did so by interfacing the multi-body tracking system with the articulated tracker by ways of a guided segmentation stage, in order to reliably extract pedestrian silhouettes in busy scenes and under significant camera egomotion. The proposed combination achieves robust articulated multi-person tracking performance in very challenging sequences.

Chapter 6 introduces a simulation-based motion model for tracking as a more advanced substitute for commonly used Kalman filters that extrapolate each tracklet independently. The developed simulation accounts for normal walking people, including collision awareness and goal-driven behavior. In our experiments on semi-crowded overhead views, the inferred predictions were constantly better than a linear model or another simulation from the literature. When applied to pedestrian tracking from a mobile platform, the method could help refining people after occlusions, thus lowering the number of identity switches in busy en-
environments. In general, the method seems to be particularly suited for prediction tasks in path planning.

Finally, in Chapter 7 we described several methods based on the texture classification of the scene. Given an input image, each patch on a regular grid is assigned to a urban texture class with the help of a classifier that takes into account texture, geometry, and depth information. Then, similar to Chapter 3, we demonstrated that the joint optimization of object detection and labeling can help both components to improve performance. The optimization was done by including object detections in a Markov random field for labeling. Furthermore, we showed how the obtained labeling can be used as intermediate representation through which one can infer the road type in front of the observer, as well as determine the presence of certain object classes, such as pedestrian crossings or cars, without the help of a specific detector.

8.2 Perspectives

Throughout the thesis, the system and its various extensions have been successfully applied to several video sequences. As of now, the system is still far away from the long-term goal of interpreting a scene in a similar manner as a human observer, guiding visual attention and trying to analyze the intentions of other traffic participants. However, the current state constitutes a good basis for a number of improvements and possibilities for immediate future work, which we will summarize in the following.

Integration. One of the most challenging and imminent tasks is the integration of the proposed vision system with an actual robotic or autonomous driving platform. There, both its robustness over hour-long runs, as well as its application to path planning need to be investigated. This will not only require further runtime optimizations of the components, but also the careful design of their interplay with the rest of the system. As the runtime of the various components is becoming acceptable, the power consumption of modern GPUs becomes another restricting factor for mobile scenarios. Thus, once the methods are bet-
ter understood, a possible implementation on dedicated hardware (e.g., FPGAs) should be investigated.

**Evaluation.** Related to an actual implementation is the more comprehensive evaluation of the tracking approach. This also involves a more thorough comparison to other tracking approaches: as indicated in the literature review of Chapter 4, many different components are available for data association and optimization, and there is no clear indication yet which part plays what role. Such an evaluation requires a powerful tool that can handle the various complexities of the system, including occlusions, areas of interest, and any anomalies in either ground truth or tracker output. In the context of path planning, localization accuracy is another problem that also puts considerable demands on the ground truth itself. To this date, a satisfactory solution for a completely automatic evaluation is still not existent.

**Extensions.** Throughout the course of a prolonged testing, one is more than likely to encounter new challenges for the system that have to be accounted for. For instance, one will have to go beyond the flat ground plane assumption to account for arbitrary road geometries—possibly dangerous items might not even be located on the road itself. The tracking needs to support more object categories and their respective motion models. Even for pedestrians and cars, erratic motions might currently lead to tracking failure. Inclusion of more categories is directly linked to the system’s scalability. While we demonstrated successful handling of pedestrians and cars, actual urban scenarios contain a multitude of object (sub-)categories. In fact, even seemingly simple classes like “car” pose a challenge due to their variability in viewpoint and type. To circumvent the separate design of a single detector for every instance, multi-class approaches need to be developed that should not only scale more favorably given the number of classes, but should also make sure that one image region is not assigned to more than one object instance, which can be a problem when several detectors are run independently.

**Multi-level multi-object tracking.** Once given such a large number of objects, the question then becomes how this translates to the actual
object tracking framework. While the current system contains the basic capability to handle different hypotheses and their physical exclusion (also, in parts, due to our simulation-based motion model), its scalability will also become an issue, once almost every physical location in front the camera is occupied by one of many object classes, moving from some unknown location to another. This also entails the handling of (partial) occlusions in an integrated fashion between detector and tracker. In such crowded cases, it then becomes questionable whether each agent needs to be modeled independently, and in how far we can operate in dense environments (think Japan) with the current approaches to detection and tracking. Generally speaking, the tracking and its articulated extension could be thought of as two specific levels of a multi-level motion analysis system. This system could be extended further on both ends, with face and gesture recognition providing more fine-grained analysis, whereas crowd-based tracking could constitute the higher, more general levels.

**Scene understanding.** Besides the main system, we also presented some components that are currently “on the horizon” and not ready to be deployed on a robot. Especially the urban scene analysis opens up a number of interesting possibilities for future research. Indeed, many current systems in mobile robotics apply specialized lane finding, object and traffic sign recognition approaches, and their interplay with a more general scene understanding is a challenging area of future research. On the one hand, the scene analysis can be extended to take advantage of more cues, including, *e.g.*, crowd analysis or global map data. On the other hand, its integration with the other presented components is an exciting challenge: can tracking or prediction indeed benefit from such an advanced analysis? If so, how much? How can one control the influence of priors obtained from more global sources? What are the new bottlenecks for reliable prediction?

**Sensor fusion.** Lastly, the inclusion of further sensor modalities can boost the various components’ performance. Visual odometry can benefit from both GPS and inertial measurements units (IMUs), allowing for drift-free localization that is largely independent of available static structure or lighting conditions, while still reducing local drift of the
IMU using vision. Object localization and detection might receive additional input from laser or other image sensors (e.g., omni-directional or infra-red cameras), allowing for a wider range of lighting conditions, while introducing semantics for laser points. Scene analysis can profit from global map data obtained with the help of self-localization, thus setting useful priors for class labels of the image segmentation, as well as entrance and exit points for tracking. Incongruencies between local image data and global map data might even help in the detection of dangerous situations.

While the presented system’s focus lies with urban environments and possible uses in path planning, a multitude of other application scenarios can be envisioned. These range from human-computer interaction (in the close range of a robot) to tracking in surveillance, movie production, sports, or traffic flow analysis. Depending on the actual use case, components can be adapted, extended, or fed with more information. Still, the basic premises already discussed in the introduction should remain the same: the use of hypothesize-and-test and probabilistic formulations for the various components, and their tight interaction in a system view.
In Section 7.3.1, two possibilities for a CRF’s edge functions were discussed. The class-dependent model is particularly interesting, as it allows one to reason about two patches co-occurences. Formally, the edge function can be regarded as a $C_p \times C_p$ matrix $\theta$, with $C_p$ the number of classes,

$$\psi(x_i, x_j; \theta) = \theta_{\psi, i, j}. \tag{A.1}$$

For better classification performance, we differentiate between horizontal and vertical edges:

$$\psi(x_i, x_j; \theta) = \begin{cases} \theta_{\psi, h, i, j} & \text{if } (i, j) \text{ is a horizontal edge} \\ \theta_{\psi, v, i, j} & \text{if } (i, j) \text{ is a vertical edge} \end{cases} \tag{A.2}$$

Due to the matrices’ symmetry, there are $C_p^2 + C_p$ parameters to learn. In contrast to [Wojek and Schiele, 2008] we optimize the parameters globally. This is done by maximizing the posterior probability $O$ of $\theta$ over all $M$ training images, where $\{x\} := \{x^{(m)}\}_{m=1}^M$ and $\{y\} := \{y^{(m)}\}_{m=1}^M$

$$O(\theta|\{x\}, \{y\}) \propto \Pr(\theta) \cdot \Pr(\{x\}|\{y\}; \theta) \tag{A.3}$$

$$= \Pr(\theta) \cdot \prod_{m=1}^M \Pr(x^{(m)}|y^{(m)}; \theta) \tag{A.4}$$

$$\approx \Pr(\theta) \cdot \prod_{m=1}^M \frac{1}{Z(y^{(m)}; \theta)} \gamma(x^{(m)}|y^{(m)}; \theta) \tag{A.5}$$

$$\theta^*_\psi = \arg\max_{\theta_\psi} O(x|y; \theta) \tag{A.6}$$
$Z(y; \theta)$ is the partition function, such that $\sum_x \frac{1}{Z(y; \theta)} \Phi(x|y; \theta) = 1$. As we’re operating on fractions rather than absolute counts, we choose a Gaussian distribution $\Pr(\theta) = e^{-\frac{1}{2\sigma^2} \|\theta\|^2}$ for the prior.

In its negative logarithmized representation, this becomes

$$L(\theta) = \log(\mathcal{O}(x|y; \theta))$$
$$= \frac{\|\theta\|^2}{2\sigma^2} - \sum_{m=1}^{M} \left(-L_Z(y^{(m)}; \theta) + L_\Upsilon(x^{(m)}|y^{(m)}; \theta)\right) \quad (A.8)$$
$$\theta^*_\psi = \arg\min_{\theta} L(x|y; \theta) \quad . \quad (A.9)$$

This is a convex optimization problem (c.f. [Vishwanathan et al., 2006]) that can in principle be solved with any optimization algorithm. However, for random fields that are larger than just a few nodes, the computation of the partition function $Z$ quickly becomes intractable. Possible solutions to this problem—as the one we used in the following section—are described in [Vishwanathan et al., 2006].

### A.1 Optimization

One solution for this problem is to avoid the computation of the objective function itself and instead just compute its gradient. This implies however that the optimization algorithm must only depend the gradient. This is true for gradient descent algorithms.

To compute the gradient, we have to rewrite the objective function of Eq. (7.11) as

$$\Upsilon(x|y) = e^{\langle \zeta(x,y), \theta \rangle} \quad , \quad (A.10)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $\zeta(x, y)$ is called the sufficient statistics of the distribution. The log partition function used in (A.8) is now written as

$$L_Z(y; \theta) = \log \sum_x e^{\langle \zeta(x,y), \theta \rangle} \quad , \quad (A.11)$$

which is the cumulative generating function of the exponential family. This means that

$$\frac{\partial}{\partial \theta} L_Z(y; \theta) = \mathbb{E}\mathcal{O}(x|y; \theta) [\zeta(x, y)] \quad , \quad (A.12)$$
where $E$ is the expectation value, leading us to the gradient

$$
g(\theta) := \frac{\partial}{\partial \theta} \mathcal{L}(\theta) = \frac{\theta}{\sigma^2} - \sum_{m=1}^{M} \left( \varsigma(x^{(m)}, y^{(m)}) - \mathbb{E}_{O(x|y^{(m)}; \theta)}[\varsigma(x, y^{(m)})] \right).$$

(A.13)

The parameters $\theta$ and the $\sigma$ are defined above. To solve Eq. (A.13), we thus need only to define the sufficient statistics $\varsigma$ in the right way, in order to receive the same objective function that we already defined in Eq. (7.16). Since we need only the gradient of $\theta_\psi$ and not $\theta_\phi$, there is no need to define $\varsigma_\phi$ for the node functions. For the edge functions, this goal can be reached by defining the sufficient statistics $\varsigma$ as the number of occurrences of each pair of classes.

The expected number of cooccurrences, $\mathbb{E}_{O(x|y; \theta)}[\varsigma(x, y)]$, can be approximated with belief propagation. This has the additional advantage that the training is very specific to the used inference algorithm. However, this results in a rather slow training phase.
List of Publications


Bibliography


[Breitenstein et al., 2009] M. Breitenstein, F. Reichlin, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle fil-


[Lawrence, 2007] N. D. Lawrence. Learning for larger datasets with the
   gaussian process latent variable model. In Proceedings of the Eleventh

   bags of features: spatial pyramid matching for recognizing natural
   scene categories. In Proceedings of the IEEE Conference on Computer

   and posture manifolds for tracking. In Proceedings of the International

   ing using multi-level structured models. In Proceedings of the European

[Leibe et al., 2005] B. Leibe, E. Seemann, and B. Schiele. Pedestrian
detection in crowded scenes. In Proceedings of the IEEE Conference
   on Computer Vision and Pattern Recognition, 2005.

   Dynamic 3d scene analysis from a moving vehicle. In Proceedings of
   the IEEE Conference on Computer Vision and Pattern Recognition, June 2007.

detection and trajectory estimation for multi-object tracking. In Pro-

[Leibe et al., 2008a] B. Leibe, A. Leonardis, and B. Schiele. Robust ob-
ject detection with interleaved categorization and segmentation. Inter-

[Leibe et al., 2008b] B. Leibe, K. Schindler, N. Cornelis, and L. Van
   Gool. Coupled detection and tracking from static cameras and mov-
   ing vehicles. IEEE Transactions on Pattern Analysis and Machine

[Leonardis et al., 1995] A. Leonardis, A. Gupta, and R. Bajcsy. Seg-
   mentation of range images as the search for geometric parametric

[Lerner et al., 2007] A. Lerner, Y. Chrysanthou, and D. Lischinski.


Curriculum Vitae

Name Andreas Ess
Date of Birth 15.12.1980
Place of Birth Baden, Switzerland
Citizenship Austria

Education

2005 – 2009 Doctor of Science ETH
ETH Zurich, Computer Vision Laboratory
2000 – 2005 Master of Science ETH in Computer Science
ETH Zurich, Department of Computer Science
1991 – 1999 AHS Matura
Bundesgymnasium Feldkirch, Austria

Occupations

2005 – 2009 Research Assistant
ETH Zurich, Computer Vision Laboratory
2004 – 2005 Research Intern
Siemens Corporate Research, Princeton
2002 – 2004 Freelance Software Engineer
Texas Instruments
1996 – 2000 Software Engineer
Omicron electronics AG, Austria/Houston