Surround Vehicle Analysis

Master's Thesis 2016
Jacob Velling Dueholm
Miklas Strøm Kristoffersen

AALBORG UNIVERSITY

UC San Diego

AALBORG UNIVERSITY
STUDENT REPORT

School of Information and Communication Technology
Selma Lagerlöfs Vej 300
9220 Aalborg Øst, Denmark
http://sict.aau.dk

Title:
Surround Vehicle Analysis

Theme:
Master's thesis: Vision, Graphics, and Interactive Systems

Project Period:
Fall 2015 and spring 2016

Project Group:
VGIS 1040

Authors:

Jacob Velling Dueholm

Miklas Strøm Kristoffersen

Supervisors:
Thomas Baltzer Moeslund
Andreas Møgelmose
Mohan Manubhai Trivedi

Number of Prints:
5

Number of Pages:
171

Date of Completion:
June 2, 2016

Abstract:

Keeping an overview of surrounding vehicles is a nearly impossible task using only human senses. This motivates automatic tracking of surrounding vehicles using cameras with use in both passive and active safety applications.

This study presents the development of a novel framework for tracking vehicles in full surround using computer vision techniques. The framework consist of a vehicle detector, a modified tracker optimized for multi-perspective tracking, and an association of tracks in real world coordinates to achieve consistent trajecotories in full surround.

A vision-based dataset is collected using four GoPro cameras with more than 4000 annotated vehicles which is used in the evaluation of the detector, tracker, and multi-perspective tracker. A trajectory dataset is collected from 50 sequences using the developed framework on which a trajectory analysis is made using machine learning to demonstrate its uses in naturalistic driving studies and advanced driver assistance systems.

Preface

This student report covers a 50 ECTS master's thesis made by Jacob Velling Dueholm and Miklas Strøm Kristoffersen studying Vision, Graphics and Interactive Systems at Aalborg University. The main work is done abroad at University of California, San Diego (UCSD), from November 2015 to May 2016 in the Computer Vision and Robotics Research (CVRR) laboratory closely coupled to Laboratory for Intelligent Vehicles and Safe Automobiles (LISA) under supervision of professor Mohan M. Trivedi. This work has so far lead to one accepted paper in the IEEE Computer Vision and Pattern Recognition Workshops (CVPRW) on Automatic Traffic Surveillance (ATS), a journal article under review in IEEE Transactions on Intelligent Vehicles (T-IV), and a planned submission to IEEE Intelligent Transportation Systems Conference (ITSC).

The target group of this report are graduate students, supervisors, and other parties with an interest in multi-perspective tracking of vehicles. The central aspects of this project are multi-camera setup, detection and tracking of vehicles in single perspectives, track association between perspectives along with trajectory analysis.

Reading Guide

This report is meant to be read independently of the papers in appendix. The report allows a more detailed description and the thought process which is less transparent in papers. The report is structured in five parts. The first part introduces and defines the term multi-perspective tracking. The second part describes a preliminary system functioning as a proof of concept, before investing in the final setup used to gather a dataset and further develop the framework as described in the third part. Part four completes the project, and part five consists of papers in their current state.

Acknowledgment

The authors would first and foremost like to thank professor Mohan M. Trivedi for opening up his laboratory to us and always welcoming us with a smile. We thank you for giving us insight to the research community and for letting us contribute in such a novel and interesting field.

A wide thanks goes out to the entire CVRR laboratory, Ravi, Eshed, Sujitha, Sean, Frankie, Akshay, Rakesh, and Kevan, for the daily interaction when hanging out by the printer, and for the good times driving around San Diego "collecting data". A special thanks to Ravi and Eshed for sharing your profound expertise and keep suggesting innovative ideas whenever we hit a wall.

We also would like to thank professor Thomas B. Moeslund for making this unique experience possible, together with Andreas Møgelmose for the weekly correspondence of valuable comments. Lastly we would like to thank all our family and visitors, dragging us out of the laboratory to spend some time road tripping the beautiful California.

Contents

Part I

Introduction

Chapter 1

Keeping an Eye on the Road

Vehicles are an integral part of the everyday transportation for a large part of the population. In particular, USA has adopted a culture heavily based on the use of vehicles. As of 2014, approximately four vehicles were registered for every five persons that lived in USA [51]. As one can imagine, this means a large number of vehicles are driving on the roads – 252 million in 2014 to be exact, and that number keeps increasing for every year that passes by according to the trend shown in Figure 1.1. The excessive number of vehicles

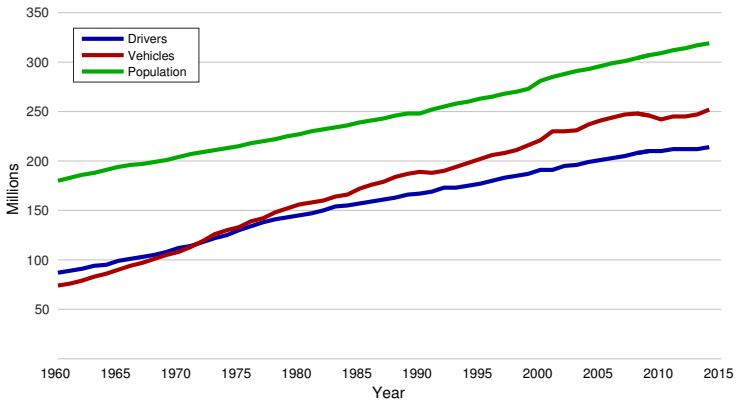


Fig. 1.1: The development in the number of registered vehicles, licensed drivers, and the population in USA [51].

leads to congestion, especially in the larger cities, and an average American spent approximately one hour in a vehicle each day in 2009 [50]. This adds up to a lot of hours spent in vehicles in USA every day. Unfortunately, the

intensive use of the roads led to 32,675 fatalities from 29,989 fatal crashes in 2014 [53]. That is approximately 90 people that are killed in vehicle accidents every day in USA alone.

Several initiatives have been taken through the years to bring down the number of fatalities caused by vehicle crashes. Classic examples include passive safety systems such as the airbag and the seat belt. It is estimated that 12,174 lives were saved by seat belts, and that approximately 3,000 lives could have been saved in 2012 if all US vehicle occupants were restrained [52]. In recent years, intelligent applications for vehicles have received increasing attention. These are often referred to as advanced driver assistance systems (ADASs), as they seek to make drives more convenient for the driver, and not least to assist the driver in potentially dangerous situations.

An understanding of pre-crash circumstances is key to the development of crash countermeasures. Naturalistic driving studies (NDSs) seek to expose the contributing factors to near crashes and crashes, concerning among others driver behavior and driving context. Large NDSs have been conducted [6, 9, 38, 54], and many hours have been spent manually labeling events from the collected data. The sensor suites are designed to monitor both the inside and the outside of the vehicles, and it has for example been found that driver inattention to the road is a contributing factor in almost 80% of crashes [9]. It is thus not only interesting to see how vehicles drive on the road, but also what the state of the drivers are while doing it. Multiple studies have proposed automatic extraction of critical events inside and outside a vehicle in NDS data [31, 43, 48], which allows for faster analysis of e.g. driver behaviors. In terms of awareness of the driver, it is relevant to study where the driver is looking in the time up to a crash, as this is most likely the direction of attention of the driver (see Figure 1.2). Every second spent in a vehicle is a fight of attention between driving and secondary tasks such as using a smartphone, eating, and talking to passengers. Secondary tasks are the most frequent contributing type of inattention in crashes [9]. The second most frequent contributing type of inattention might be a bit surprising, as this is actually driving-related inattention, which is e.g. looking in side and rear mirrors. In fact, driving-related inattention is a contributing factor in approximately 30% of crashes [9]. Though the study actually finds the driving-related inattention to be a protective behavior[1], it does raise a question of the limits to human sensing. For instance, the driver can only

[1]Drivers that check their surroundings are often more precautious [9].

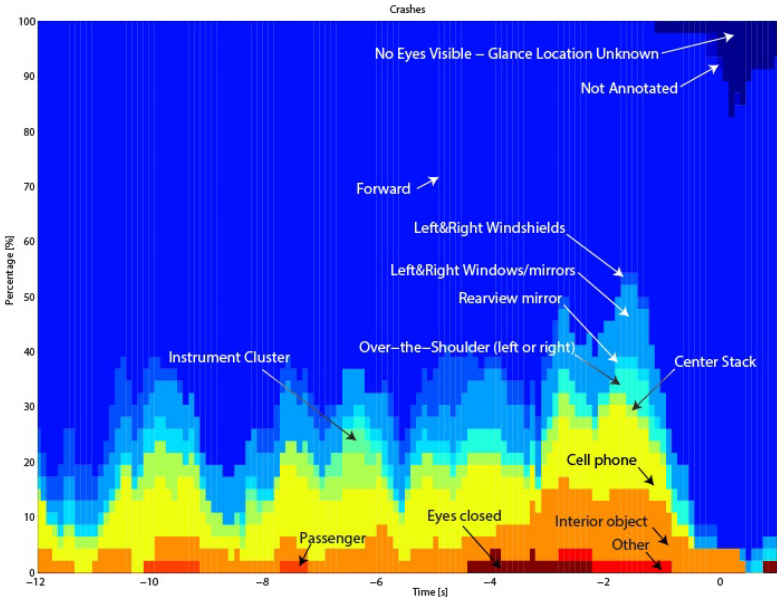


Fig. 1.2: Where the driver looks in the seconds up to a crash annotated manually in the SHRP2 NDS [54].

look in one direction at a time, and thus has to keep an internal map (that is often based on assumptions of how vehicles are going to move) of what is happening around the vehicle. This can be thought of as a similar approach as prediction-correction based estimation of dynamics (e.g. the Kalman filter): The driver relies on a prior knowledge of typical dynamics of vehicles to estimate the position of surrounding vehicles. These estimates are supported by what the driver actually sees, and the knowledge is updated by e.g. observed aggressiveness of surrounding drivers. However, when the driver is not looking at a vehicle (e.g. a vehicle positioned behind the vehicle), the estimated movement of the vehicle is only based on what the driver thinks it is going to do. This is where the driver (and in particular Kalman filters) is likely to fail if the assumptions do not hold true, and potentially dangerous situations arise. The question is thus how a system can help the driver observe and understand what is happening around the vehicle, since the driver can not observe the full surroundings at all times, and furthermore might be inattentive due to secondary tasks.

Some research and development even revolve around the idea of completely removing the driver from the loop. The result is fully autonomous

vehicles that currently receive notable coverage in media due to the potential revolution of vehicle transportation. Just like human drivers, autonomous vehicles need to sense the surroundings in order to navigate safely among other vehicles. To this end, they rely on a wide range of sensors such as lidars, radars, GPSs, and visual cameras [30]. In terms of sensing of surrounding vehicles the autonomous vehicles mainly use lidar and radar, and multiple autonomous vehicles have successfully driven public roads on pre-mapped routes.

In the near future, the driver will most likely remain in the loop, and an increase in ADASs will ensure a slow transition to fully autonomous vehicles (e.g. adaptive cruise control, driver drowsiness detection, parking assistance, and collision avoidance). An important branch of research within ADASs deal with visual cameras, because of the large amount of information contained in images compared to for example 3D point clouds from lidars containing purely spatial information. This allows for e.g. classification of traffic signs [34], traffic lights [24], turn and brake signals [44], and eye gaze direction of the driver [49]. In terms of looking out applications the focus has primarily been on front-facing cameras, thus the road ahead of the vehicle, which is also where the main attention of the driver is directed (see Figure 1.2). One of the most popular publicly available datasets, the KITTI Vision Benchmark Suite [20], use a front-facing camera for observing vehicles in front of the ego-vehicle[2]. In thread with the sensing problem of surrounding vehicles, the research within vision-based methodology is very limited. Knowledge, as to what multiple cameras (that observe surrounding vehicles all the way around the ego-vehicle) are potentially capable of achieving and which challenges will be met, is thus unexplored.

This study takes the first steps towards observation and understanding of vehicle movements all the way around an ego-vehicle using visual cameras. It is investigated how vehicles are successfully located and followed in the proximity of the ego-vehicle, and furthermore an analysis of trajectories collected on highways in USA shows how trajectory modeling can be used to learn trends in driver behavior. Potentially, the multi-perspective framework is useful for both NDS and ADAS, and even autonomous vehicles might eventually add a full surround looking visual camera setup to the sensor suite.

[2]Generally in this thesis, the ego-vehicle is the vehicle that is equipped with cameras and thus observing the surroundings.

Chapter 2

Reader's Guide

This thesis serves as an in-depth description of three papers produced during the development. These papers are attached as appendices A, B, and C.

- *Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies*, accepted for IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016
- *Trajectories and Behaviors of Surrounding Vehicles Using Panoramic Camera Arrays*, submitted for IEEE Transactions on Intelligent Vehicles
- *Vision for Intelligent Vehicles & Applications (VIVA): Multi-Perspective Vehicle and Trajectory Challenge*, incomplete paper intended for submission to IEEE Intelligent Transportation Systems Conference, 2016

Much of the work has therefore been aimed at paper submissions, which means that the system is not a chronological process resulting in one final prototype. In Part II (Proof of Concept) the first system is described. It is based on the first paper, and uses six cameras. The conclusions from this work is used to improve the system in Part III (Multi-Perspective Tracking and Trajectory Analysis), which uses a different setup with four cameras. This setup is the basis for the last two papers. As the two systems are designed to solve the same problem, some methods will recur, but in general methods will be explained in details in Part III. Both parts contain a local summary of what is included, and the reader is suggested to resolve to these in order to get a quick overview of the differences.

Scope of Project

It is not the aim of this study to solve all aspects of the problem. Some noteworthy limitations include:

- Only highways are considered, since these are subject to simple traffic structure
- Weather conditions are sunny in all recordings, for which the climate of San Diego is to blame
- No data have been captured at nighttime
- During capturing the ego-vehicle is kept at the same relative position on the road, thus no lane changes are performed

Part II

Proof of Concept

Summary

This part describes a preliminary framework built to test if vision-based full surround is an option at all, and if so, how to obtain a viable setup. The framework is complete from mounting the cameras on the testbed to obtaining an analysis of trajectories found surrounding the ego-vehicle.

The setup consists of six uncalibrated cameras to achieve a full surround with a slight overlap between each perspective. Vehicle detection and tracking is applied in each separate perspective. Tracks are associated between perspectives in the image plane by utilizing the overlapping regions, resulting in long trajectories all around the vehicles. Ten sequences are analyzed and visualized, reducing the drives into an event list for easy interpretation. Lastly, the setup is reflected upon for further improvements.

The work described in this part is published in IEEE Conference on Computer Vision and Pattern Recognition Workshops, Automatic Traffic Surveillance, 2016, titled *Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies*. The paper is found in Appendix A.

Chapter 3

An Experimental Study

Using multiple cameras on a moving platform to gain a full surround view is a novel area of research with only few publications and many unanswered questions. The closest work [57] published in January 2016, utilize the Ladybug3 camera to record omni-images in full surround at 1 Hz as seen in Figure 3.1. The omni-image is divided into five pieces to detect vehicles at different viewpoints. It should be noted the Ladybug3 is able to capture 1600×1200 omni-images at 6.5 FPS, and 16 FPS using JPEG compression according to the datasheet[1]. The aim of this project is to not only detect vehicles, but a full framework by also tracking and analyzing trajectories. A small dataset is therefore gathered for the completeness of proving the possibilities of a full surround vision-based framework.

Fig. 3.1: Omni-image captured by a Ladybug3 camera used by [57].

3.1 Camera Setup

The Ladybug3 seems to truncate vehicles driving close by, in its attempt to capture the full surround from a single position. Instead six individual cameras are used in this pilot study positioned on the rooftop rails as seen in

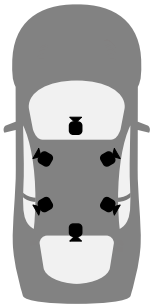
[1] `https://www.ptgrey.com/support/downloads/10149`

Figure 3.2 giving a full surround view with slightly overlapping regions.

The front and rear view cameras are of higher priority and is capturing at an increased 1280×960 resolution, compared to the side views 640×480 using wide-angle lenses which introduces distortion. Using this configuration the testbed is able to capture in full surround synchronously at 15 frames per second (FPS). No calibration is used in this setup, neither extrinsic nor intrinsic.

(a) PointGrey Ladybug3 Used by [57]

(b) Side view camera with wide-angle lense



(c) PointGrey Camera Configuration

(d) Sixcam setup. The front perspective is located inside the cabin.

Fig. 3.2

3.2 Data Collection

The data used in this system is gathered by the LISA lab, since no public available datasets were to be found at the time. The instrumented vehicle is driven on U.S. highways in southern California. The captured data are comparable to European databases. The main differences are larger roads with more lanes and a larger count of pickups and vans. U.S. highways are ideal for capturing data with multiple lanes, and five lanes are not uncommon on a single stretch. Sample pictures are shown in Figure 3.3. The weather is mainly sunny with possibility of clouds. At least one perspective will thereby be affected by sun glare with a full surround setup as seen in Figure 3.3c. No post-processing is done to correct colors or distortion.

(a) Front Left **(b)** Front **(c)** Front Right

(d) Rear Left **(e)** Rear **(f)** Rear Right

Fig. 3.3: A sample frame from the six cameras that show some of the challenges (e.g. sun glare).

Ten sequences are extracted and used for further analysis, to avoid footage with ego-vehicle lane changes which is unaccounted for in this work. The sequences contain activity around the ego-vehicle e.g. overtakings and lane changes. The length of the sequences varies from a few seconds up to a minute.

3.3 Multi-Perspective Trajectory Estimation

The challenge of tracking vehicles in full surround is divided into three modules as illustrated in Figure 3.4. The first module is the vehicle detection applied to each of the six cameras. These detections are used for the tracking module, associating detections in consecutive frames to track each vehicle over time. These two modules are well-researched topics within computer vision with off-the-shelf methods. The *Multi-perspective Tracking* module is responsible for associating tracks across perspectives to obtain a longer track history. This is a novel topic without much guidance. In this proof of concept the multi-perspective association is done in the image plane given the uncalibrated camera setup.

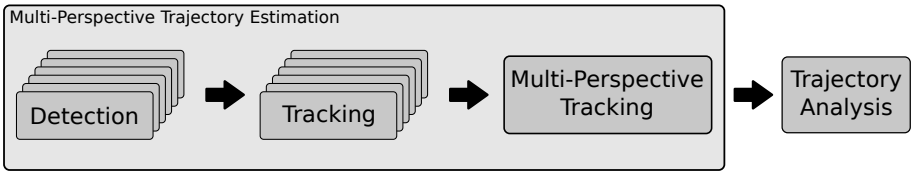
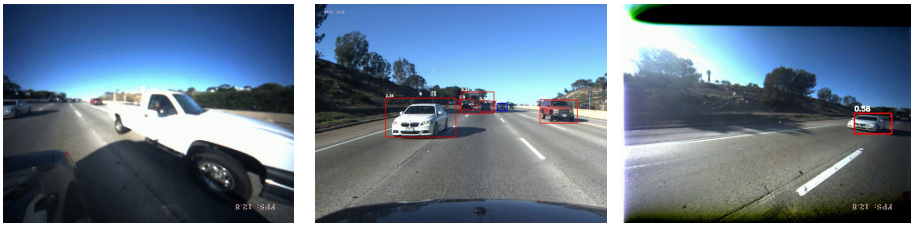


Fig. 3.4: Flow diagram of trajectory estimation and the afterwards analysis of trajectories. Detection and tracking is applied in each individual view, to be associated in the image plane.

Vehicle Detection

The deformable part models (DPM) vehicle detector is used, as DPM is known as a common baseline with several variations. The specific implementation used is pre-trained on the KITTI dataset by Geiger et al. [18]. This implementation uses the DPM in two stages. First stage is a regular DPM on the entire image, and the second stage is applied to an upscaled image around the horizon to detect vehicles at a distance as seen in Figure 3.5c. The detections are overall found promising with high accuracy in the higher resolution front and rear perspective. The second detection stage has shown improvements in detecting vehicles in especially the side perspectives. Note this is at the expense of computation time. No further gains were found with higher upscaling. A more detailed explanation of the DPM detector is found in Section 7.1. The detections in the side perspective is still challenging despite the use of a second stage. The same detector is used in all six perspectives only changing the horizon parameter. Examples of DPM detections are shown in Figure 3.5.

(a) Front Left **(b)** Front **(c)** Front Right



(d) Rear Left **(e)** Rear **(f)** Rear Right

Fig. 3.5: DPM detections in all six perspectives. Detections with a score higher than zero is shown as bounding boxes. A red rectangle indicate detections from the first detection stage, and a blue bounding box indicates a higher score is found in the upscaled detection stage. Note the white pickup truck not being detected in (d) probably due to heavy distortion. The horizon is marked in (c) together with the top and bottom margin used in the second detection stage.

Vehicle Tracking

Using the detections from the previous module, the tracker objective is to associate detections between frames by assigning identification numbers (ID). Ideally every vehicle are assigned only one ID throughout the scene. To carry out this task the Markov decision process (MDP) tracker [61] is used (a detailed description of the tracker is available in Section 8.1), with freely available source code. Minor corrections are made as the MDP tracker is originally made for tracking pedestrians. Since vehicles have another aspect ratio than pedestrians the template size is re-defined using the KITTI dataset [20]. A histogram of annotations for the object challenge is seen in Figure 3.6, from where a mean of 1.5 is used as the new aspect ratio.The MDP tracker is applied to each individual view.

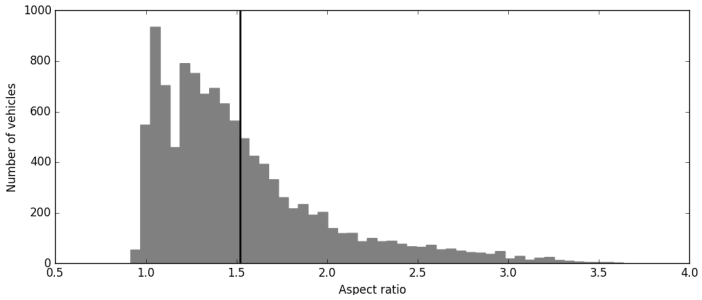


Fig. 3.6: Histogram of aspect ratio of annotated bounding boxes in the KITTI dataset [20].

Tracking and Association Between Perspectives

With vehicles being tracked in each individual perspective, the task of the *multi-perspective tracking* module is to link ID for the same vehicle across different perspectives. The association is done purely based on positions in the image plane, following a set of heuristic rules as described in Equation 3.1.

Each track consisting of an ID and detections is denoted $a_m^k = [ID, d_n]$, where a_m^k is the mth tracked vehicle in camera k and the detection $d_n = [t, x_1, y_1, x_2, y_2, s]$ are described with a time/frame index, t, and a bounding box with the top-left corner (x_1, y_1), bottom-right corner (x_2, y_2) and a detection score, s. The camera under evaluation is denoted $k \in [1, 2, \cdots, K]$ with K being the total number of cameras, in this case six. The cameras are known to overlap, from where pre-defined overlap regions are determined for each

perspective denoted $\Omega^k = [\Omega_L^k, \Omega_R^k]$ for overlap with respect to the left and right adjacent camera. These boundaries are displayed in Figure 3.7, with the overlap with the left adjacent camera marked by yellow and the right overlap by red. For an association to take place, a track must be present in one of the overlapping regions in the perspective in which it is currently tracked, and a corresponding track must be present in the adjacent perspective. Each trajectory is only evaluated once, being the first frame it appears. The bounding box of the new trajectory is firstly examined to be positioned in either the left or right overlapping region. A single pixel is sufficient to satisfy this first criteria. Secondly, the corresponding adjacent perspective is examined for possible candidates to be associated with. If a match is found in the adjacent perspective, this ID is carried over to new the trajectory as well, thereby associating tracks between perspectives. In the case with multiple possible matches in the adjacent perspective, a constraint is added, where an ID only can exist once in each perspective, or else the closest match is chosen.

$$b_l = \begin{cases} [a_m^k, a_{m'}^{k-1}] & \text{if } \Omega_R^{k-1} < a_{m'}^{k-1}(x_2) \text{ and } a_m^k(x_1) < \Omega_L^k \\ [a_m^k, a_{m'}^{k+1}] & \text{if } \Omega_R^k < a_m^k(x_2) \text{ and } a_{m'}^{k+1}(x_1) < \Omega_L^{k+1} \end{cases} \tag{3.1}$$

Associated trajectories between cameras are described as $\boldsymbol{B}^T = \{\boldsymbol{B}_{k,k\pm1}^T\}$ in the time interval T. Note that k wraps around, such that k_1 and k_K are adjacent perspectives. Each set of associations between two cameras k and $k \pm 1$ is $\boldsymbol{B}_{k,k\pm1}^T = \{b_1, b_2, \cdots, b_L\}$ where $b_l = [a_m^k, a_{m'}^{k\pm1}]$ is the lth association.

As an example, see Figure 3.7, where a silver vehicle cuts-in from the left in the front perspective seen from two time instances. The silver vehicle is correctly assigned the ID 4 from the left adjacent perspective. Note how a vehicle can be present in three perspectives simultaneously, as seen with ID 4 in Figure 3.7b. The fact that a vehicle can be present in more than two perspectives is a disadvantage of the setup. A mis-association is inevitable and will in this case be hard to correct.

Setting up heuristic rules for all case scenarios is a tedious task, and a less viable solution if this is to work in all generic scenarios found on-road. In such case other cues must be used to assist the association such as depth or appearance cues which probably requires color correction. However for the use in free-flow low-density scenes this naive implementation is found sufficient as a proof-of-concept.

(a)

(b)

Fig. 3.7: Track association of ID 4 from left-front to the front perspective seen over two time instances. The yellow and red vertical lines mark the overlap with the left and right adjacent camera respectively, which are used in the association in the image plane.

3.4 Trajectory Analysis

Given the persistent tracks across perspectives enables an NDS analysis to reduce sequences of driving into events for a fast interpretation. Ten sequences of real-world data are used in the evaluation with various overtakings and cut-ins. The ten sequences are described with 14 different events as illustrated in Figure 3.8. Events are registered from tracks depending on which cameras they have been spotted in. Inverse perspective mapping (IPM) is used in the front and rear perspectives to obtain the position of vehicles in front and behind the ego-vehicle (A detailed description of IPM is available in Section 9.1). The IPM is also used as a visualization tool to represent filtered vehicle trajectories in a top-down view. The trajectories are filtered using an average of the last n samples to give a more smooth trajectory.

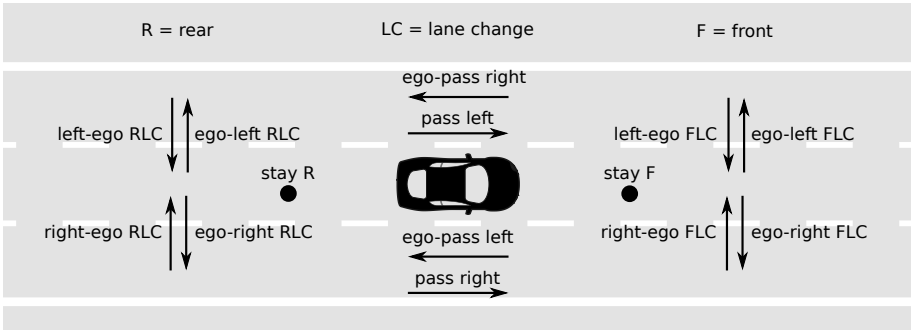


Fig. 3.8: The 14 events detected in the trajectory analysis.

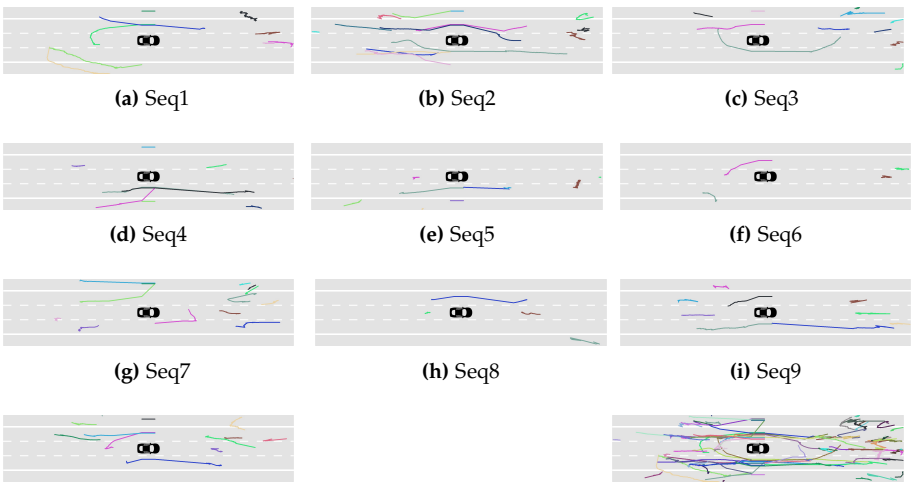
The events contained in the trajectories are compared to manually found ground truth for all ten sequences and summarized in Table 3.1 and Figure 3.9. The precision and recall is generally found to be high for both events and sequences. The two false negatives seen in sequence three and seven are most likely caused by the filtering, shortening the trajectories, resulting in a missing a lane change.

3.5 Concluding Remarks

Given six cameras, a full framework demonstrating the concept of full surround tracking has shown to be a possibility, with several experiences along the way. Each association between perspectives is a risk of error of why

Table 3.1: Events detected by the system for all ten sequences compared to ground truth [System/GT].

Event	Seq1	Seq2	Seq3	Seq4	Seq5	Seq6	Seq7	Seq8	Seq9	Seq10	Precision	Recall
Stay front	2/1	0/1	1/1	1/0	0/1	1/1	1/1	1/1	1/1	1/1	0.78	0.78
Stay rear	0/0	0/0	0/0	0/0	1/1	0/0	0/0	1/1	1/1	0/0	1.00	1.00
Pass on left	3/4	3/4	1/2	1/1	0/0	1/1	3/3	1/1	0/0	3/5	1.00	0.76
Pass on right	0/0	1/1	1/1	0/0	1/1	0/0	0/0	0/0	0/0	1/1	1.00	1.00
Ego-pass on left	0/0	0/1	0/1	4/4	1/1	0/0	0/0	0/0	1/1	0/0	1.00	0.75
Ego-pass on right	0/0	0/0	1/1	0/0	0/0	0/0	0/0	0/0	1/1	0/0	1.00	1.00
In front, left to ego-lane	1/0	2/1	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1/1	0.50	1.00
In front, right to ego-lane	0/0	0/0	1/1	0/1	1/1	0/0	1/1	0/0	0/0	0/0	1.00	0.75
In front, ego-lane to left	1/0	0/0	0/1	0/0	0/0	0/0	0/1	0/0	0/0	0/0	0.00	0.00
In front, ego-lane to right	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1.00	1.00
In rear, left to ego-lane	0/0	0/0	1/1	0/0	0/0	0/0	0/0	0/0	0/1	0/0	1.00	0.50
In rear, right to ego-lane	1/1	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1.00	1.00
In rear, ego-lane to left	1/1	0/0	0/0	0/0	0/0	1/1	0/0	0/0	0/0	1/1	1.00	1.00
In rear, ego-lane to right	0/0	1/1	0/1	0/0	0/0	0/0	1/0	0/0	0/0	0/0	0.50	0.50
Precision	0.7	0.88	1.0	0.83	1.0	1.0	0.83	1.0	1.0	1.0		
Recall	0.88	0.7	0.6	0.83	0.8	1.0	0.83	1.0	0.8	0.78		



(a) Seq1 **(b)** Seq2 **(c)** Seq3

(d) Seq4 **(e)** Seq5 **(f)** Seq6

(g) Seq7 **(h)** Seq8 **(i)** Seq9

(j) Seq10 **(k)** Total

Fig. 3.9: Visualization of the ten sequences along with all the trajectories in total. Evaluated in Table 3.1.

the number of cameras should be kept at a minimum. Detections of truncated vehicles near the image border is found to be of increased importance to ease the association between views. Likewise would a tracker prolonging by predicting the tracks close to the border aid the association between views. Association in the image plane is found sufficient for low-density scenes without multiple vehicles in the overlapping areas. Further information such as depth would be preferred to make a stronger association, or by calibrating the cameras to associate in a common road-plane. The trajectory analysis of 14 simple events were done using a number of heuristic rules. If more or more advanced events are to be analyzed it might be worth considering a data-driven approach instead writing tedious rules for each case.

An alternative and untested framework is to stitch all the images as preprocessing, which would circumvent the problem of truncation and association between perspectives. The stitching is assumed to be unfeasible given only a slight overlap used in this setup, but has not been further investigated.

Part III

Multi-Perspective Tracking and Trajectory Analysis

Summary

This part describes the final setup used with several improvements compared to the previously presented system in proof of concept. The setup consists of four GoPro cameras for a full surround coverage. Vehicles are detected in each separate perspectvive by a DPM detector, and tracked by a MDP tracker. The tracker is optimized for tracking in multiple perspectives. Tracks are projected to a common road plane using IPM. The tracks are Kalman filtered in the road plane followed by association to obtain trajectories going 360° around the ego-vehicle.

Detection, tracking and multi-perspective tracking are evaluated on 33 seconds of annotated data using various metrics. Trajectories from 50 sequences are analyzed classifying both supervised and unsupervised. Lastly an online classification is shown, examining how early a lane change can be recognized.

The work presented in this part has so far led to two papers. The journal paper in Appendix B is currently under review for IEEE Transactions on Intelligent Vehicles, titled *Trajectories and Behaviors of Surrounding Vehicles Using Panoramic Camera Arrays*. The paper in Appendix C marks the publication of the dataset used in this work to be submitted to IEEE Intelligent Transportation Systems Conference, titled *Vision for Intelligent Vehicles & Applications (VIVA): Multi-Perspective Vehicle and Trajectory Challenge*. The dataset is further to be presented at the IEEE Intelligent Vehicle Symposium in Gothenburg 2016 as part of the VIVA workshop.

Chapter 4

System Overview

This chapter gives an overview of the setup and system used in the remaining report. The setup is based on the gained experience from the previous setup. One of the findings were to minimize the number of cameras used to reduce the risk of error in transitions between perspectives. A total of four cameras are therefore used contrary the six cameras. New GoPro cameras have been acquired to increase the image resolution and overall image quality.

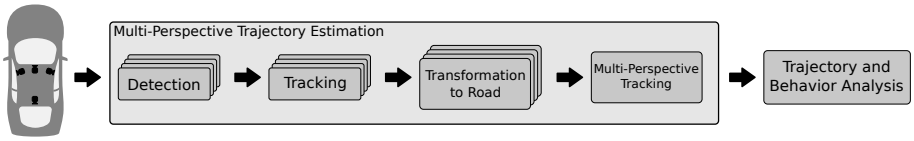


Fig. 4.1: Flow diagram of the final system using four GoPro cameras. Vehicles are detected and tracked in each individual perspective, and projected to a common road plane. This produces trajectories in all the way around the ego-vehicle to explore on-road behaviors.

The system flow is illustrated in the diagram of Figure 4.1. The detection module detects vehicles in all four individual perspectives. The tracker associates detections over time in the image planes, and transforms the tracks from all four perspectives to a common road plane using homographies found during calibration. This allows tracks to be associated in the road plane, being the key difference from the system used in the proof of concept. Several modifications are made to optimize the MDP tracker towards multi-perspective association. An extensive data collection process has been initiated revealing trajectories of surrounding vehicles to be further studied.

Chapter 5

Camera Setup

This chapter is devoted to all the practical considerations when working with a multi-camera setup. This includes the choice of camera, mounting on an actual testbed, and the post-processing involved, correcting for effects as mis-synchronization and distortion.

5.1 Choice of Camera

The Point Grey cameras used in the proof of concept are replaced with new GoPro HERO3+ cameras with a better image quality. Their field of view (FOV) is found to be wide while only slightly distorting the image and thereby seen as a considerable upgrade, despite the lack of synchronization of multiple cameras.

The GoPro HERO3+ is able to capture video of up to 4K resolution (3840x2160) at real-time frame rates. This is considerable better than current datasets, e.g. KITTI that uses 1392×512. However, the full resolution of the GoPro is not utilized since the added resolution also requires more computation and storage. Furthermore the battery consumption is higher at higher resolutions, not being able to capture for as long time.

A resolution of 2.7K (2704x1440) is chosen to maximize the horizontal field of view (125°) according to the datasheet[1]. This enables raw capturing 2704×1440 images at 24 FPS with a battery time of approximate two hours. The raw video is later downsampled to 12 FPS for efficiency.

[1] `https://gopro.com/support/articles/hero3-field-of-view-fov-information`

5.2 Camera Placement

Theoretically, only three GoPros are needed to cover 360° using a horizontal FOV of 125°. However, this assumes all cameras being located at the center of the vehicle which would truncate vehicles close by. In practice the cameras are mounted in outer positions of the roof rails as seen Figure 5.1, where four GoPros are used to obtain a full surround with overlapping views. An even number of cameras are preferred to have a designated front and rear view.

Fig. 5.1: Configuration of four GoPros mounted on the test vehicle to achieve a full surround view.

A wide selection of GoPro housings and mounts are available. The two side view cameras are mounted with a higher degree of freedom to be oriented outwards. The front and rear are mounted directly on the clamp to give the most stabilization which is found a necessity on U.S. highways. All cameras are mounted in an open frame with direct USB accessibility for data extraction without removing the cameras. This housing is thus susceptible to rainy weather.

5.3 Synchronization

The main disadvantage of using GoPros is the lack of synchronization between multiple cameras. The synchronization is not needed to be perfect down to a single frame in this application, but examined to get the best possible out of the setup. The frame rate is found constant for all GoPros without

drift, thereby no need for time aligning frames between cameras. This makes the synchronization a question of only determining an offset for each camera.

Each GoPro has an internal clock which is used to timestamp each frame. These clocks can be set one at time through the GoPro app, to synchronize each clock with a smartphone. This synchronization method has shown unreliable results, with up to several frames offset between cameras. The clocks are also shown to drift over time longer periods of time, requiring re-synchronization before each use. Another approach is to use the GoPro Wi-Fi remote. This allows to start and stop capturing of all connected GoPros with the push of a single button. This method is found surprisingly accurate down to only a few frames offset or even none. The synchronization is evaluated and compensated for by manual visual inspection of actions in the overlapping regions.

5.3.1 Synchronization of Velocity Data

The data gathered are not only images but also velocity of the ego-vehicle. This is used to obtain absolute velocities instead of only relative. Absolute velocities can be used in the analysis to determine if surrounding vehicles overtaking the ego-vehicle are speeding, or if the ego-vehicle simply drives slowly.

The ego-velocity is logged from the on-board diagnostics (OBD-II) port located under the steering wheel using an OBD-II-dongle. This allows reading of various vehicle kinematics as vehicle speed, engine speed, engine load etc. through a Python API [2] to a PC. The ego-velocity is sampled at approximately five Hz, and timestamped using the PC's local time. These timestamps are used to synchronize the velocity data to the image data. The image data are likewise timestamped, but not necessarily using the same clock, of why an offset is determined by manual inspection. Every image timestamp is matched with the closest velocity timestamp searching both back and forward in time. This procedure is done once for every drive since the GoPro clocks are found to drift over long periods (between drives).

[2] `http://openxcplatform.com`

5.4 Calibration

In the following is the intrinsic calibration and undistortion of the GoPro cameras described using the checkerboard calibration software in OpenCV.

The GoPro cameras suffer from barrel distortion which warps the image as seen in Figure 5.2a, where the crash barrier seems curved. The effect is more severe far from the center as seen near the image boundary. This affects the succeeding modules such as the detector, which is trained to recognize appearances. Detections near the image boundary are of special interest when dealing with association between perspectives, which makes undistortion relevant for this project. Alternatively to undistortion, one could train a detector on distorted images.

(a) Original Distorted **(b)** Undistorted

Fig. 5.2: Distortion sample of vehicles in the left perspective.

The effect can be reduced by calibrating the intrinsic parameters, which can be used to "straighten" out the image. The intrinsics are determined using a checkerboard of known size as the one best seen in Figure 5.4, being 8×6 with quadratic squares of 107 mm. Intersections of each square are found in the image plane and compared to the known 3D spatial distribution of the checkerboard, to find the transformation to unwarp the image. A total of 10+ frames is recommended with distributed checkerboards throughout the entire image at varying depth for the best result. The found checkerboards used for the calibration in this project are illustrated in Figure 5.3 using an excess of 72 sample images. Note our calibration is performed offline using recorded videos since GoPro got limited streaming capabilities. This complicates the procedure of gathering well distributed board positions throughout all parts of the image, without the feedback of when a checkerboard is found.

The found checkerboards are used to determine the calibration matrix

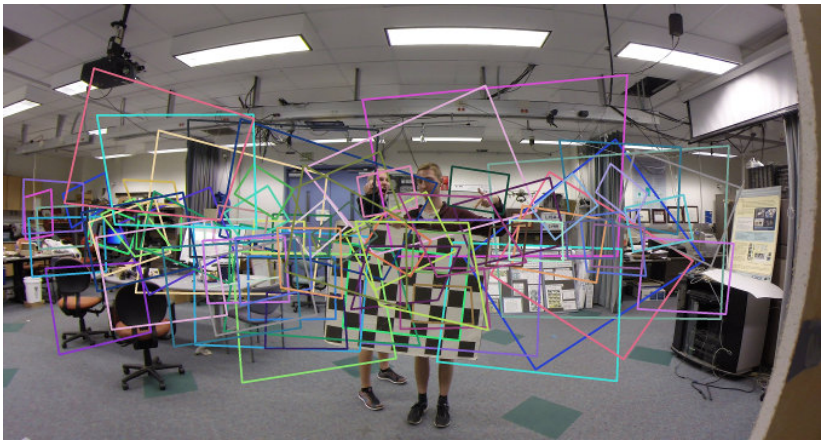
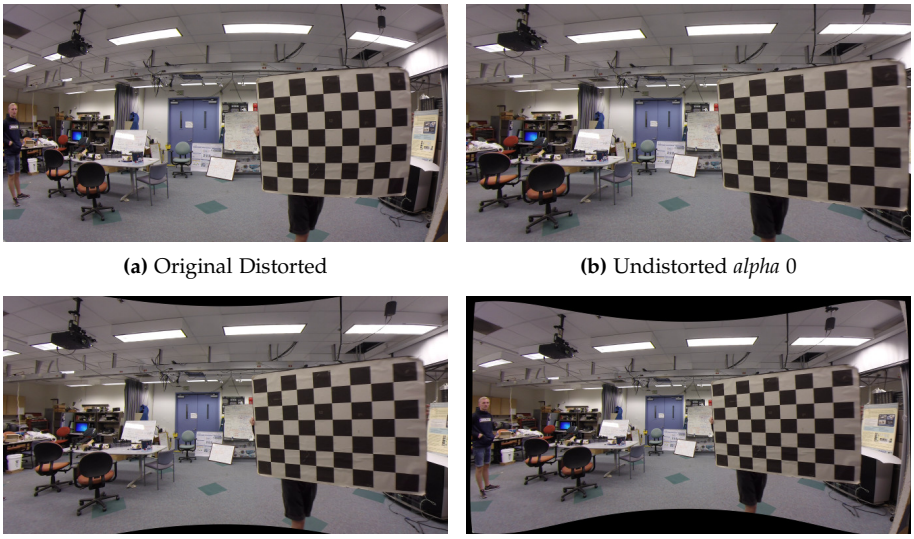


Fig. 5.3: Checkerboard distribution used to calibrate the intrisic parameters of the GoPro.

and distortion coefficients, which are necessary to unwarp the image. This process is performed for all four GoPros with almost identical results, which verifies the manufacturing standard of the GoPros. The calibration matrix is refined using the scaling parameter *alpha* to determine the level of unwarping as seen in Figure 5.4. A value of zero gives no invalid black pixels, while using a value of one will contain the entire original image. It is noted that the object in the center of the image becomes smaller using a higher *alpha* value. The checkerboard of the undistorted image is seen to have straight lines, as for the lines of ceiling. There might even be tendency to overly stretch the image at the boundary, which might be caused by the lack of checkerboards in that particular area. A *alpha* value of 0.4 have been chosen as compromise between loss of width and the size of objects.

The distorted image is often cropped in order to remove the redundant black regions or in order to define a region of interest i.e. avoid processing of sky pixels or the hood/trunk of the ego-vehicle. Approximately 10% of the images in the side perspectives can be removed, and up to 40% for the front and rear. No cropping is used in the processing of images in this project, only for visualization, and is therefore seen as a potential speed up.



(a) Original Distorted

(b) Undistorted *alpha* 0

(c) Undistorted *alpha* 0.4

(d) Undistorted *alpha* 1

Fig. 5.4: Different levels of unwarping by varying *alpha*. A value of zero contains no black pixels, where a value of one wraps the entire image to fit in the original size.

Chapter 6

Dataset

An important part in the development and evaluation of a system is the data, and it is crucial that the data fit the purpose of the study. Several datasets are publicly available for vehicle detection and tracking, but none of these focus on a full surround view. Parts of the system presented in this work can potentially be evaluated using the common benchmarks such as the KITTI Vision Benchmark Suite [20]. However, in order to fully expose the performance of the system, full surround data are needed.

All data used in this work are collected with the intention of developing and evaluating the system, and eventually parts of the data will be made available to the research community as part of the Vision for Intelligent Vehicles & Applications (VIVA) challenge. The data are gathered using the testbed described in Chapter 5 with help from the LISA lab to drive the vehicle. The database is collected on U.S. highways in southern California over 2.5 hours of driving. The drives are divided into 50 sequences consisting of challenging behaviors found around the ego-vehicle including e.g. overtaking, cut-ins, and cut-outs. The sequence duration vary around 30-60 seconds, to be able to capture one or multiple such events. No lane changes of the ego-vehicle are performed, and only flat sections are used. The dataset does not include any training data, but methods evaluated on the KITTI benchmark prove to work on the data as well, for which reason it is suggested to use the large amount of publicly available training data. In this work it is further possible to use the data collected in the proof of concept system (see Part II) for training.

6.1 Ground Truth Annotations

Ground truth is annotated in order to evaluate methods on the dataset. The ground truth is obtained for each of the four perspectives by manually annotating bounding boxes as shown in Figure 6.1 in the format as seen in Equation 6.1, where (x_1, y_1) denotes the top-left corner, and (x_2, y_2) denotes the lower-right corner.

$$[frame, id, occlusion, truncation, x_1, y_1, x_2, y_2] \tag{6.1}$$

Each vehicle is assigned an identification number, *id*, to evaluate tracking. Note the *id* is consistent between perspectives giving the option of multi-perspective tracking.

Fig. 6.1: Sample images from one of the sequences in the dataset with overlaid ground truth annotations at six different time instances. The bounding boxes are color coded by *id* and labled with *PO*, *HO*, *PT*, and *HT* denoting partial occlusion, heavy occlusion, partial truncation and heavy truncation, respectively.

The occlusion and truncation tags are both divided into three levels being *No*, *Partial*, and *Heavy*. Here, *No* equals 0%, *Partial* includes vehicles up 50%, while *Heavy* covers 50%+ for both occlusion and truncation. Three levels are chosen to simplify the annotation workload, while maintaining a certain division for analysis purposes. At the time of writing a single sequence of 33 seconds (400 frames) has been annotated. Note the annotations are done in all four perspectives, resulting in 4000+ annotations for this single sequence, worth two days of annotating using a customized Python annotator as seen

in Figure 6.2. This annotator is also used for annotating break and turn signals in [44].

Fig. 6.2: wxPython based annotator developed specifically to annotate this dataset. The red cursor is eases the drawing of bounding boxes, together with hotkeys for assigning IDs and labeling occlusion and truncation levels and several other features.

The bounding box annotations allow for evaluation of both detection and tracking. In order to evaluate 3D tracking, we use the estimated homographies to transform each ground truth trajectory to the road plane (further details for this procedure is described in Section 9.1). The middle of the bottom of the bounding box is used as vehicle position in the image plane. The road plane annotations have four entries:

$$[frame, id, x, y] \tag{6.2}$$

The transformed ground truth trajectories in the road plane are shown in Fig. 6.3. The road plane trajectories seem noisy which might seem unnatural since it is considered ground truth, and one might feel the urge to filter this result. The noise is caused by vibrations of cameras, which when transformed to the road plane is especially evident in the driving direction (the x-direction). As the same transformation is used for both the ground truth and the test trajectory under evaluation this error will be reduced, hence no filtering despite the noisy looking ground truth.

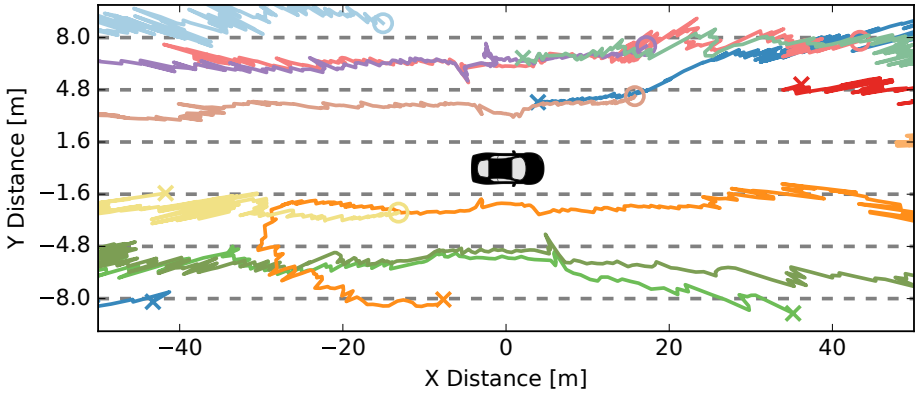


Fig. 6.3: Ground truth trajectories in the ground plane. The trajectories are marked with a start position `X` and an end position `O` to indicate relative velocity to the ego-vehicle.

6.1.1 Dataset Content

This section shortly describes the data obtained from the ground truth bounding box annotations. As seen from Figure 6.4 the data in the four perspectives vary notably in aspect ratio, average bounding box size, distribution of truncation level, and the number of perspectives in which the vehicles are present. Thus, vehicles in the side perspectives tend to have a larger aspect ratio, a larger size in both width and height, and a larger percentage of truncation compared to the front and rear perspectives. An important note is that all vehicles that appear in the side perspectives are visible in at least one other perspective during the sequence. A great motivation for tracking the vehicles all the way around the ego-vehicle appear from the bottom right histogram, which shows the number of perspectives in which vehicles are present. Note that almost half of the vehicles are present in three perspectives, and only about 40% stay in one perspective. Though, the data are chosen to challenge the multi-perspective tracking, this result clearly shows why it is interesting to follow vehicles all the way around the ego-vehicle.

6.2 Evaluation

The collected data with annotations allow for evaluation of multiple applications. In this section the methods used to evaluate vehicle detectors, single-perspective multiple object trackers, and 3D trackers, are described.

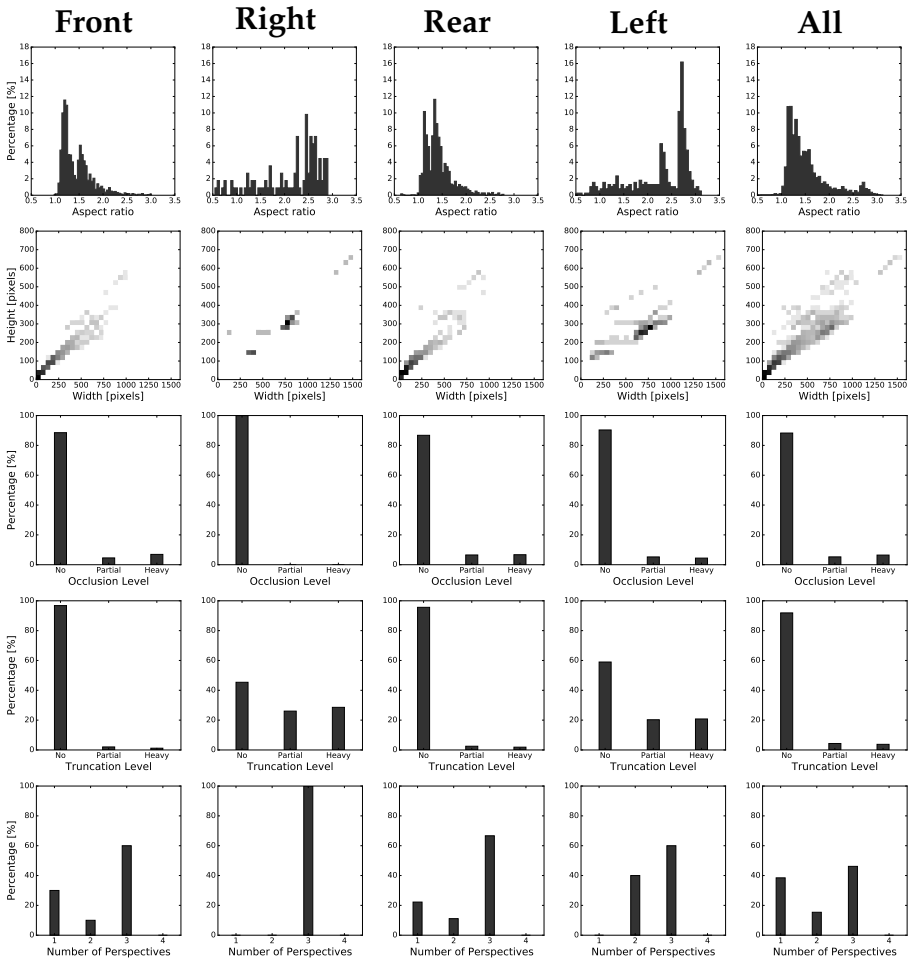


Fig. 6.4: Histograms of bounding box annotation measures. Each column represents a perspective, and the rightmost column is the total. The rows are from top to bottom: Bounding box aspect ratio, bounding box width and height in pixels, occlusion level of vehicles, truncation level of vehicles, and the number of perspectives in which the visible vehicles are present during the sequence.

6.2.1 Detection Metrics

Well-established metrics exist for both detection and single-perspective tracking. In detection the average precision (AP) is commonly used, calculated as the area under the precision-recall curve. Where:

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \tag{6.3}$$

With TP denoting the number of true positives, FP the number of false positives, and FN the number of false negatives. The precision-recall curve is generated for a detector by varying the threshold for the score (from very strict to very loose). For a ground truth bounding box to be matched with a detected bounding box, an overlap defined as the intersection over union is required to be at least 0.7 in this work.

$$\text{Overlap} = \frac{h_1 \cap h_2}{h_1 \cup h_2} \tag{6.4}$$

Where h_1 and h_2 are the two bounding boxes. The highway is a simple setting for testing of vehicle detectors, however vehicles driving in the opposite direction on the other side of the crash barrier might potentially be detected, though not of interest to this study. For this reason, ignore regions are defined in the perspectives, where these vehicles are visible.

6.2.2 Single-Perspective Tracking Metrics

The single-perspective multiple-object tracking is evaluated using the CLEAR MOT metrics (MOTA and MOTP) [2], together with metrics including fragmentations (Frag) and ID switches (IDS) [32], mostly tracked (MT), and mostly lost (ML) [59]. Multiple object tracking accuracy (MOTA) is defined as:

$$MOTA = 1 - \frac{FN + FP + IDS}{TP + FN} \tag{6.5}$$

Multiple object tracking precision (MOTP) is:

$$\text{MOTP} = \frac{\sum_{t,i} \text{Overlap}_{t,i}}{\sum_t c_t} \tag{6.6}$$

Where c_t denotes the number of matches in frame t, and $\text{Overlap}_{t,i}$ is the overlap between target i in frame t and the corresponding ground truth target. MOTP is thus an average precision of bounding boxes that have been matched with a ground truth box. A fragmentation is added every time a

ground truth trajectory is split. An ID switch is added if a ground truth trajectory is matched with another ID than the one that is currently associated. A ground truth trajectory is counted as mostly tracked if it is associated more than 80% of the time by definition. Likewise is a ground truth trajectory accounted as a ML if associated in less than 20% of the time.

6.2.3 3D Tracking Metrics

The field of multi-perspective 3D tracking is less explored without any common metrics. The problem, however, is not so different from tracking in single perspectives. For this reason, similar metrics are used in the road plane. Instead of association between ground truth bounding boxes and tracked bounding boxes, this will require association between points in the road plane. To this end, a weighted euclidean distance from ground truth trajectory points is used. This does however mean that the cost of matching a candidate to ground truth is not normalized (as the bounding box overlap definition), and thus MOTP is not well-defined ([33] suggest to normalize the distance with the matching threshold). The original definition of MOTP says that it is the average dissimilarity between all true positives and their corresponding ground truth targets. With bounding box overlap a score as close to 1.0 is ideal. Using euclidean distance changes the ideal score to be as close to zero as possible, as it is now defined as the average distance between true positives and their corresponding ground truth targets. To reduce the confusion, this score is referred to as multiple object tracking euclidean precision (MOTEP):

$$\text{MOTEP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{6.7}$$

Where c_t denotes the number of matches in frame t, and $d_{t,i}$ is the weighted euclidean distance between target i in frame t and the corresponding ground truth target. Another important aspect is the choice of matching threshold, which is traditionally chosen as either 0.5 or 0.7 bounding box overlap. A direct transfer to the road plane domain would be a static distance allowed from each ground truth trajectory point. However, as a nature of inverse perspective mapping small variations close to the ego-vehicle will not be as severe as small variations further away. It is proposed to make the matching criterion a function of the x-distance from the ego-vehicle. A target is matched to a ground truth target if it fulfills:

$$d_{t,i} < a|x| + b \tag{6.8}$$

where $|x|$ is the absolute x-coordinate of the ground truth target, a is the gradual increase in allowed distance, and b is the allowed distance at $|x| = 0$. Specifically, $a = 0.04$ and $b = 2$. It should however be noted that variations in the y-direction is more likely to cause erroneous matches. For this reason, the weighted euclidean distance is defined as:

$$d_{t,i} = \sqrt{(gt_x - tr_x)^2 + 4\left(gt_y - tr_y\right)^2} \tag{6.9}$$

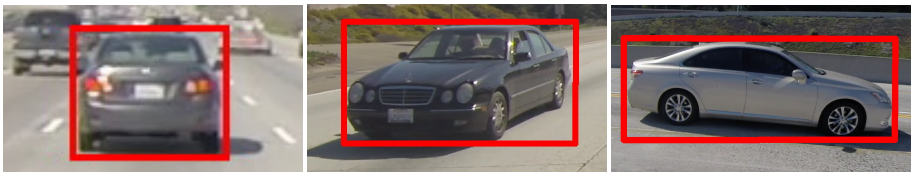
where $gt = [gt_x \; gt_y]^{\mathrm{T}}$ is the 2D ground truth position and $tr = [tr_x \; tr_y]^{\mathrm{T}}$ is the position of the target. Thus, distances in the y-direction have a double weight.

The metrics suggested above do however not encapsulate the importance of ID switches that happen between perspectives. For this reason, it is intended to include trajectory similarity measures in the future, which could e.g. be longest common subsequence (LCS).

Chapter 7

Vehicle Detection

The art of object detection is a well researched area applied in a wide variety of applications detecting faces, pedestrians, and lately vehicles, all sharing many of the same methodologies and detection schemes. The common goal is to find the object within the image, often marked by a bounding box as seen in Figure 7.1. The subfield of vehicle detection is in this project limited to highway scenarios which simplifies the problem compared to urban driving, which also has to take oncoming and parked vehicles into account, just to name a few.

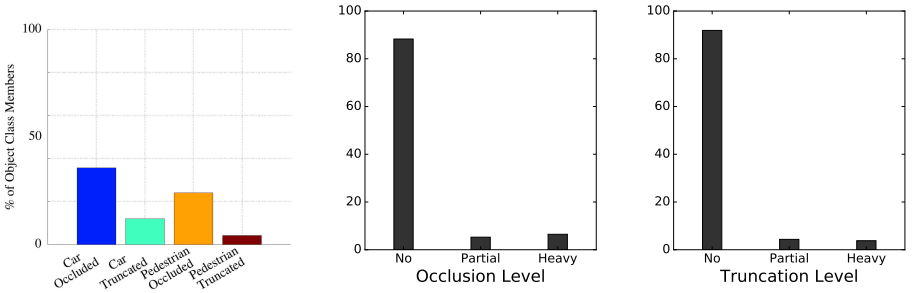


(a) Rear **(b)** Front Left **(c)** Right

Fig. 7.1: Bounding boxes marking vehicle objects in an image.

This project focuses on passive visual sensors, but other possible solutions include active sensors as radar and lidar, of a fusing hereof. Vehicle detection is especially challenged by a combination of varying viewpoint, a high intra-class variation, and frequent occlusions and truncations. The viewpoint depends on the application where highway applications simplify the situation to only a few orientations in each perspective (e.g. mainly rear-views for a front-faced camera).

The high intra-class variation is challenged by the vehicles found on roads such as cars, vans, and pickups, but also within each class with variation in shape, size, and color. The significance of occlusions and truncation is found in Figure 7.2 from the KITTI dataset to be approximate 35% occluded and 10% truncated. Note the KITTI dataset is mainly captured in urban and rural scenes thereby not directly comparable to this highway scenario. Since only 10% of vehicles are truncated these are often down prioritized, also not evaluated when being more than 50% occluded in the KITTI *Hard* difficulty.



(a) KITTI Dataset [19]. Approximate 35% occlusion and 10% truncation.

(b) Our dataset occlusion levels for all perspectives.

(c) Our dataset truncation levels for all perspectives.

Fig. 7.2: Comparison between the KITTI dataset with only front-faced perspective in mainly urban driving and the multi-perspective highway data.

Vehicles are typically occluded by other vehicles, and often found in both high and low density scenes. One might argue an occluded vehicle is of little importance to the ego-vehicle since there is a third vehicle in between. Another might say these are of special importance since these can appear in short notice due to being hidden, thus an early detection is important. Likewise is it of importance to detect truncated vehicles. Especially in a multi-perspective setup where detections near the image boundary are used to associate detections between perspectives. The truncation problem have in previous dataset been of less importance. As first introduced in the PASCAL Challenge [13] in 2007 as true/false flag, and later in KITTI [20] from 2012 as percentage by back projecting 3D annotated objects to the image plane. KITTI furthormore had three difficulties with *Hard* difficulty with up to 50% truncated objects.

Only static images are used in traditional detection evaluations, which heavily favors the use of visual cues only, opposed the goal of this application

of detecting and tracking vehicles in a continuous video feed. This allows the use of motion cues as seen in Figure 7.3, using optical flow near the image boundary for an early detection of overtaking truncated vehicles by looking at the direction of the optical flow. This approach might however be less suitable detecting receding vehicles but might be a good supplement for earlier detections, especially in the side perspectives. Note the moving platform excludes methods relying on background subtraction often found in surveillance applications.

Fig. 7.3: Early detection of vehicles near the image boundary using optical flow [17].

Among the most popular detection schemes are Viola and Jones [55] sliding-window search with cascaded classifiers, detecting objects in a very efficient manner. A likewise popular framework is the use of histograms of gradients (HOG) features with a linear support vector machines (SVM) as classifier. State-of-the-art detectors within the field include Aggregated Channel Features (ACF) [11], SubCat [41] and ConvNet. Their performance are found on the KITTI object evaluation of cars as seen in Table 7.1. For the full table see KITTI Homepage[1]. Some detectors also estimate the orientation of detected objects, where a similar table is found on the KITTI website scoring both detection and orientation estimation. The vehicle orientation is not further used in this work and is therefore ignored for the remainder of this work.

The choice of detector is in this work a compromise between implementa-

[1] `http://www.cvlibs.net/datasets/kitti/eval_object.php`

Table 7.1: Selected detection performances on the KITTI object evaluation dataset for cars. Detectors are ranked according to AP on the moderate difficulty.

Rank	Method	Moderate	Easy	Hard	Runtime	Environment
4	SubCNN	89.32 %	90.86 %	79.33 %	2.0 s	GPU @ 3.5 Ghz
9	Faster R-CNN	81.84 %	86.71 %	71.12 %	2.0 s	GPU @ 3.5 Ghz
11	Regionlets	76.45 %	84.75 %	59.70 %	1.0 s	>8 cores @ 2.5 Ghz
15	SubCat+HSC	75.46 %	84.14 %	59.71 %	5.5 s	2 cores @ 2.5 Ghz
16	SubCat	75.46 %	84.14 %	59.71 %	0.7 s	6 cores @ 3.5 Ghz
18	FCNN	70.67 %	87.69 %	61.49 %	0.1 s	1 core @ 2.5 Ghz
21	SubCat	66.32 %	81.94 %	51.10 %	0.3 s	6 cores @ 2.5 Ghz
22	OC-DPM	65.95 %	74.94 %	53.86 %	10.0 s	8 cores @ 2.5 Ghz
23	DPM-VOC+VP	64.71 %	74.95 %	48.76 %	8.0 s	1 core @ 2.5 Ghz
24	On-road CNN	63.08 %	77.50 %	52.52 %	0.2 s	GPU @ 1.0 Ghz
26	DPM-C8B1	60.99 %	74.33 %	47.16 %	15.0 s	4 cores @ 2.5 Ghz
27	ACF-SC	58.66 %	69.11 %	45.95 %	0.3 s	1 core @ >3.5 Ghz
28	LSVM-MDPM-sv	56.48 %	68.02 %	44.18 %	10.0 s	4 cores @ 3.0 Ghz
29	RCNN	55.97 %	68.20 %	46.70 %	10.0 s	GPU @ >3.5 Ghz
31	LSVM-MDPM-us	55.42 %	66.53 %	41.04 %	10.0 s	4 cores @ 3.0 Ghz
32	ACF	54.74 %	55.89 %	42.98 %	0.2 s	1 core @ >3.5 Ghz

tion time and performance. Implementation time is highly affected by if the framework is publicly available, or if training is necessary which requires annotated data preferably on our data or a similar database. The performance is in this project primarily measured on precision and recall, with limited consideration of real-time.

The DPM detector (LSVM-MDPM-sv from Table 7.1) is chosen as detector in this project based on its freely available code [22], and pre-trained vehicle models [18, 65], which has shown promising detection results, without the need of further optimization. The idea of deformable parts is appealing to combat both occlusions and truncations which is found to be of interest for multi-perspective tracking. The remaining of this chapter will be a further description of the DPM detector and its performance on our dataset.

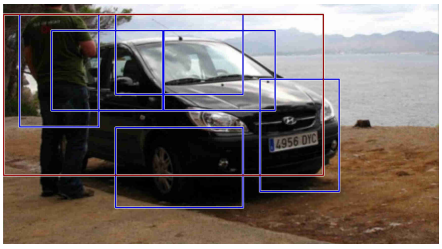
7.1 Deformable Part Models

The DPM [15][2] is a popular choice of detector for detecting viewpoint variant objects. Such as vehicles which change a lot along with viewpoint and intra-class variability. The DPM combats the viewpoint variation by using a mixture of models, training a model for each viewpoint. Furthermore parts are introduced to handle intraclass variation by allowing the model to "stretch",

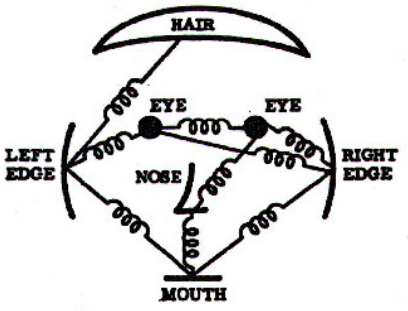
[2] `https://people.eecs.berkeley.edu/~rbg/latent/`

where detection of parts can help in occluded or truncated cases.

The DPM is based on the findings of Dalal and Triggs [8], showing the effect of training a linear SVM using HOG features and detecting objects with a sliding window method, which can be seen as a template matching of HOG features. This approach outperformed all previous approaches by a large margin on the Pascal VOC challenge. The DPM extends the system of Dalal and Triggs by introducing local parts to the global root detection window. An example is seen in Figure 7.4a where the root is depicted in red and parts in blue. The parts are organized in a pictorial structure, connecting the parts in a "spring-like" configuration as seen in Figure 7.4b. This allows the parts to move with respect to the root within certain limits i.e. the distance between parts are deforming. Note the model is not modeling interaction between parts, but only between root and parts.



(a) A DPM detection with the root filter shown in red and parts in blue.



(b) Pictorial structure connecting face parts together in a "spring-like" configuration.

Fig. 7.4: DPM detection and pictorial structure.

The performance of the DPM detector has made a remarkable impression on especially the Pascal VOC challenge dataset, detecting between 20 different objects in a wide variety including cars, persons, horses, chairs etc. An evaluation of the Pascal VOC challenge 2006 cars dataset is shown in Figure 7.5 by the original DPM authors Felzenszwalb et al.. This shows the impact of using a mixture of two models compared to one, and also the use of parts. Lastly is the slight performance in precision by using prediction of the bounding box. The DPM is further explained by its two key components, detection and training.

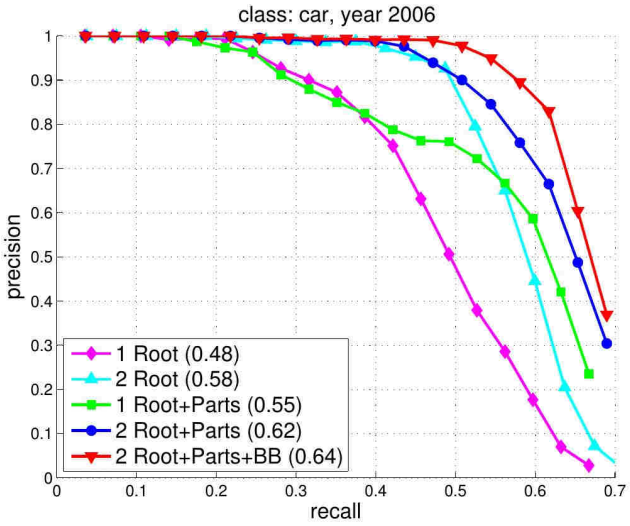


Fig. 7.5: Performance of DPM on Pascal VOC challenge [15]. AP in parentheses.

7.1.1 DPM Detection

The DPM detection scheme describes the process of finding and scoring detection candidates i.e. hypotheses, in an image. Objects at different sizes and distances are detected using a feature pyramid. For each frame, a feature scale pyramid is built from HOG appearance features by smoothing and subsampling to detect objects at different scales as shown in Figure 7.6.

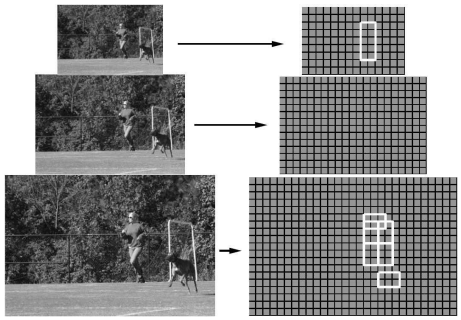


Fig. 7.6: A HOG feature pyramid build for each frame by smoothing and subsampling. The root filter is scanned through the entire image, while part filters are applied at twice the resolution [15].

The DPM detection framework is applied to each level in the feature pyramid for each component in the pre-trained mixture model. A model contains

a root filter, n part filters, and deformation costs for each part filter as seen in Figure 7.7. The training of such a model is found in the next section, and the model is assumed given for now.

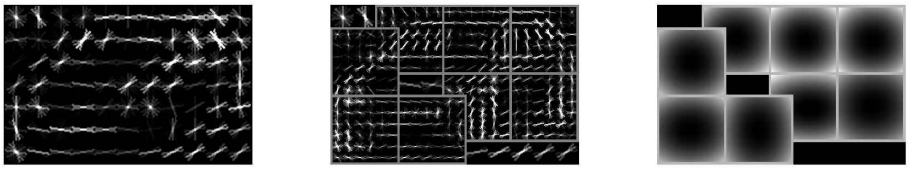


Fig. 7.7: One component of a mixture model. From left to right: root filter, eigth part filters, and deformation costs [63].

The DPM detection framework for a single level in the feature pyramid is visualized for a person in Figure 7.8. The method is directly applicable to other objects as vehicles using a different model. Each level is scanned by the root filter, while the part filters are applied at twice the resolution. The response of the part filters are smoothened to account for spatial uncertainty by a distance transformation using the deformation costs.

Each hypotheses is scored using Equation 7.1. Here the sum of the root filter ($i = 0$) and the part filters are summed and subtracted with the deformation costs.

$$\text{score}(p_0, \cdots, p_n) = \sum_{i=0}^{n} F_i' \cdot \phi_v(H, p_i) - \sum_{i=1}^{n} d_i \cdot \phi_d(dx_i, dy_i) + \text{bias} \qquad (7.1)$$

where

F is the filter parameters. F_0 being the root filter, $F_{1,\cdots,n}$ being part filters.
$\phi_v(H, p_i)$ is a subwindow of the feature pyramid H at the p_i position.
d_i is the part filters deformation costs.
$\phi_d(dx_i, dy_i)$ is the parts displacement feature vector from anchor position.
bias makes scores comparable across components in the mixture model.

The parts displacement is calculated based on the deviation from the anchor point (v_i) defined relative to the root filter (x_0, y_0) as seen in Equation 7.2. Note the root filter is scaled by a factor of two to match the part filters position found in twice the resolution of the feature pyramid. The displacement is used in the quadratic function of Equation 7.3 to define deformation costs when deviating from the anchor point. The quadratic function gives the

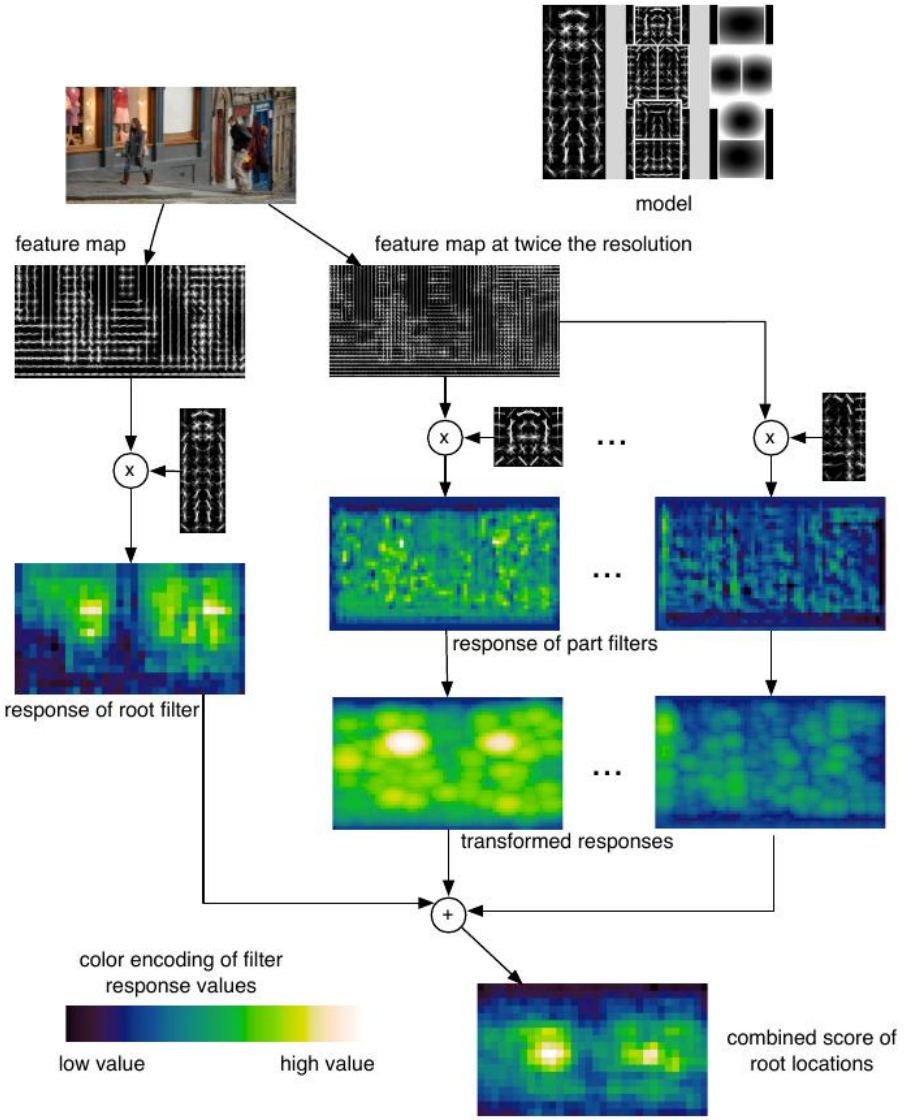


Fig. 7.8: DPM detection scheme a single level in the feature pyramid using a person model. Every position of the root filter is scored as a combination of the root filter and the part filters found in twice the spatial resolution. [15].

circular shape of the deformation cost seen in the model in Figure 7.7.

$$(dx_i, dy_i) = (x_i, y_i) - (2\,(x_0, y_0) + v_i) \tag{7.2}$$

$$\phi_d(dx, dy) = (dx, dy, dx^2, dy^2) \tag{7.3}$$

Only the hypotheses with the highest confidence score is kept for each placement of the root filter as seen in Equation 7.4. Again, the score is a combination of both root and part filters. Since each part is modeled independently of each other, the optimal placement can also be found independently, which is solved in an efficient manner using dynamic programming.

$$\text{score}(p_0) = \max_{p_1,\ldots,p_n} \text{score}(p_0, \cdots, p_n) \tag{7.4}$$

After having traversed the entire HOG feature pyramid, scoring detection hypotheses at different scales, we are often left with multiple detections for each object. A non-maximum suppression (NMS) is therefore applied, by first sorting all detections by score, removing detections which overlap a higher detection by more than an overlapping threshold. The overlap is determined by the intersection over union between the bounding box of two hypotheses as described in Equation 6.4.

Specific for the `voc-release4` code used in this work is the use of bounding box prediction. The prediction refines the existing bounding box by training a linear least-square regression model for each component in the mixture model. The parts found with greater spatial precision is thereby used to predict a more accurate bounding box as seen in Figure 7.9. This has shown improvements for some classes in the Pascal VOC challenge, as for the *car* class as previously shown in 7.5.

Fig. 7.9: DPM detection bounding box before and after prediction.

7.1.2 Training DPM using Latent-SVM

The question of how to actually train a parts model still remains. The challenge consists of automatically learning parts location while only the root is annotated. Each model in the mixture model can be trained individually. The model parameters to be determined consist of a root filter, part filters, deformation costs and a bias as seen in Equation 7.5 using the SVM object function in Equation 7.6.

$$\beta_k = (f_{k0}, \cdots, f_{kP}, d_{k1}, \cdots, d_{kP}, b_k) \tag{7.5}$$

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \tag{7.6}$$

The input are annotated images e.g. the weak-labeled Pascal VOC dataset with annotated bounding boxes and class labels. Weak-labeled since no component labels or part labels. A bounding box is marking not only the object of interest, but also a considering amount of background. The training data is divided among the components in the mixture model by aspect ratio. Training data containing vehicles facing left and right will thereby be grouped together, if no further information is provided. This can be somewhat prevented by examining mirrored versions to divide left and right orientations.

The target test set distribution to be evaluated is imbalanced, in the way that the scanning windows will mainly consist of non-vehicles. The training data should roughly replicate the same distribution, thereby an overweight of negativ training samples. The SVM fits this problem by its sparse approach, only considering the support vectors which contributes the most. Furthermore a Latent-SVM (LSVM) must be used given the parts locations are unknown relative to the root.

The DPM uses a hardcoded number of parts, which is first initialized based on the energy of the root filter. A part is either anchored along the central vertical axis of the root filter, or it is placed off-center with a symmetric counter part as seen in Figure 7.10. Using symmetry the number of parameters to be learned can be reduced by half. The initialization is important since the optimization are susceptible to local minima based on the semi-convex nature of the LSVM. The initializing model is trained using the training data and fixed parts placement using regular SVM. A random part is placed on the highest energy spot, and a second random part are placed on the second highest energy spot and so forth. This initialization is tried for all possible combinations returning the configuration which covers the

maximum amount of energy.

A two step iterative algorithm is used to train the complete latent model containing the hidden part locations and deformation costs. The algorithm alternates between using the current model to find high scoring hypotheses, and using the found hypotheses to optimize the model parameters. The algorithm is shown in Algorithm 1.

Algorithm 1 Training LSVM, [15].

1: $F_n := \emptyset$
2: **for** *relabel* $:= 1$ **to** *num-relabel* **do**
3: $F_p := \emptyset$
4: **for** $i := 1$ **to** n **do**
5: Add `detect-best`(β, I_i, B_i) to F_p
6: **end for**
7: **for** *datamine* $:= 1$ **to** *num-datamine* **do**
8: **for** $j := 1$ **to** m **do**
9: **if** $|F_n| \geq$ *memory-limit* **then**
10: break
11: **end if**
12: Add `detect-all`$\big(\beta, J_j, -(1+\delta)\big)$ to F_n
13: **end for**
14: $\beta :=$ `gradient-descent` $\big(F_p \cup F_n\big)$
15: Remove (i, v) with $\beta \cdot v < -(1+\delta)$ from F_n
16: **end for**
17: **end for**

The outer loop determines the number of iterations. Followed is the first step in line 3-6, which in the first iteration uses the initialized model β to score hypotheses using the detection scheme described in Subsection 7.1.1. The best scoring detections are stored, if they also overlap with the ground truth bounding box B for the given image I. The positive examples are used in step two, together with hard negatives to optimize β in the latent SVM objective function. The next iteration will use the new β and the algorithm continues to next iteration.

The training algorithm is used for each component in the mixture model independently. The trained part models in Figure 7.10 is seen to be more detailed with the wheels being more significant. Furthermore, the deformation cost is found to be tighter. A better optimized solution could be found by

searching over a number of different mixture components and parts.

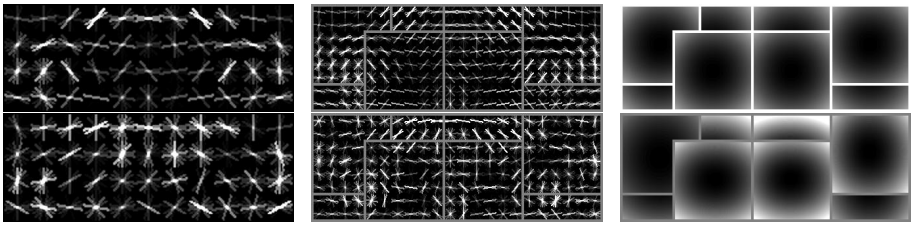


Fig. 7.10: Top row, initialized models. Bottom row, trained models. [21].

7.2 Implementation and Evaluation

This section describes implementation details of the DPM detector and an evaluation is made and compared to the SubCat detector.

The implemented version of DPM by Geiger et al. [18, 65] differs from the original DPM implementation [14] in several ways. It is worth noticing the Geiger et al. implementation is based on the KITTI dataset, "only" containing cars, pedestrian, and bicycles, compared with the 20 classes of the Pascal VOC dataset of which the original DPM implementation is aimed at. Furthermore, the annotations of the KITTI data contain orientation which is used to divide the training data, opposed to only aspect ratios of bounding boxes. The vehicle model used in the Pascal VOC Challenge consisted of two mixture models with each six parts, in comparison the KITTI trained eight mixture models with each eight parts. The Geiger et al. implementation is furthermore expanded to two detection stages for better recognizing vehicles further away.

Table 7.2: DPM parameters used unless else is specified. All values are default values.

Description	Value
Levels in the HOG feature pyramid	10
Components in the Mixture Models	8
Parts in each component	8
Non-Maxmimum Suppression Overlap	0.2
Minimum Score Threshold	-1

The first stage functions as a regular DPM on the entire image, while the second stage operates on an up-scaled version in an area around the specified horizon. This second stage is not used with high resolution GoPro data. The DPM parameters used in this evaluation is summarized in Table 7.2.

Qualitative results are shown in Figure 7.11 for all four perspectives. The detections are further thresholded, only using detections with a score above zero to remove most false positives.

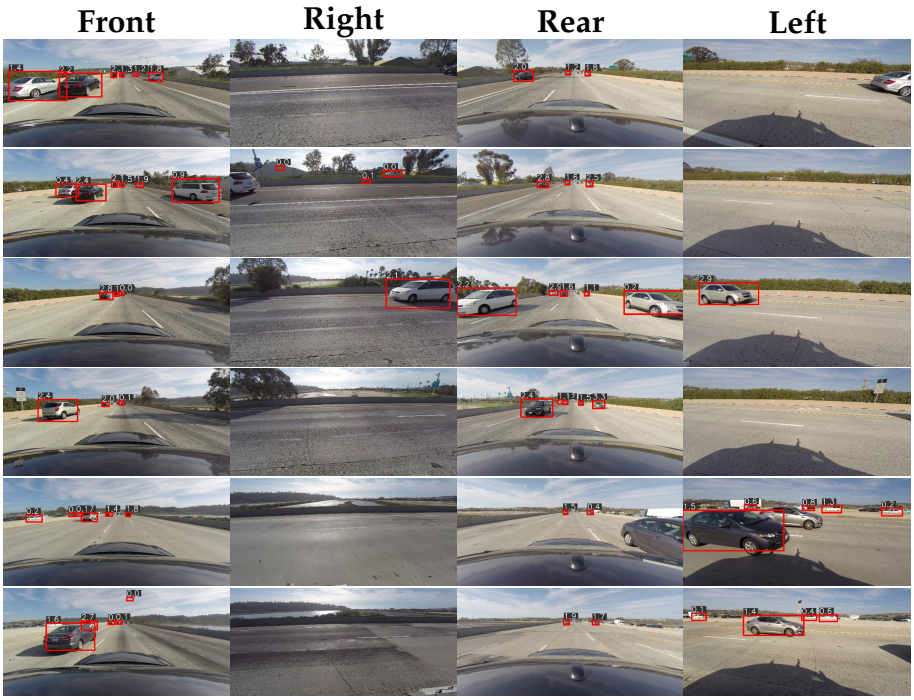


Fig. 7.11: DPM detections in the four perspectives.

The qualitative results are difficult to evaluate over entire sequences. A quantitative evaluation is therefore performed on the annotated sequence with ground truth using Dollárs `MATLAB` toolbox [10] to calculate precision-recall curves. The precision-recall curves are calculated by varying the detection score threshold against ground truth bounding boxes. An overlap of 0.7 is commonly used for vehicles, and 0.5 for all other objects. The AP is commonly used as a detection metric found as the area under the precision-recall curve. The quantitative results are found in Figure 7.12 and Table 7.3. The DPM is also tested using a faster setting with only a single level in the fea-

ture pyramid. The DPM is furthermore compared to state-of-the-art SubCat detector.

All detectors are found to perform considerable better in the front and rear perspectives compared to the side perspectives. A remarkable drop is seen between using an overlap of 0.5 and 0.7, where DPM is found to be less precise than SubCat, and even more so using only one level in the feature pyramid. Note the SubCat graphs are not fully complete which indicates a too high threshold cap. Remember, all detectors are trained on the KITTI database consisting mainly of urban scenes in Germany, and tested on our data from Californian highways.

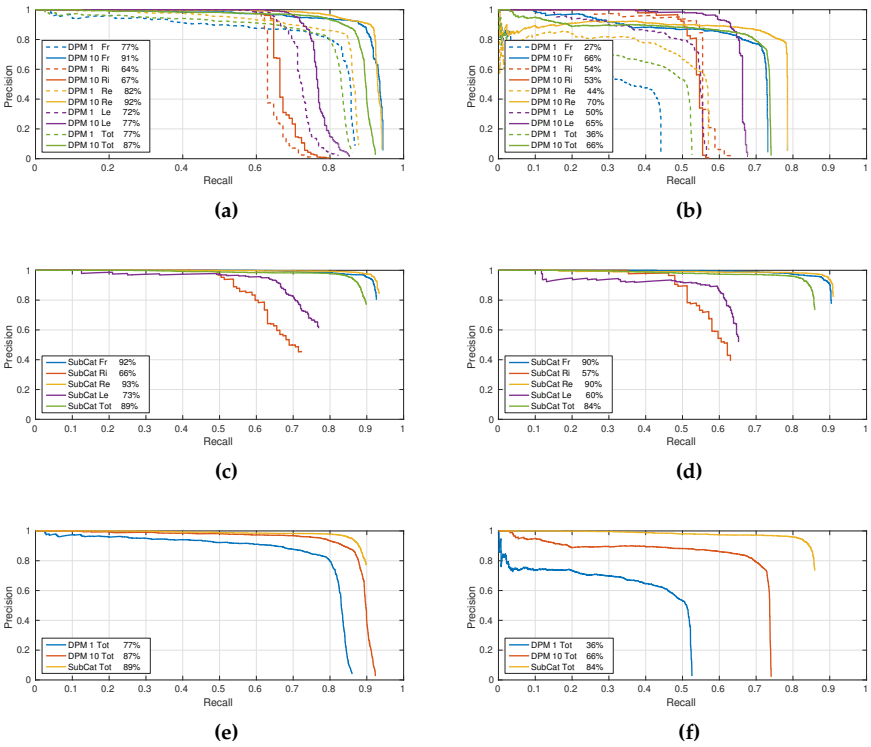


Fig. 7.12: Precision-recall curves for the detectors in the four perspectives and in total. Left column overlap criterion of 0.5. Right column overlap of 0.7. AP is included in the legends.

The average precisions are summarized in Table 7.3 together with runtimes for each of the three evaluated detectors. The DPM is generally found to be slow with a runtime up to 60 seconds per frame, or 10 seconds in what is

Table 7.3: Performance of the detectors [DPM 1 Level/DPM 10 Levels/SubCat].

Perspective	**Runtime** [s]	**AP 50% Overlap[%]**	**AP 70% Overlap[%]**
Front	10/61/1	77/91/92	27/66/90
Right	10/60/1	64/67/66	54/53/57
Rear	10/60/1	82/92/93	44/70/90
Left	10/59/1	72/77/73	50/65/60
Total		77/87/89	36/66/84

considered fast mode. Newer DPM implementations with focus on real-time have reported run-times of 10 FPS on 640×480 resolution images [62]. A newer DPM framework is released `voc-release5` using a cascaded detector for early rejection of unpromising hypotheses with 10-20 times faster detections without noticeable loss in precision or recall. This requires training a new model, preferably on own data, and also gives the option of training a separate detector for each perspective to reduce the number of components in each mixture model. Further improvement includes downsampling the image to lower resolution which requires further analysis of its affect on succeeding modules.

Chapter 8

Vehicle Tracking

Multi-object tracking is like object detection a major field within vision-based research. The possibility of locating targets throughout a video opens up a wide range of possible applications within among others robotics, surveillance, and ADAS. In theory, everything can be tracked, and static objects might as well be of interest (e.g. if the system is moving), but recently the focus have been on tracking pedestrians and vehicles with publicly available benchmarks. This allows novel methods to compare against each other using similar challenges and metrics. It is thus difficult for a method to claim superiority to other methods if it is not tested under the same conditions.

Most notable, the KITTI Vision Benchmark Suite [20] evaluates on-road tracking of vehicles in various environments. Selected state-of-the-art tracking performances are listed in Table 8.1 (see Subsection 6.2.2 for a detailed description of the metrics). One major difference from evaluation of detectors is that there is no clear way of ranking the methods. This is due to the fact that several metrics are important and one metric encapsulating the total performance has not yet been defined. Thus, the choice of tracker is highly application specific. One has to be aware when comparing the trackers, as they all depend on detections. Often, detections are computed for each frame, making the simplest task to associate bounding boxes across frames. This is called tracking-by-detection, and is a common choice in tracking applications, as opposed to tracking without detections. This is among others due to the case when a target leaves the scene and re-enters (and thus needs re-initialization), and that tracking methods without continuous support from a detector tend to drift. The use of detectors does however impact the score

of a tracker, and a tracker might score better than others simply because of better detections. This is handled in the MOT2015 benchmark [28, 32] by making detections publicly available, and indicating if a method uses the public detections or not.

Table 8.1: Selected tracking performances on the KITTI object tracking evaluation dataset for cars. Trackers are not ranked, but sorted according to MOTA score. The arrows indicate if a low or high score is best.

Method	Online	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	Frag↓
NOMT	✗	**69.73 %**	79.46 %	**56.25 %**	12.96 %	36	**225**
MDP	✓	69.35 %	**82.10 %**	51.37 %	13.11 %	135	401
NOMT-HM	✓	67.92 %	80.02 %	49.24 %	13.11 %	109	371
SCEA*	✓	67.11 %	79.39 %	52.13 %	**10.98 %**	106	466
TBD	✗	49.52 %	78.35 %	20.27 %	32.16 %	**31**	535

One of the first decisions when choosing a tracking method is whether the tracker needs to run offline or online. Offline (also referred to as batch) trackers have the advantage that all frames are available at run-time, and thus a tracker can use previous, current, and future information at a given frame. In terms of intelligent vehicle applications this is often not possible, since the tracking needs to run online while new images are captured. If a tracker is to run real-time, no future frames are available, and thus an online method is required. Generally, offline methods outperform online methods, but in Table 8.1 it can be seen that for the KITTI benchmark the difference in performance is not significant. This might be due to the focus on online methods for vehicle applications.

Pedestrian and vehicle trackers have a lot in common, and trackers are often evaluated in both domains. Tracking of pedestrians is considered a more difficult challenge because of e.g. larger variations in between-frame appearance, frequent occlusions, and complex movement patterns. The best MOTA score for pedestrian tracking in the KITTI benchmark is 39.34%, which is remarkably lower than the corresponding MOTA score of 69.73% for vehicles.

In the following, the MDP tracker is investigated and modified to a multi-perspective vehicle tracking focus. That is, the tracking in each perspective is optimized to accommodate the multi-perspective aspect of the system. The description is two-fold. The first section describes the theory and implementation of the tracker. In the second section, modifications are presented that enhance the performance specific to the application of this study. Finally, the tracker is evaluated on four different perspectives.

8.1 The MDP tracker

The highest MOTA scoring online vehicle tracker on the KITTI object tracking benchmark is (as of May 2016) the MDP tracker [61] with publicly available code [60]. It is based on Markov decision processes (MDPs) and the principle of learning to track. The tracker applies to various objects, but is implemented for pedestrian tracking. Though, results are available for vehicle tracking, there is no description of changes from pedestrian to vehicle tracking.

The tracker is applied to all available perspectives separately, and does not contain any vehicle association between perspectives, though the tracker in each individual perspective is optimized for multi-perspective tracking. The performance of the tracker is evaluated in multiple perspectives with two different detectors on the data collected in this work.

8.1.1 Tracking Using Markov Decision Processes

This section takes a practical approach to the theory of MDPs and reinforcement learning. Specifically, the topic will be described in terms of the MDP tracker presented by Xiang et al. [61]. A detailed investigation is beyond the scope of this study. The tracker is originally designed for tracking pedestrians, but is described with vehicles as targets in the following. Theoretically, a MDP can have an infinite number of states and actions. In this study finite MDPs are inspected, since a known number of states and actions are defined. The finite MDP is defined as a tuple $(S, A, T(s, a, s'), R(s, a))$, where:

- $S = \{s_1, s_2, \cdots, s_N\}$ is a set of N states.
- $A = \{a_1, a_2, \cdots, a_k\}$ is a set of k actions. $A(s)$ are the specific actions available to state s.
- $T(s, a, s')$ is the state transition model, also defined as $P(s'|s, a)$ that is the probability that action a in state s will lead to state s'.
- $R(s, a)$ is the reward function, also referred to as the reinforcement function that defines the immediate reward received after executing action a in state s.

Furthermore, the MDP has a start state, s_{start}, and possibly one or more terminal states, s_{terminal}. For the task of tracking, [61] defines four states and seven

actions as shown in Figure 8.1. Thus, $S = \{Active, Tracked, Lost, Inactive\}$. And as an example $A(Active) = \{a_1, a_2\}$. When a vehicle is detected it starts in the *Active* state from which it can transition to *Tracked*, a_1, or *Inactive*, a_2, if the detection is a false positive. A vehicle in the *Tracked* state can stay in that state, a_3, or transition to a *Lost* state, a_4, if the vehicle disappears due to e.g. occlusion. Likewise, a vehicle in the *Lost* state can stay in that state, a_5, transition back to *Tracked*, a_6, or transition to *Inactive*, a_7. The start and terminal states are defined as: $s_{\text{start}} = Active$ and $s_{\text{terminal}} = Inactive$. The latter is a consequence of the fact that there are no possible actions in state *Inactive*, $A(Inactive) = \varnothing$, making it the terminal state of the MDP. Note that the actions are deterministic. Thus, the transition model is $P(s'|s, a) = 1 \ \forall a$. This means that whenever an action is executed the MDP enters the state associated with that action. In other applications the actions can be stochastic, and thus might lead to different states depending on the probability $P(s'|s, a)$.

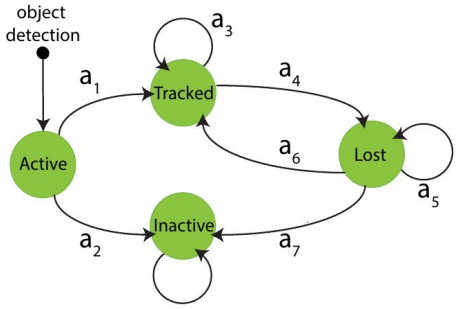


Fig. 8.1: MDP states and actions defined in [61].

Using this MDP configuration, there is no clear way of defining the reward function. Consider for example the reward for executing action a_4 in state *Tracked*, that is $R(Tracked, a_4)$. Intuitively, the MDP should be negatively rewarded when transitioning a vehicle from a *Tracked* state to a *Lost* state, as this is seen as a bad action for the track, especially if the vehicle is not lost in the specific frame. However, if the vehicle is indeed lost (e.g. due to full occlusion) then it is a good thing that the MDP transitions to a *Lost* state, and does not keep the vehicle tracked. The reward for transitioning thus depends on each specific situation and it is difficult to see how one could determine the relative weights of the parameters involved in each of them.

A common way to optimize the MDP is to use the reward function to compute a policy, π, which is a mapping from state space, S, to action space, A. The policy determines which action to execute in each state. By utilizing

the reward function, this policy can be trained using reinforcement learning. With deterministic actions the optimal policy is the one that maximizes the total rewards obtained. Note that the sequence of actions in theory can be infinitely long, for which reason a discount, $\gamma \in [0, 1)$, is often introduced to decay rewards as they age in the sequence. However, since the reward function is not defined in this study it is not possible to use the common method for solving the MDP. Instead, the policy is defined and inverse reinforcement learning (IRL) is used to train the reward function [39]. IRL in MDPs focus on the problem of extracting a reward function given observed optimal behavior. Specifically, the behavior of the MDP with policy, π, is compared to the optimal behavior as observed from ground-truth trajectories of annotated training data[1].

MDP Policy and Reward Function

As previously mentioned, decision making in the MDP follows a policy. Thus for each state the policy defines which action to choose. In the following the policy in each state is described. Furthermore, the reward associated with each action will be learned.

***Active* State Policy:** Two actions are possible in this state, either transition to *Tracked* or *Inactive*. This can be considered as a simple binary classification problem, which needs decision as to whether the detection is a true positive or a false positive. A simple solution is to threshold the detections according to their score, and rely on the detector to output true positives with a higher relative score than false positives. Instead, an offline supervised trained SVM is used with a normalized feature vector, $\boldsymbol{\phi}_A$, consisting of five entries: x and y coordinates of upper left corner, width, height, and score of the detection. The reward function is defined as [61]:

$$R(Active, a) = y(a)(\boldsymbol{w}_A^{\mathrm{T}} \boldsymbol{\phi}_A + b_A) \tag{8.1}$$

Where $a = \{a_1, a_2\}$, $(\boldsymbol{w}_A, b_A)$ defines the hyperplane of the SVM, and:

$$y(a) = \begin{cases} 1 & \text{if } a = a_1 \\ -1 & \text{if } a = a_2 \end{cases} \tag{8.2}$$

The action with the highest reward is chosen, which in this case will always be the result of the SVM classification (as $y(a)$ is used to ensure this).

[1]The training data used are from the proof of concept data captured using six cameras.

It should however be noted that the policy is redundant with the classification performed by the vehicle detector. Thus, removement of false positives by transitioning to *Inactive* may be limited, and one should be aware of the risk of transitioning incorrectly to *Tracked*. In [61] they rely on the MDP to quickly transition to *Inactive* through actions a_4 and a_7, but it does not change the fact that a false positive has been tracked. For this reason, a re-routing of the MDP is introduced in Subsection 8.1.2, which utilizes the sparsity of incorrect detections to deny false trajectories.

***Tracked* State Policy:** The main task of the *Tracked* state is to track vehicles between frames. If this is possible the given vehicle should remain in the *Tracked* state, otherwise it should transition to *Lost*. Thus, two actions are available in this state, a_3 and a_4. By maintaining an online appearance model of the vehicle it is possible to determine if the vehicle is present in a new frame. Specifically, a grayscale template of the target is updated (in a "lazy" manner) every time the vehicle transitions to *Tracked*. The "lazy" update reduces the risk of drifting, which could happen if the template was updated every frame. Also, a history of the appearance is stored, which is simply the latest 10 tracking results.

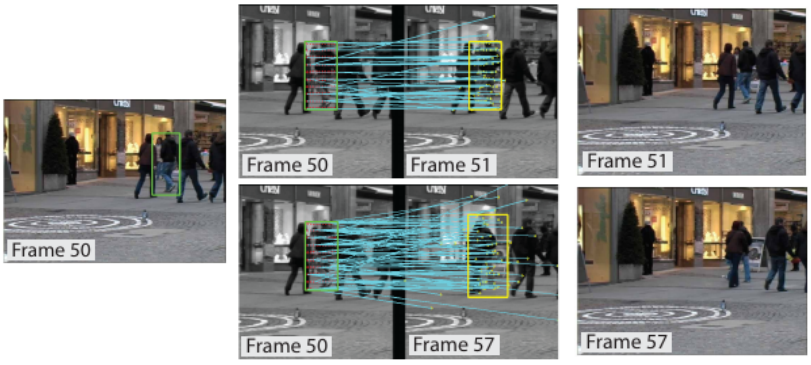


Fig. 8.2: An example of a stable (top) and an unstable (bottom) optical flow [61].

The appearance model is used in a method based on the Median-Flow tracker [25] (part of the widely used Tracking-Learning-Detection method presented in [26]). Firstly, a grid of uniformly distributed points is initialized in the template. The points are used to compute motion flow between the template and a new frame. The flow is calculated using pyramidal iterative Lukas-Kanade tracking [5, 29]. As illustrated in Figure 8.2 the stability of the motion flow is a good indicator of the presence of the vehicle (shown with

pedestrians in the figure). In order to extract the stability of the motion flow it is furthermore computed backwards. That is, the resulting points from the forward flow computations are used to estimate the flow back to the template image. Thus, a forward-backward (FB) error can be obtained as:

$$e_{\text{FB}} = ||\boldsymbol{x} - \hat{\boldsymbol{x}}|| \tag{8.3}$$

Where $\boldsymbol{x}$ is the original position of the point in the template, and $\hat{\boldsymbol{x}}$ is the resulting point in the template from the forward-backward flow estimation. An example is shown in Figure 8.3, where point 2 has a large FB error and

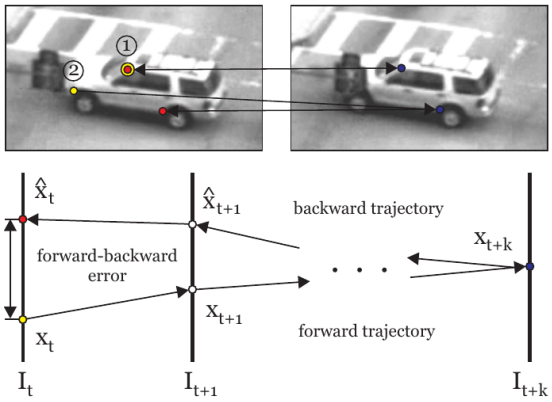


Fig. 8.3: Example of a FB error computation [25].

point 1 has a low error. A reasonable measure of the stability of the motion flow is the median FB error, e_{medFB}. If e_{medFB} is lower than an experimentally defined threshold, θ_{medFB}, then the vehicle is most likely correctly tracked, and all points with a FB error smaller than θ_{medFB} are used to predict the new location of the vehicle. The prediction is limited to a translation and a scaling. The translation is achieved as the median motion flow of the remaining points, and the scaling is the median change in ratio of distances between points before and after the flow computations.

However, as previously described a tracker may be initialized by a false positive detection. If this false positive is part of the background, the motion flow will be stable between frames, and the tracker will thus keep tracking the background as a false positive. The problem can be solved by using the detections available in each frame. False positive detections tend to have an impersistent occurrence pattern, and it is thus reasonable to require a tracker to frequently receive matching detections. To this end, the history

of the tracked vehicle is examined. Specifically, a mean bounding box overlap, o_{mean}, of the latest 10 tracking results and corresponding detections (the detection with the largest overlap with the tracking result in each frame) is computed. If o_{mean} is larger than a threshold, θ_{meanO}, the tracking result is accepted. In the situation with a young track that has less than 10 previous frames, the number of available frames is used. The predicted bounding box is corrected to be the mean of the tracking result and the corresponding detection in the current frame, if the overlap between these is larger than θ_{meanO}, otherwise the tracking result is kept as the bounding box. Finally, the reward function is defined as:

$$R(\textit{Tracked}, a_3) = \begin{cases} 1 & \text{if } e_{\text{medFB}} < \theta_{\text{medFB}} \text{ and } o_{\text{mean}} > \theta_{\text{meanO}} \\ -1 & \text{otherwise} \end{cases} \tag{8.4}$$

$$R(\textit{Tracked}, a_4) = \begin{cases} -1 & \text{if } e_{\text{medFB}} < \theta_{\text{medFB}} \text{ and } o_{\text{mean}} > \theta_{\text{meanO}} \\ 1 & \text{otherwise} \end{cases} \tag{8.5}$$

Which in other words means that if a tracked vehicle has a stable motion flow to a new frame, and it has been detected regularly within the last 10 frames, it is positively rewarded to keep the vehicle tracked. Otherwise, the vehicle is transitioned to a *Lost* state.

***Lost* State Policy:** One of the powers of the MDP tracker is its ability to resume tracking an otherwise lost vehicle. This is possible by transitioning from *Lost* to *Tracked*. In total there are three possible actions for a vehicle in a *Lost* state: stay as *Lost*, a_5, transition back to *Tracked*, a_6, or terminate the tracker, a_7. Given a lost vehicle and a new frame the task is to decide whether the vehicle has reappeared. To achieve this, the tracker requires the detector to provide a new detection of the vehicle. The purpose is thus to correctly associate the lost vehicle to a new detection of the same vehicle. If the vehicle is successfully associated it is transitioned to *Tracked*, otherwise it is kept *Lost*. When the vehicle has been lost for θ_{lost} frames it is terminated by transitioning it to *Inactive*.

Several features are used to determine if the lost vehicle is in a new detection. The feature vector, $\boldsymbol{\phi}$, has a total of 12 elements as shown in Table 8.2. All features (except ϕ_9, ϕ_{11}, and ϕ_{12}) are averaged over the appearance history of the vehicle. As previously described the history consists of the last 10 frames of the vehicle in a *Tracked* state. The first five features are based on the median FB error previously explained in terms of the *Tracked* state policy.

Table 8.2: Feature representation for association of a vehicle with a new detection [61].

Type	Notation	Feature Description
FB error	$\phi_1, \cdots, \phi_5$	Mean of the median forward-backward errors from the entire, left half, right half, upper half, and lower half of the templates in optical flow
NCC	ϕ_6	Mean of the median Normalized Correlation Coefficients (NCC) between image patches around the matched points in optical flow
	ϕ_7	Mean of the NCC between image patches of the detection and the predicted bounding boxes from optical flow
Height ratio	ϕ_8	Mean of the ratios in bounding box height between the detection and the predicted bounding boxes from optical flow
	ϕ_9	Ratio in bounding box height between the target and the detection
Overlap	ϕ_{10}	Mean of the bounding box overlaps between the detection and the predicted bounding boxes from optical flow
Score	ϕ_{11}	Normalized detection score
Distance	ϕ_{12}	Euclidean distance between the centers of the target and the detection after motion prediction of the target with a linear velocity model

The main differences here are that the FB error is also computed from the left, right, upper, and lower halves of the template, and that the region in the new image (in which the motion flow is computed) is limited around the bounding box of the detection. Also note that the median FB error is averaged over the 10 templates. The next two features (ϕ_6 and ϕ_7) describe similarity of image patches using normalized correlation coefficients (NCC). The former uses the neighborhood around points of the motion flow computations. The latter uses the full bounding boxes of the predicted and detected bounding boxes. Next, ϕ_8 and ϕ_9 encapsulate the change in height of the bounding box, ϕ_{10} is similar to o_{mean} as previously described for the *Tracked* state policy, and ϕ_{11} is the normalized detection score. Lastly, ϕ_{11} is a distance between the centroid of the detection and the predicted centroid computed from the average change of centroid placement according to the history. The predicted centroid is computed using (8.10) and (8.11).

The feature vector is used in a SVM to classify a detection as a match or a mismatch. The SVM is trained offline using optimal behavior observed from ground truth annotations. The weights are initialized randomly, and the MDP tracker follows the policy described above to track vehicles in a training sequence with detections from the detector. The weights are updated every time the tracker makes a mistake in the *Lost* state according to manually annotated ground truth bounding boxes with vehicle ID. Two errors can happen: The tracker incorrectly associates a lost vehicle and a new detection ($\cancel{a_6} \rightarrow a_5$), or the tracker fails to associate a lost vehicle with a new detection of the vehicle ($\cancel{a_5} \rightarrow a_6$). The former is added as a negative sample to the training set, while the latter is added as a positive sample. The procedure is iterated until the tracker makes no mistakes in the *Lost* state, or the sequence has been iterated more than θ_{ite} times.

The trained weights of the SVM are static during run-time of the tracker, since no feedback is available. The reward function is then defined as:

$$R(Lost, a) = y(a) \left(\max_{k=1}^{M} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(k) + b \right) \right) \tag{8.6}$$

Where $a = \{a_5, a_6\}$, $\boldsymbol{\phi}(k)$ is the kth feature vector of M detections in the current frame, $(\boldsymbol{w}, b)$ defines the hyperplane of the SVM, and:

$$y(a) = \begin{cases} 1 & \text{if } a = a_6 \\ -1 & \text{if } a = a_5 \end{cases} \tag{8.7}$$

Finally, the reward for transitioning to *Inactive* is defined as:

$$R(Lost, a_7) = \begin{cases} \infty & \text{if the vehicle has been lost for } \theta_{\text{lost}} \text{ frames} \\ -\infty & \text{otherwise} \end{cases} \tag{8.8}$$

Which concludes the policy and reward definition of the MDP. To summarize, the status of a vehicle is described by a MDP. Multiple vehicles require multiple MDPs. For each new frame the MDP of a vehicle follows the policy of the current state, and executes the action with the highest reward.

8.1.2 Tracking Vehicles in a Multi-Perspective Application

The MDP tracker is originally designed for tracking pedestrians, for which reason, it is optimized for tracking vehicles in this study. The first change is the aspect ratio of the template used for associating vehicles between frames, which is chosen based on typical vehicle aspect ratios in the annotations of the KITTI dataset [20] and of one annotated sequence collected in this work as shown in Fig. 8.4.

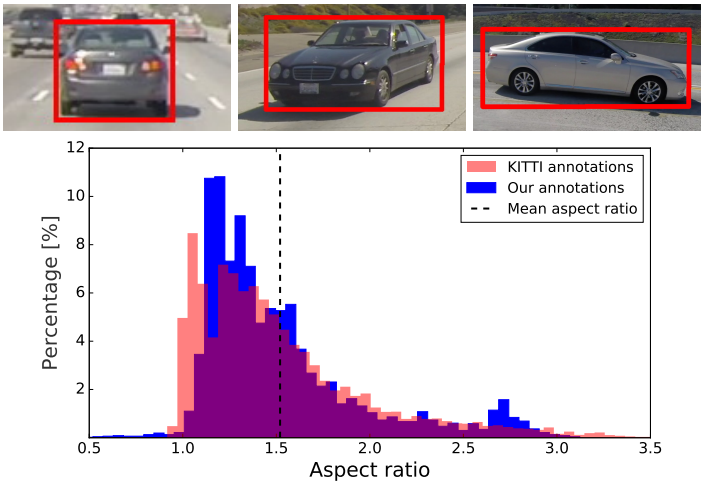


Fig. 8.4: Histogram of annotated vehicle aspect ratios in the KITTI dataset [20] shown with the red color and histogram of annotated vehicle aspect ratios in our data shown with the blue color. The means are approximately equal and shown with the vertical line at approximately 1.5. Above are some examples of aspect ratios increasing from left to right.

Note that the aspect ratio of vehicles varies with the orientation at which they are observed. From this follows that vehicles observed in the side views

will have a larger aspect ratio than vehicles observed in the rear or front view in a highway scenario, where the driving direction is mainly straight forward. Thus, optimally the aspect ratio should be optimized for each of the four perspectives. An aspect ratio of 1.5 is, however, experimentally found to be sufficient for all four views, which is the mean of the annotated bounding boxes aspect ratios.

The variation in appearance is less prominent for vehicles compared to pedestrians, and typical motion does not see abrupt changes as with pedestrians. This allows us to further constrain the creation of new trajectories as seen in Fig. 8.5. Thus, when a new vehicle is detected, it starts in an *Active* state,

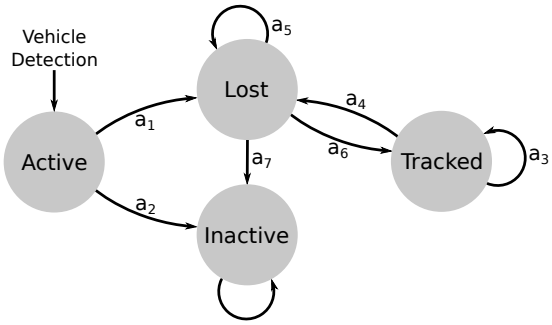


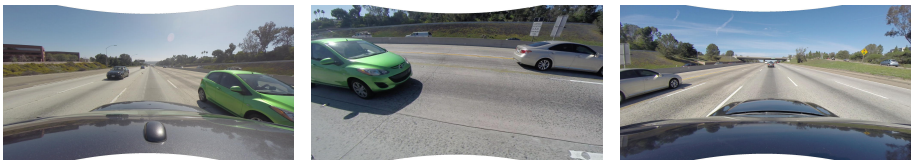
Fig. 8.5: The MDP states and actions designed for tracking vehicles. A new vehicle is not allowed to transition directly from the *Active* state to the *Tracked* state as in the original implementation [61] shown in Figure 8.1.

from where it can transition to a *Lost* state via action a_1 if it is determined to be a correct detection, or to an *Inactive* state via action a_2 if it determined to be a false detection. This stands in contrast to the original implementation where a new vehicle is able to transition directly to a *Tracked* state. The purpose of this re-routing in the MDP is to reduce the number of false positive trajectories caused by spurious detections that incorrectly transition from the *Active* state to the *Tracked* state instead of the *Inactive* state. The trade-off is that correctly tracked vehicles are slower at reaching the *Tracked* state, since they are kept in the *Lost* state until they have been observed θ_{minseen} times. The action a_5 is thus not only the result of no association between a lost track and new detections, but also of a track that is too young to transition to the *Tracked* state. The reward for transitioning from *Lost* to *Tracked* is changed to:

$$R(Lost, a_6) = \begin{cases} -\infty & \text{if tracker age } < \theta_{\text{minseen}} \\ \max\limits_{k=1}^{M} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(k) + b \right) & \text{otherwise} \end{cases} \tag{8.9}$$

Note that in the situation where a vehicle enters *Lost* from *Active* the vehicle has not been tracked yet. Thus, only one template is available for computing $\boldsymbol{\phi}$, which might reduce the robustness of associating the vehicle to a new detection.

In this study the tracking in each perspective has to take into account the purpose of linking trajectories between slightly overlapping perspectives. Thus, trajectories need to prolong as close to the image borders as possible. A situation that meets several challenges such as missing detections caused by truncation and severe appearance variations as the observation angle changes rapidly in the proximity of the ego-vehicle. An example of these challenges is shown in Fig 8.6. Note from the figure that the truncated vehicles are located



(a) Rear **(b)** Left **(c)** Front

Fig. 8.6: Example of challenges of detection and tracking at the image borders.

in regions where they might not be detected in any of the views due to truncation. One solution, that does not require a change of setup, is to design the vehicle detector for detecting truncated vehicles. A topic that has previously received attention, but remains an unsolved problem [47]. In this study the focus is instead on extending the trajectories despite the missing detections. This is achieved by predicting bounding boxes for lost trackers. Let α be the set of templates in the history of a vehicle that are sorted with increasing age. Thus, the newest template is at the first entry, α_1, and the oldest template is at the last, α_L, where L is the number of available templates (maximum 10). An average change of the bounding box parameters is computed using:

$$\dot{\omega} = \frac{1}{L}\sum_{m=1}^{L}\frac{\omega|_{\alpha_m} - \omega|_{\alpha_{m+1}}}{t|_{\alpha_m} - t|_{\alpha_{m+1}}} \tag{8.10}$$

Where ω defines all four parameters of the bounding box, x, y, w, h, and t is the time index of a template. The bounding box prediction is then:

$$\hat{\omega}_{t|_{\alpha_1}+\tau} = \omega|_{\alpha_1} + \dot{\omega}\tau \tag{8.11}$$

Where τ is the time from the last available template to the current time to

which the bounding box parameters are predicted. The bounding box is used as a guess of where the vehicle has moved, which is tested using the pyramidal iterative Lucas-Kanade method to obtain the motion flow from templates of the vehicle. The FB error of the flow is used as a measure of the stability of the prediction. If the median of the FB errors is below a threshold, θ_{medFB} (also used in the *Tracked* state policy), the prediction is accepted as a valid vehicle association. If that is not the case, the motion flow for the left and right half of the bounding boxes are investigated, since parts of the vehicle might have left the image. If the median FB error shows to be below a more strict threshold, θ_{medFBLR}, for either the left or the right half, the prediction is assumed to be a true association. The procedure is shown in Algorithm 2.

Algorithm 2 Algorithm for extension of trajectories.

```
Predict bounding box using (8.10) and (8.11) for x, y, w and h
e_FB ← compute median of FB error
if e_FB < θ_medFB then
    Prediction is accepted
else
    e_FBL ← median of FB error for left half
    e_FBR ← median of FB error for right half
    if e_FBL < θ_medFBLR or e_FBR < θ_medFBLR then
        Prediction is accepted
    end if
end if
```

The extension of the track is performed until τ gets larger than a predefined value or more than half of the bounding box is outside the image. Drifting of the tracker is limited by not updating the template even if the prediction is accepted. The extension of tracks changes the reward for transitioning from *Lost* to *Tracked* to its final form:

$$R(Lost, a_6) = \begin{cases} -\infty & \text{if tracker age } < \theta_{\text{minseen}} \\ \infty & \text{else if } e_{\text{FB}} < \theta_{\text{medFB}}, e_{\text{FBL}} \text{ or } e_{\text{FBR}} < \theta_{\text{medFBLR}} \\ \beta & \text{otherwise} \end{cases} \tag{8.12}$$

Where $\beta = \max_{k=1}^{M} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(k) + b \right)$ is the maximum SVM score for the new detections. Note that β is negative if no new detections match the vehicle.

8.2 Evaluation

In this section the vehicle tracking is evaluated using the detections evaluated in Section 7.2. Evaluation of multi-object tracking methods has received a lot of attention in research, and multiple widely used benchmarks have emerged. These mainly focus on pedestrians [16, 32] and vehicles [20, 58]. As with vehicle detection, the KITTI Vision Benchmark Suite [20] is currently the go-to dataset for evaluation of vision-based on-road vehicle tracking. This dataset delivers a large number of annotations from multiple modalities. The bounding boxes are, however, annotated from 3D lidar points and projected to the image plane causing some of the ground truth bounding boxes to be slightly inaccurate. Furthermore, the cameras are front facing, and thus only situations occurring in front of the vehicle are examined. In the following evaluation, the tracker is tested in four perspectives (the setup described in Chapter 5), and the results are listed in Table 8.3. A minimum bounding box height of 35 pixels and a bounding box overlap of 0.7 are used. Vehicles that are more than 50% truncated are ignored.

Table 8.3: Single perspective tracking results reported for two detectors and the modified MDP tracker, Multi-Perspective MDP (MPMDP). The arrows indicate if a high or low score is best.

Track/Persp.	MOTA↑	MOTP↑	IDS↓	Frag↓	MT↑	ML↓	Recall↑	Precision↑
Front								
SubCat-MPMDP	0.82	0.83	1	1	0.80	0.00	0.83	1.00
DPM-MPMDP	0.71	0.78	0	0	0.80	0.10	0.81	0.89
Rear								
SubCat-MPMDP	0.82	0.85	0	9	0.75	0.00	0.87	0.94
DPM-MPMDP	0.87	0.80	1	4	0.75	0.00	0.87	1.00
Left								
SubCat-MPMDP	0.76	0.77	0	1	0.40	0.20	0.76	1.00
DPM-MPMDP	0.77	0.80	0	1	0.40	0.40	0.77	1.00
Right								
SubCat-MPMDP	0.55	0.83	0	0	0.33	0.33	0.55	1.00
DPM-MPMDP	0.62	0.82	0	0	0.67	0.33	0.62	1.00
Total								
SubCat-MPMDP	0.81	0.84	1	11	0.65	0.08	0.83	0.97
DPM-MPMDP	0.79	0.79	1	5	0.69	0.15	0.83	0.95

All recent benchmarks rely on the CLEAR MOT metrics [2], and it has become common to report tracking results in multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP). Furthermore, common metrics include ID switches (IDS), fragmentations (Frag), mostly tracked (MT), mostly lost (ML), precision, and recall. The details of the metrics are

explained in Subsection 6.2.2.

As seen from Table 8.3 the tracker generally scores better with the SubCat detector than with the DPM detector. However, the tracker using SubCat detections does seem to perform a little worse in the side perspectives. Notice the high scores in precision, which indicate the low number of false positive trajectories. The relatively low MOTA and recall scores in the side perspectives show that the system has difficulties tracking vehicles on the side of the ego-vehicle. There are multiple possible causes: From Figure 8.4 it is noticeable that the data collected from the side perspectives creates a peak around an aspect ratio of 2.75, which is not that well represented in the KITTI dataset. Thus, the detector is not tuned for these cases as also visible in the evaluation in Figure 7.12. Furthermore, the vehicles suffer more from truncation in the side perspectives (see Figure 6.4). Also, the vehicles tend to stay for a shorter duration on the side of the ego-vehicle. This results in a larger impact of the time in which a vehicle is kept *Lost*, θ_{minseen}, before transitioning to *Tracked*. It should however be noted that the results are generally impressive compared to the state-of-the-art performance achieved on the KITTI dataset (see Table 8.1). One should be cautious when comparing the results to the KITTI benchmark, which comprises more data in other types of environments than highway driving (e.g. urban driving).

For future work it would be interesting to test the MPMDP tracker on the KITTI dataset to see how it compares to other methods. However, since the focus of MPMDP is on multi-perspective tracking the test would not fully show the capabilities of the tracker, as the benchmark is not designed to do so. Instead, two trackers evaluated on the KITTI dataset are tested on the data collected in this work. Namely the TBD tracker implemented by Geiger et al. [18], and the original MDP tracker proposed by Xiang et al. [61]. The results for all four perspectives are listed in Table 8.4. Notice, the surprisingly

Table 8.4: Single perspective tracking results reported for three trackers. The arrows indicate if a high or low score is best. O shows if a tracker is online.

Tracker	O	MOTA↑	MOTP↑	IDS↓	Frag↓	MT↑	ML↓	Recall↑	Precision↑
DPM-TBD	✗	**0.83**	0.78	5	**5**	0.73	**0.04**	0.84	**0.98**
DPM-MPMDP	✓	0.79	**0.79**	**1**	**5**	0.69	0.15	0.83	0.95
DPM-MDP	✓	0.76	**0.79**	8	11	**0.77**	**0.04**	**0.85**	0.91

high MOTA score of the TBD tracker, which on the KITTI benchmark scored much lower than the MDP tracker. As previously mentioned the lower score

on the KITTI benchmark might be caused by the detector. In this test, the three trackers use the same detections (Note that MOTP is almost similar for the three trackers), and the offline TBD tracker scores best. This result is however not so important to this study, and due to the offline processing part of the TBD tracker it is not considered further. The MPMDP tracker slightly outperforms the MDP tracker in terms of MOTA score. There are some relevant differences between the two that should be noted. The MPMDP introduces a trade-off between higher precision score and lower recall score. This is because of the MDP re-routing that is seen to effectively improve the precision by removing false positive trajectories, but also introduces a slight decrease in recall caused by the missing frames in the start of the trajectories. Also note that the extension of the trajectories in the MPMDP influences these scores, as it brings down the number of false negatives and possibly introduces more false positives. It is difficult from these scores to determine exactly how each modification impacts the result. The MPMDP performs worse according to the MT and ML scores. This is caused by the side perspectives, where the combination of worse detections, tougher constraint on acceptance of trajectories and shorter trajectories results in less mostly tracked and more mostly lost vehicles.

The evaluation above proves the applicability of single-perspective trackers to different perspectives covering the full surroundings of the ego-vehicle. In future studies more attention is needed to the side perspectives, where the trackers perform worse than in the front and rear perspectives.

Chapter 9

Tracking and Association Between Perspectives

This chapter describes the last two modules of the multi-perspective trajectory estimation, the transformation to road and the multi-perspective tracking. Unlike the proof of concept system described in Part II that associates the vehicles between perspectives in the image planes, this system maps the surrounding vehicles to a top-down view of the scene, and tracks them in the road plane. A surround top-down view has previously been used in e.g. parking assistance applications, but the tracking of vehicles in the road plane all the way around the ego-vehicle has not been investigated before. This chapter describes the modules designed to achieve the surrounding trajectories, and discusses advantages, disadvantages, and alternative methods for solving the problem. As the first vision-based system, full surround trajectories are obtained, and a possible use of the trajectories is shown in a 3D visualization tool.

9.1 Transformation to Road Plane

The task of estimating world positions of surrounding vehicles is typically solved with a sensor setup that is able to obtain depth information. Common choices include LiDAR, radar, and stereo vision. However, with monocular vision no direct access to depth information is available, as recovery of structure is inherently ambiguous. One solution is to use the width of the vehicles

as shown in Figure 9.1. The variance in width is small and it is thus a simple

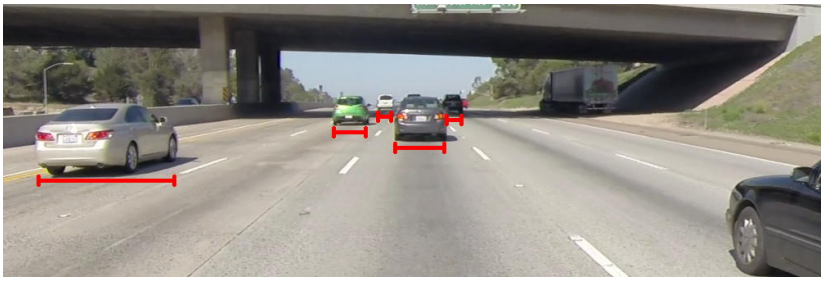


Fig. 9.1: Example of using the width of vehicles to estimate their distance from the ego-vehicle. The method is not robust against variations in orientation as can be seen from the leftmost vehicle.

measure for estimating how far away a vehicle is. However, this method is vulnerable to variations in orientation and therefore requires a front or rear view of the vehicle. This makes it a method that is not suitable for a full surround view application, and even in a monocular forward looking application the method might face problems with the angle at which vehicles are observed.

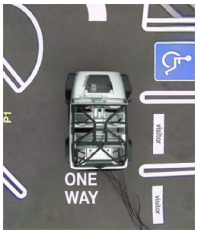
Instead inverse perspective mapping (IPM) is introduced. The method is widely used in intelligent vehicle applications for e.g. lane estimation [3], parking assistance [64], and blind spot visualization for trucks [12] (see Figure 9.2 for examples). The purpose of IPM is to achieve a top-down (also referred to as bird's eye) view of the road. It does so by estimating a projective transformation between the image plane and the road plane. It thus assumes that the road is flat, which is an assumption that requires awareness, since road irregularities may introduce large errors.

In this study we are not interested in a full IPM of the surrounding road, but rather just the location of surrounding vehicles. In the following sections it will be described how to estimate a homography matrix, how multiple homography matrices can be used in a global manner, and how a vehicle is located.

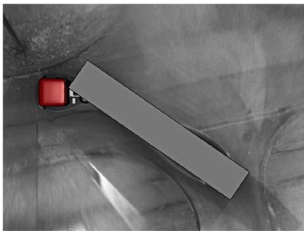
9.1.1 Inverse Perspective Mapping

In this section a front looking camera will be used to explain the theory of IPM. Points in the road plane undergo a perspective transform when viewed from a camera mounted on a vehicle as shown in Figure 9.3a. In computer

(a) Lane estimation [3].



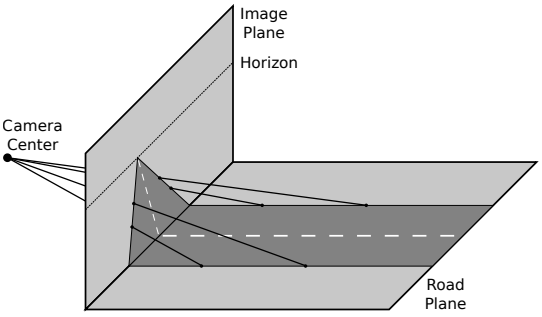
(b) Parking assistance [64].



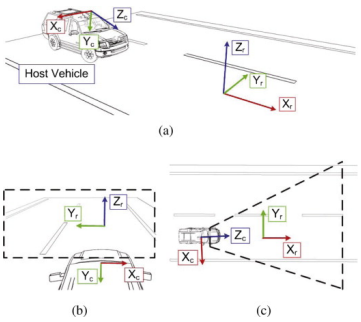
(c) Blind spot visualiztion [12].

Fig. 9.2: Examples of inverse perspective mapping applications.

vision, this planar homography is defined as an invertible projective transformation from one plane to another that maps lines as lines. The two coordinate frames are shown in Figure 9.3b. Note that the origins of the frames are likely to be chosen otherwise, and that $Z_r = 0$ is used.



(a) Capturing a road with a camera.



(b) Frames of image and road [42].

Fig. 9.3: The road undergoes a perspective transform when captured with a camera as shown in the left image. The coordinate frames of the camera and the road are shown in the right image.

The process of capturing an image of the road can be described as a central projection. Assuming only points in the road surface are used means that the central projection is a mapping between two planes. It can then be computed

using:

$$\boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x} \tag{9.1}$$

where $\boldsymbol{x}'$ is the homogeneous image point, $\boldsymbol{x}$ is the homogeneous point in the ground plane, and $\boldsymbol{H}$ is the 3x3 homography matrix. Since $\boldsymbol{H}$ is invertible it follows that:

$$\boldsymbol{x} = H^{-1}\boldsymbol{x}' \tag{9.2}$$

Which shows that points in the road can be reconstructed from the corresponding image point and the inverse homography matrix. In practice, however, it is common to estimate the homography matrix the other way around, such that:

$$\boldsymbol{x} = \boldsymbol{H}\boldsymbol{x}' \tag{9.3}$$

With:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{x}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{9.4}$$

In inhomogeneous coordinates ($z = 1$ so $x = x/z = x$) this gives two equations for a corresponding point pair:

$$x = \frac{h_{11}x' + h_{12}y' + h_{13}}{h_{31}x' + h_{32}y' + h_{33}}, \quad y = \frac{h_{21}x' + h_{22}y' + h_{23}}{h_{31}x' + h_{32}y' + h_{33}} \tag{9.5}$$

The equations are rearranged:

$$\begin{aligned} x\left(h_{31}x' + h_{32}y' + h_{33}\right) - h_{11}x' - h_{12}y' - h_{13} = 0 \\ y\left(h_{31}x' + h_{32}y' + h_{33}\right) - h_{21}x' - h_{22}y' - h_{23} = 0 \end{aligned} \tag{9.6}$$

Which can be used with matrix notation, $\boldsymbol{A}\boldsymbol{h} = \boldsymbol{0}$:

$$\begin{bmatrix} -x'_1 & -y'_1 & 1 & 0 & 0 & 0 & x_1x'_1 & x_1y'_1 & 1 \\ 0 & 0 & 0 & -x'_1 & -y'_1 & 1 & y_1x'_1 & y_1y'_1 & 1 \\ & & & & \vdots & & & & \\ -x'_N & -y'_N & 1 & 0 & 0 & 0 & x_Nx'_N & x_Ny'_N & 1 \\ 0 & 0 & 0 & -x'_N & -y'_N & 1 & y_Nx'_N & y_Ny'_N & 1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \boldsymbol{0} \tag{9.7}$$

Multiple things need to be noted in (9.7). Firstly, multiple corresponding point pairs (N) are included. Each of the point pairs results in two rows in

$\boldsymbol{A}$. Thus, the first two rows are associated with the first corresponding point pair ($\boldsymbol{x}_1 = \begin{bmatrix} x_1 & y_1 \end{bmatrix}^\mathrm{T}$ and $\boldsymbol{x}'_1 = \begin{bmatrix} x'_1 & y'_1 \end{bmatrix}^\mathrm{T}$). Furthermore, since $\boldsymbol{H}$ is known to have eight degrees of freedom [23], a minimum of four point correspondences ($N_{min} = 4$) are needed to solve the set of equations.

The homography matrix is achieved by solving for $\boldsymbol{h}$ in $\boldsymbol{Ah} = \boldsymbol{0}$. Thus, any vector $\boldsymbol{h} \neq \boldsymbol{0}$ in the null space of $\boldsymbol{A}$ is a solution. This can be solved using singular value decomposition (SVD). If N is chosen to be N_{min} then the solution is exactly determined as the one dimensional null space of $\boldsymbol{A}$, which is the column of the right singular vectors with a corresponding singular value equal to zero. If the system is overdetermined with more than N_{min} point correspondences then the column of the right singular vectors with the lowest corresponding singular value is chosen as to minimize $||\boldsymbol{Ah}||$. Often the singular values are ordered in a decreasing order, in which case the rightmost column of the right singular vectors is chosen.

OpenCV includes two functions that can be used to estimate the homography matrix, `findHomography()` and `getPerspectiveTransform()`. The latter is the simplest, and uses four point correspondences to find the exactly determined homography matrix. The former is more flexible and can use more corresponding point pairs in order to reduce errors caused by noise. It does so by using random sample consensus (RANSAC) or least median of squares (LMedS) in order to randomly estimate various homography matrices, while choosing the homography matrix with the most support.

As previously mentioned the purpose of IPM is to use the estimated homography matrix to visualize the road in a top-down view. This procedure requires a mapping of pixels from the image plane to the road plane using (9.3). OpenCV includes the function `warpPerspective()` that applies the homography to an image. In fact, it actually does the mapping in a reverse order, from destination to input, in order to avoid sampling artifacts. Thus, for each pixel in the destination image it computes the corresponding input pixel using the inverse homography.

Figure 9.5 shows an example of how IPM is used. The example uses a front looking camera and the fact that lane markings are made in compliance with the *California Manual on Uniform Traffic Control Devices for Streets and Highways* [7] and *A Policy on Design Standards Interstate System* [1]. The distance from a lane marking starts to the next lane marking starts is 14.6m (48ft) as shown in Figure 9.4 and the minimum lane width is 3.2m. This means that it is possible from a single image to estimate a homography from

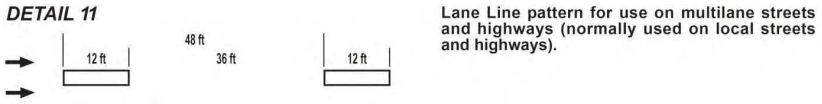


Fig. 9.4: Official guideline for lane marking on a highway [7].

image plane to road plane by choosing four or more corresponding point pairs as shown in Figure 9.5. Note that the lines connecting the points are

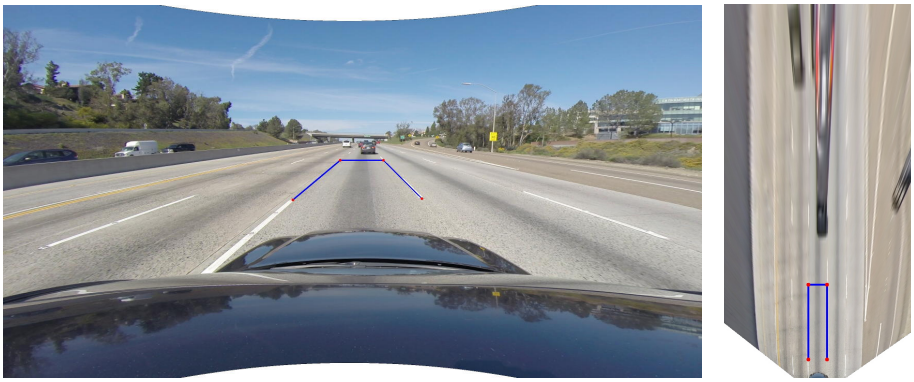


Fig. 9.5: Example of an IPM performed in a front looking camera on the vehicle.

only used as a means to show how the points relate in the image plane and the road plane, respectively. Also, it is important to note that points which are not placed in the road plane are mapped erroneously. As an example, the vehicle in front seems to be unrealistically long in the top-down view, which in terms of IPM makes logical sense, since the points are assumed to be in the road plane, and the vehicle in front vertically takes up space all the way to the horizon. This gives food for thought; what happens if the road plane changes relatively to the image plane due to a hill, road bump, or similar, in which case the imposed road plane is incorrect? Figure 9.6 shows three examples of typical erroneous IPMs. The first image shows a correct IPM, where the road is flat. The second and the third images show a tendency of the road becoming wider further away from the vehicle, and the last image shows the opposite. The second and fourth would be the typical results of the vehicle approaching an uphill or downhill slope, respectively. Thus, caution is needed when IPM is used to impose a ground plane using a camera mounted on a moving vehicle.

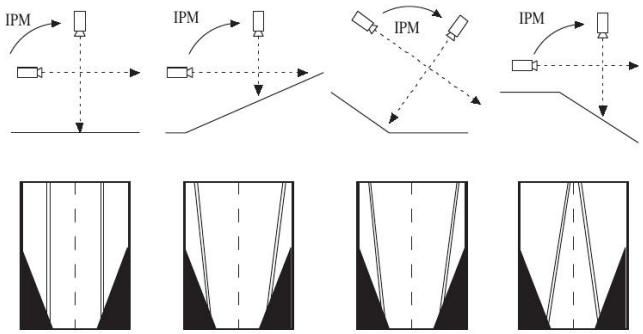


Fig. 9.6: The leftmost image shows a correct IPM, while the other three are examples of cases in which the IPM are erroneous [40]. The cases are: the ego-vehicle is approaching an uphill, the ego-vehicle is on a downhill and approaching a flat road, and the ego-vehicle is approaching a downhill

9.1.2 Calibration of Multiple Local Homographies

In the previous section it was described how to use IPM to estimate a homography for one camera and apply it to an image in order to achieve a top-down view. However, in this study multiple cameras with different orientations are used simultaneously. Using the method naively would create multiple local homographies with no relation to each other. Instead a global solution is wanted, in which the homographies map to the same coordinate frame. As shown in Figure 9.2 previous studies have used this to visualize obstacles in the near proximity of the ego-vehicle.

Two methods have been considered. The first is an extrinsic calibration of the cameras. Thus, a task of finding the rotation and translation between each pair of cameras. This can be achieved in several ways: by manual measuring, which will be prone to measurement inaccuracies; by making a rig, which will have to be created specifically for this study, and still subject to errors in measurements; by using the overlapping regions of the camera views in order to map corresponding point pairs [64], but often it is wanted to limit the overlapping region as to use as few cameras as possible; and finally by using e.g. a mirror [27] or a laser pointer [66], which in itself sounds error-prone, but is shown to be practically possible. All of these methods are time-consuming and requires a lot of manual work, which makes the system more difficult to re-produce. Therefore, instead of calibrating the cameras extrinsically, the used method takes advantage of the fact that the local homographies map to the same ground plane and it is thus only a matter of linking the local ho-

mographies in that plane. This is easily solved by knowing the relationship between points in the ground plane used for estimating the homographies. Such an approach could be applied to images captured for all cameras in a similar scene as the one shown in Figure 9.5 using the road marking standards described previously. This would allow for an on-road calibration process as shown in the left image of Figure 9.8a with four cameras. However, there is no simple method to validate the consistency of the markings, and the calibration could possibly end up using wrong distances. Instead, a static and easily measurable scene is optimal. An ideal choice is thus a parking lot as shown in Figure 9.7, where parking booths serve as consistent and easily distinguishable patterns for calibration of a multi-camera setup.

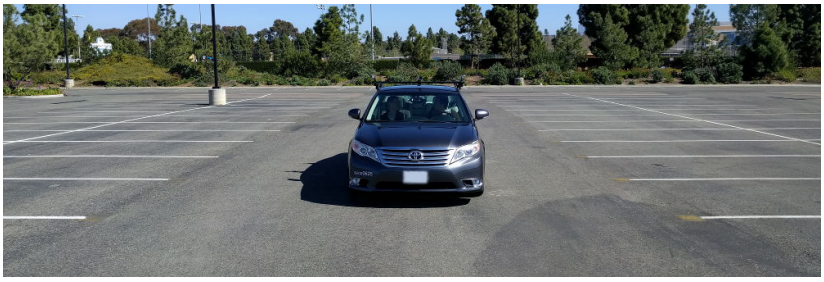
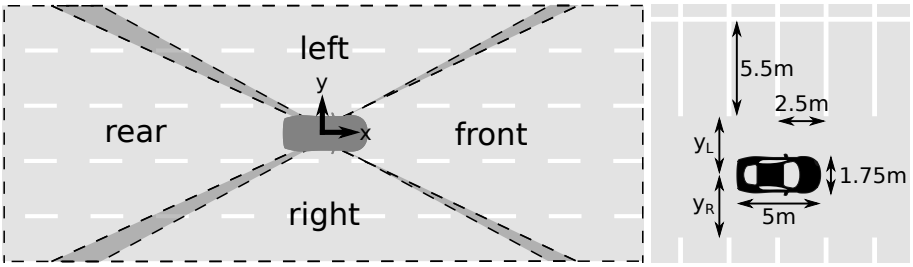


Fig. 9.7: A parking lot is a good place to calibrate the cameras to the ground plane.

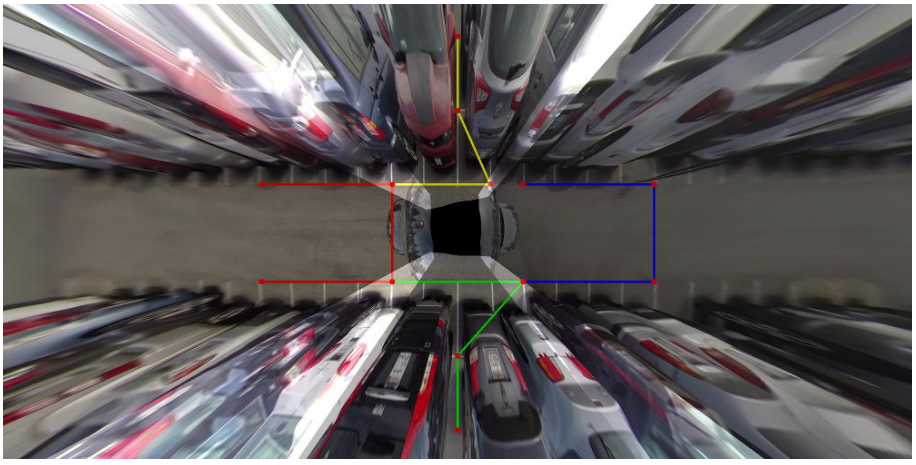
In order to achieve a mapping as the one shown in the left image of Figure 9.8a with the origin in the middle of the ego-vehicle, various distances of the parking booths and the ego-vehicle have to be measured. The rightmost image of Figure 9.8a shows the static measures: booth length and width, and vehicle length and width. Thus, every time the camera setup needs to be calibrated only two measurements (y_L and y_R) are needed, being the distance from ego-vehicle to each parking booth row. Note that more measurements might be needed to ensure that the ego-vehicle is facing a direction parallel to the booths and that it is centered in the x-direction at a booth separating line.

The method is not noiseless. In fact a measurement error of 5cm of the booth width would result in an error of 50cm just 25m in front of the ego-vehicle. Furthermore, noise is introduced when capturing the scene with a camera and selecting the points in the images, which might make the error worse. This is of course a worst-case example, since multiple measurements of randomly selected parking booths should ensure that the measurement



(a) Left: Areas covered by the cameras in the ground plane. Right: Measurements for calibration.

(b) The four perspectives each with four selected points. View by line color: Front, right, rear, left.



(c) Top-down view showing the surroundings of the ego-vehicle with the corresponding points.

Fig. 9.8: Process of calibrating four cameras to road plane using the markings in a parking lot.

error is reduced as much as possible. Nevertheless, IPM probably should not be the only choice for locating vehicles in applications where accuracy is of high importance, but could possibly be used in a multi-modal system. The accuracy is sufficient for this study, which is not a life-or-death ADAS.

Figure 9.8 shows the full process of achieving a full surround top-down view using four cameras. In Figure 9.8b the four views are shown with points selected to estimate the homography matrices. Note from the previously described theory that more than four points can be used to achieve a result, which is more robust to noise. Figure 9.8c shows the resulting IPMs, where the bright regions are overlapping regions of the cameras. The colored lines are used to show how the different points map to the top-down view, and are not part of the actual IPM calculations.

The resulting top-down view of a highway scene is shown in Figure 9.9. One vehicle is visible in the right view and partly in the rear view. Note how it is mapped in the top-down view in the overlapping region. This is a promising result that motivates the choice of associating vehicles between perspectives in the ground plane.

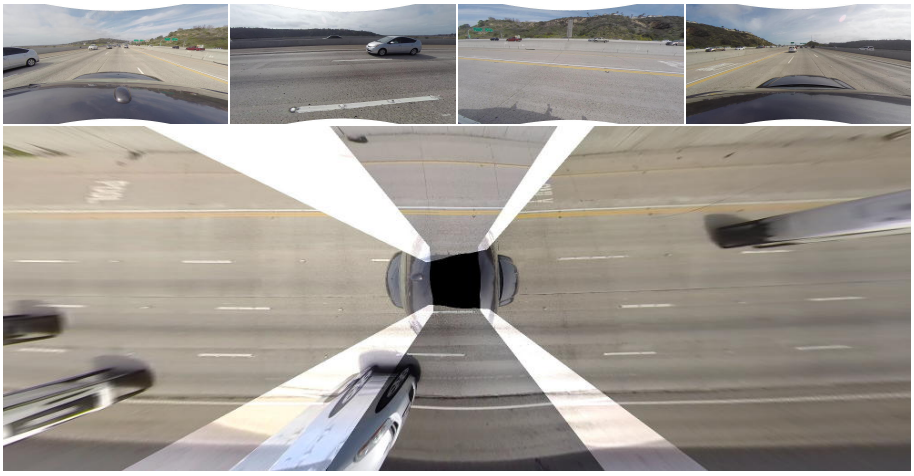


Fig. 9.9: Example of four images captured on a highway and the resulting full surround IPMs.

Figure 9.10 shows an example where the flat road assumption is violated. The resulting top-down view is an example of the situations shown in Figure 9.6, where the road in front of the vehicle is clearly not correctly mapped. This particular example is caused by a dip in the road. The rear end of the ego-vehicle is located in the dip, while the front has exited the dip. This

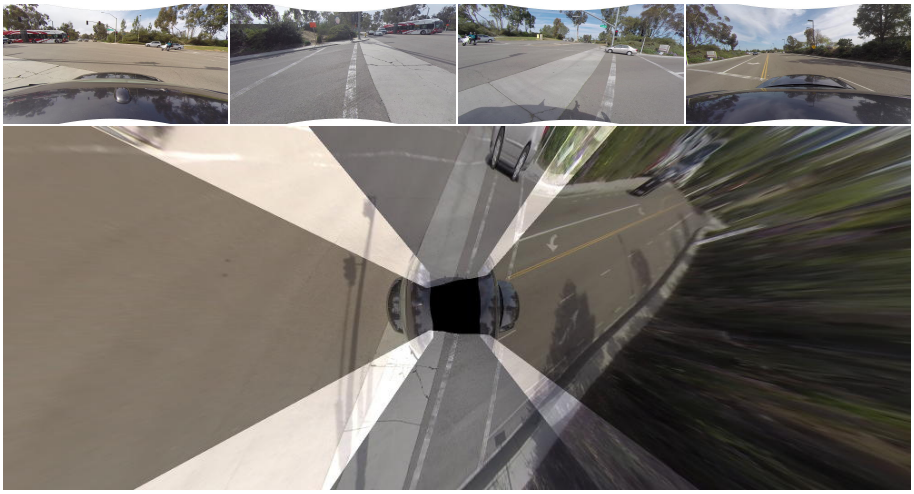


Fig. 9.10: Example of a flat road violation that causes large errors in the IPM of the front view.

causes the vehicle to have a slight angle towards the sky. Furthermore, the road in front of the vehicle has a slight down slope. Note the relatively small difference between the front looking views of the correctly mapped example (Figure 9.9) and this example (Figure 9.10). The change in angle is however enough to change the relative positions between camera and ground plane so much that it causes large errors in the IPM. This proves that caution is needed when IPM is used to compute distances. Since the data used in this work is specifically captured on flat sections of highways the errors are assumed to be limited to an acceptable level. Other than road irregularities, small movements of the cameras might happen during a drive, since the cameras are mounted with clamps (see Figure 5.1). Cameras drifting cause lasting errors until the cameras are re-calibrated to the ground plane, and is thus a severe source of inaccuracies.

9.1.3 Mapping a Vehicle

In the previous sections IPM has been used to establish a top-down surround view of the road. As previously mentioned the visualization in itself is not important to this application. This section explains how vehicles are located in the road plane and mapped to a top-down view in order to compute the distance to the ego-vehicle.

Firstly, examine how vehicles are mapped in Figures 9.5, 9.8c, 9.9, and

9.10. It is noteworthy that even though the vehicles are heavily distorted in the top-down view, the places they touch the road seem to be correct. This is a consequence of the nature of IPM, since the tires of the vehicles are touching the ground they are also mapped correctly. Thus, the location of vehicles may be inferred from where they touch the ground. For this purpose it is noted that the bounding boxes of the vehicles (see e.g. Figure 7.11) has a simple measure being the bottom line of the box. Specifically, the middle of the bottom of the bounding box can be used as shown in Figure 9.11. The advantage is that this point is easily computed from the bounding box.

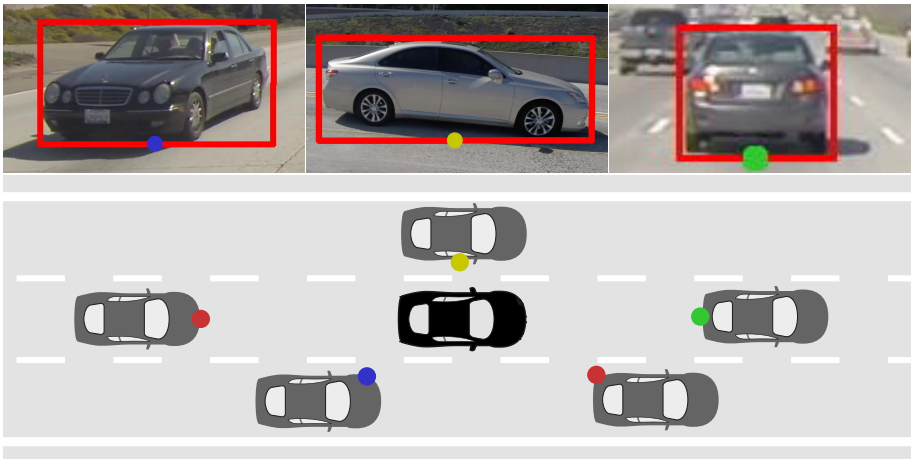


Fig. 9.11: Example of using the middle of the bottom of the bounding boxes for mapping vehicles.

However, at different viewing angles the point will not represent the same point on the vehicles in world space as illustrated in Figure 9.11. Since the vehicles are tracked in video sequences there will be no abrupt changes, and the point will gradually change for each vehicle. Also note that inaccurate bounding boxes can cause errors in the mapping. In particular, vehicles that are far away from the ego-vehicle, where small errors in bounding box can cause large errors in the road plane.

Since the cameras are overlapping, a vehicle might be tracked in two perspectives simultaneously resulting in two points in the road surface. These points can be merged to one point using the spatial property that two points belonging to the same vehicle are placed close to each other in the road surface. This is essentially a form of vehicle association between perspectives, but the purpose is to make sure that each vehicle is only represented by one

point in the road surface.

Though, the highway sections are flat, vibrations might still cause inaccuracies in the IPMs. To reduce the noise, each trajectory from the MPMDP is mapped to the road plane and filtered using an average of the last θ_{pts} points.

9.2 Road Plane Tracking Using Kalman Filtering

In the previous section the surrounding vehicles were mapped to a road plane common to all perspectives of the ego-vehicle. This section handles the association of vehicles that are mapped from multiple perspectives during their movement around the ego-vehicle, and the noise involved with inverse perspective mapping. To do so, an online Kalman filter is used for each trajectory.

9.2.1 The Kalman Filter

The choice of a Kalman filter introduces an important assumption that the dynamics of vehicles are linear, which will be showed to be sufficient to this application. However, be aware that vehicles do follow nonlinear paths, and optimally the filter would have to include this in the model (e.g. by using the extended Kalman filter).

The state of a vehicle in the road plane is modeled as the $\boldsymbol{x}_{\text{pos}} = [x, y]^{\text{T}}$ position and the $\boldsymbol{x}_{\text{vel}} = [\dot{x}, \dot{y}]^{\text{T}}$ velocity. Thus, the state vector, $\boldsymbol{x}_k$, is of the form:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{x}_{\text{pos}}{}^{\text{T}} & \boldsymbol{x}_{\text{vel}}{}^{\text{T}} \end{bmatrix}^{\text{T}} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^{\text{T}} \tag{9.8}$$

This means that a vehicle for each frame is modeled with a 2D position and velocity. Following the prediction-correction scheme shown in Figure 9.12, the model is used to predict where a vehicle has moved between two frames:

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} \tag{9.9}$$

Where $\boldsymbol{A}$ is the model transition matrix, and the control input is left out (since not used in this application). The linear motion of the vehicle can be computed as:

$$\boldsymbol{x}_{\text{pos}}^{k} = \boldsymbol{x}_{\text{pos}}^{k-1} + \boldsymbol{x}_{\text{vel}}^{k-1}, \quad \boldsymbol{x}_{\text{vel}}^{k} = \boldsymbol{x}_{\text{vel}}^{k-1} \tag{9.10}$$

This means that $\boldsymbol{A}$ is:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9.11}$$

Note that Δt in this application is equal to one, since $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$ are one frame apart. The prediction in (9.9) is based on what was known in the last frame, and the result is referred to as the a priori state estimate of the current frame (some times noted as $\boldsymbol{x}_k^-$). The model for performing this estimate is rarely assumed to be perfect, for which reason an uncertainty is added. This is done in form of the process noise covariance matrix, $\boldsymbol{Q}$, which is a measure of how much the model might be corrupted by noise, and thus how trusted the a priori state estimate is. The process noise is used to estimate the a priori error covariance:

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{Q} \tag{9.12}$$

Thus, the Kalman filter has two a priori estimates that describes how the model believes a vehicle has moved from the previous frame, and how certain it is of this estimate, $\boldsymbol{x}_k^-$ and $\boldsymbol{P}_k^-$. These calculations form the predictive part of the Kalman filter. Note that these are based on the model only.

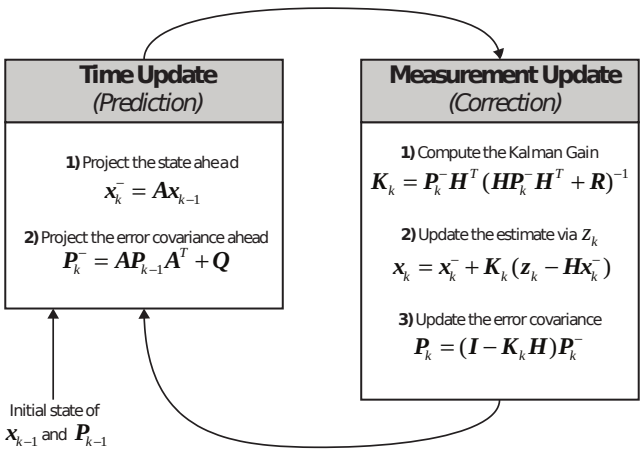


Fig. 9.12: Prediction-correction scheme of the Kalman filter with equations.

The correction part of the filter combines the a priori estimates that are based on what was known from the previous frames with what is known from the current frame. Specifically, the filter receives a new measurement,

$\boldsymbol{z}_k$, that is information about the vehicle from the current frame (note that sometimes the filter might not receive a new measurement, and thus has to rely purely on the a priori estimates). In this application the measurement is a new 2D position of the vehicle from the previously described modules:

$$\boldsymbol{z}_k = \begin{bmatrix} x_{\text{road}} & y_{\text{road}} \end{bmatrix}^{\text{T}} \tag{9.13}$$

A measurement transition matrix, $\boldsymbol{H}$, is used to relate a measurement with the state vector:

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x}, \quad \boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{9.14}$$

This allows the combination of the a priori state estimate and the new measurement to an a posteriori state estimation:

$$\boldsymbol{x}_k = \boldsymbol{x}_k^- + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k^- \right) \tag{9.15}$$

Where $\boldsymbol{K}_k$ is the Kalman gain that weights the a priori state estimate and the new measurement. Thus, the Kalman gain has to take into account the model uncertainty, $\boldsymbol{Q}$, and the noise corrupting measurements that is described by the measurement noise covariant matrix, $\boldsymbol{R}$. Note that a higher Kalman gain results in more weight on the measurement, and a lower Kalman gain gives more weight to the estimate of the model. Specifically, the Kalman gain is computed using:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^{\text{T}} \left(\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^{\text{T}} + \boldsymbol{R} \right)^{-1} \tag{9.16}$$

Thus, larger values of the measurement noise covariance, $\boldsymbol{R}$, will decrease the Kalman gain, and thereby reduce the weight of the measurement. Likewise, larger values of the a priori error covariance, $\boldsymbol{P}_k^-$, will increase the Kalman gain, and thus reduce the weight of the a priori state estimate (remember that the process noise covariance, $\boldsymbol{Q}$, is included in the a priori error covariance). Lastly, the error covariance is corrected:

$$\boldsymbol{P}_k = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H} \right) \boldsymbol{P}_k^- \tag{9.17}$$

This is the a posteriori error covariance, which is used in the calculations of the future a priori error covariance. This concludes the correction step of the Kalman filter, and the a posteriori state estimate is the output for the current frame. The iterative process is shown in Figure 9.12.

9.2.2 Parameters of the Kalman Filter

The theory above introduced the internals of the Kalman filter. In this section the choice of noise covariance matrices is described. Two matrices are used to model the uncertainty of the model, $\boldsymbol{Q}$, and the measurement, $\boldsymbol{R}$, respectively. Both of these are in theory allowed to vary over time, but are in practice kept static.

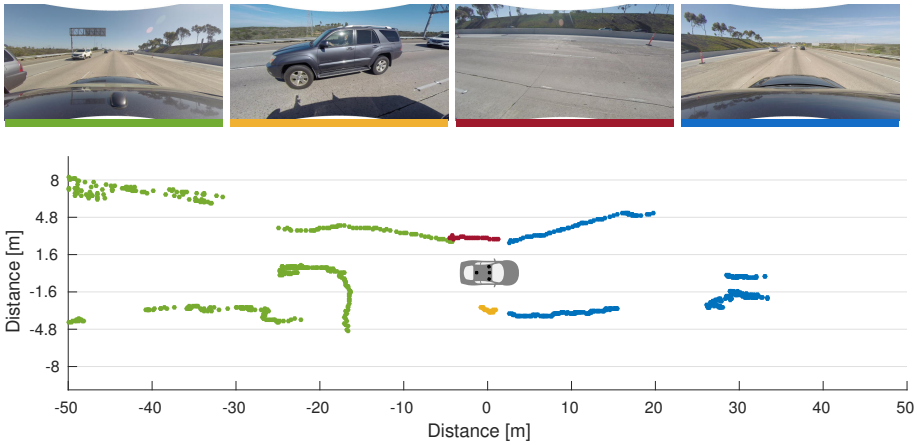
The process noise covariance matrix, $\boldsymbol{Q}$, is a 4x4 matrix that as previously mentioned describes the uncertainty of the model. The matrix has to take into account the random noise that can corrupt the model. In practice, the values of $\boldsymbol{Q}$ are experimentally determined, but note that the fact that the dynamics of vehicles are not completely linear, suggests that the values of $\boldsymbol{Q}$ are increased.

The measurement noise covariance matrix, $\boldsymbol{R}$, is a 2x2 matrix that as previously mentioned describes the uncertainty of the measurement. This matrix describes noise introduced in the process of estimating the position of a vehicle. Thus, the values can be set according to a comparison between ground truth points and estimated points (the variance in distance). However, ground truth points have to undergo the same homography, which is most likely the main contributor of noise in the estimation. This means that the measurement noise covariance will be too low. Instead, the values are experimentally determined. Note that the IPM is more likely to introduce larger errors in the x-direction (since the per pixel distance change increases further away from the ego-vehicle), and thus $\boldsymbol{R}$ needs to make sure that the 2D measurement is not weighted equally. Alternatively, $\boldsymbol{R}$ could change according to how far away from the ego-vehicle the measurement is.

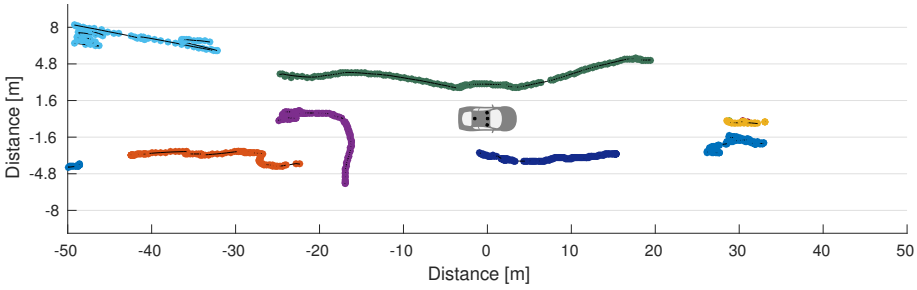
9.2.3 Tracking Vehicles from Multiple Perspectives

With the Kalman filter in hand it is possible to track a vehicle. However, with multiple vehicles on the road, more Kalman filters are needed. Vehicle points are assigned to filters by minimizing the cost in distance between the a priori state estimations (the predicted positions) and the new points in the road plane. A distance threshold, θ_{dis}, rejects assignments that are not close enough to the tracker, $||\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k^-|| > \theta_{\text{dis}}$. Optimally, this threshold is based on the error covariance, as to adjust the threshold according to the uncertainty of the tracker. Alternatively, it is based on the distance from the ego-vehicle in a similar way as $\boldsymbol{R}$. Each time a vehicle is unassigned it initializes a new

filter. Furthermore, vehicle trackers are allowed to survive for a number of frames without any assignments, where they rely on the model to predict positions. This is to ensure that the trackers survive in the regions where vehicles transition from one view to another, and that they survive during full occlusions if they should reappear. An example of tracking in a sequence is shown in Figure 9.13. Note the two vehicles that moves between perspectives



(a) Trajectory points in the road plane colored by perspective in which they are tracked by the MPMDP tracker: Rear, Right, Left, Front.



(b) Kalman filtered trajectories colored by ID. Note for example the green trajectory spanning three different perspectives of the ego-vehicle.

Fig. 9.13: An example of tracking in the road plane in a sequence after 150 frames. The current frame is shown for the four perspectives in the top.

in the adjacent lanes of the ego-vehicle. Both are correctly associated between the perspectives from the available spatiotemporal information.

Another way to implement the tracking in the road plane using Kalman filters, would be to directly assign a MPMDP trajectory to a filter, and handle

the association of Kalman filters in the regions, where vehicles might transition to another perspective. It would also be interesting to study how the system would perform if the single perspective tracker was removed, and detections thereby used directly by the Kalman filters in the road plane. This would have two main challenges compared to the current system: It would not be possible to rely as much on the road points since the single perspective tracker ensures persistence of the trajectories, and the Kalman filters would have to deal with more false positives and negatives; In the current system, the points are filtered in the road plane (to remove noise from IPM) using the trajectory they belong to in order to accommodate the Kalman filtering, thus the Kalman filter would have to deal with more noise making the association more difficult.

The resulting trajectories are based on relative movement to the ego-vehicle, and velocities of surrounding vehicles are thus relative to the ego-vehicle. Absolute velocities can easily be obtained by adding the ego-velocity obtained from the CAN bus, which are useful for several purposes such as analyzing the average speed of surrounding vehicles.

9.3 Evaluation

In this section the road plane tracking (also referred to as 3D tracking) is evaluated using the detectors and single perspective trackers evaluated in Section 7.2 and 8.2, respectively. 3D tracking is the least explored area within evaluation compared to detection and image plane tracking. Leal-Taixé et al. [28] evaluate tracking of pedestrians in 3D in the Multiple Object Tracking Benchmark (MOTChallenge). To do so, they among others use the CLEAR MOT [2] metrics defined with euclidean distance instead of bounding box overlap. Further details of the metrics used for this evaluation are described in Subsection 6.2.3.

The ability to track vehicles in the road plane is evaluated on a manually annotated sequence and ground truth positions in the road plane are obtained using the same homographies as the system. The results are listed in Table 9.1. The most notable result here is that the choice of detector clearly influences the scores, and SubCat achieves a much higher MOTA score than DPM. The large difference seems to origin from the number of false positives according to the precision scores. This is probably due to the lower bounding box precision of the DPM, which causes misplacements in the road plane and

Table 9.1: Multi-perspective tracking results using two different detectors. The arrows indicate if a low or high score is best.

Methods	MOTA↑	MOTEP↓	IDS↓	Frag↓	MT↑	ML↓	Recall↑	Precision↑
SubCat-MPMDP-3D	0.64	1.23	5	93	0.46	0.15	0.79	0.85
DPM-MPMDP-3D	0.42	1.23	3	83	0.38	0.15	0.74	0.70

can lead to vehicles not being matched with their corresponding ground truth point. It should however be noted that the ground truth trajectories are subject to noise caused by the mapping to the road plane (see Figure 6.3), which toughens the task of correctly associating ground truth with corresponding tracks. But since both the system and the ground truth are subject to the same IPM noise, the evaluation reduces the influence of IPM in the scores.

To highlight the performance of association between perspectives an additional test is used. In the test, the number of correct associations of vehicles between perspectives in the road plane is compared to the number of ground truth transitions. The results are listed in Table 9.2 and a full table is available in Paper B. Note that the performance of the three trackers are very similar,

Table 9.2: Associations between perspectives using three different single perspective trackers in 11 sequences compared to ground-truth. See the full table in Paper B.

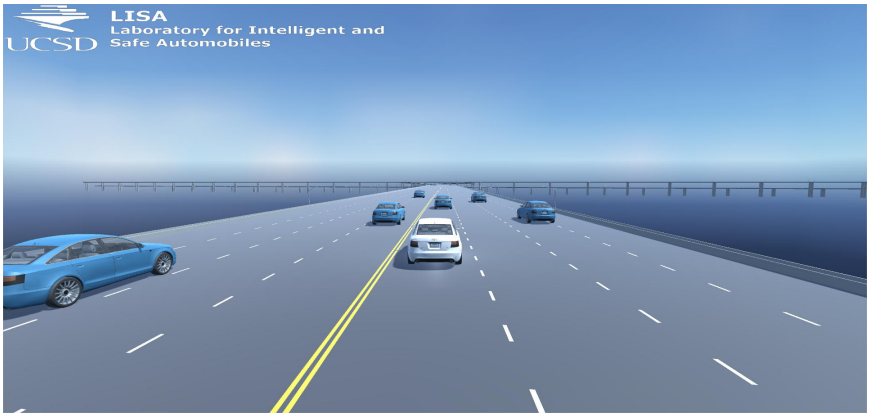
Methods	System/GT	Recall
DPM-MPMDP-3D	75/82	0.92
DPM-MDP-3D	74/82	0.91
DPM-TBD-3D	73/82	0.90

and it might be minor details that cause the differences. Nevertheless, the MPMDP tracker shows to get a higher score than its competitors, which is due to the focus on extending the trajectories at the image borders. It is important that trajectories do not change ID between perspectives, as this will split the trajectories and hinder a full surround trajectory analysis. It is thus an important result that the MPMDP tracker helps improve the association between perspectives.

3D Visualization of Trajectories

This section is not part of the actual system, and is only used to show a possible use of the trajectories estimated by the system. The vehicle trajectories in the road plane open up a wide range of possible usages. One of these is

to visualize the scene in a virtual 3D environment. A 3D visualization allows reconstruction of scenarios, and enables the user to move freely around in the scene. A highway scene is created using Unity3D, and used to set up a TCP/IP server that receives road plane positions that are visualized in real-time. This means that the visualization tool can run on a separate machine, and it is possible to use it on-road for real-time rendering of the surrounding vehicles. Note that the latter requires the trajectory estimation to run in real-time as well, which is not currently the case. The ego-vehicle is assumed static, though it is possible to use the ego-velocity obtained from the CAN bus. Thus, movements of vehicles around the ego-vehicle are relative to the ego-vehicle. A powerful aspect of the visualization is that it is possible to see how certain scenarios unfold in time, and instead of having to keep an eye on four perspectives it is possible to become a part of the world. An example of a time instance from one of the highway sequences is shown in Figure 9.14a and 9.14b. Likewise, it is possible to visualize urban scenes as the ones present in the KITTI dataset, where pedestrians can be tracked as well (see Figure 9.14c).



(a) An example from a highway sequence with tracking of surrounding vehicles.

(b) The user is in control of the camera.

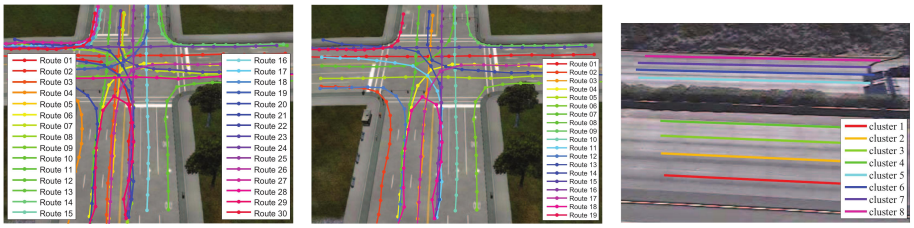
(c) An example of an urban scene.

Fig. 9.14: 3D visualization of on-road scenarios using Unity3D.

Chapter 10

Trajectory Analysis

Trajectories are now found in the road plane from the previous modules. The next task is to analyze the trajectories in a meaningful manner. Setting up well defined heuristic rules for all cases is a cumbersome approach, especially with changing environments. This includes changes from highway to urban driving, or minor changes as simply the number of lanes available which changes the dynamics of the scene and new rules need to be applied. Instead a data-driven approach is examined, learning automatically with no or little human intervention of how to group and model trajectories. This enables classification of new trajectories to be used in both offline and online applications. Offline applications include automatically classification of immense amounts of NDS data, where online applications include abnormality detection and ultimately near-future prediction of surrounding vehicles.

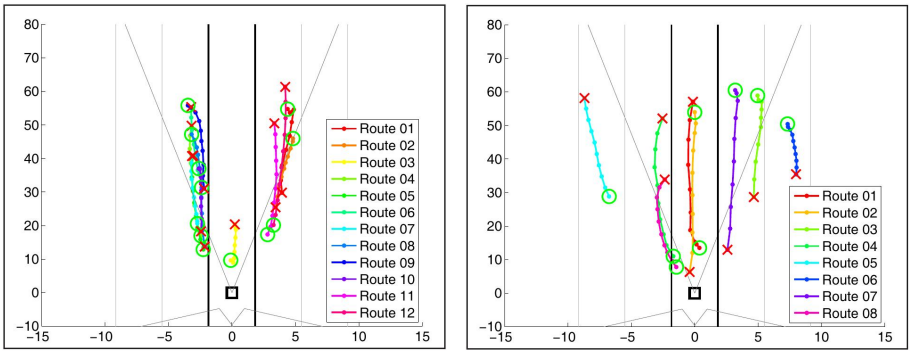


(a) **(b)** **(c)**

Fig. 10.1: Example use of spectral clustering on a simulated interscetion (a), clustering trajectories into an excess number of clusters. An agglomerative merge is used to merge similar clusters into the correct number of clusters (b). In (c) the approach is tested on real data captured at a U.S. highway [35].

So far the research have primarily been focused on traffic surveillance applications, learning routes of intersection as shown in Figure 10.1 for a simulated intersection and data from a real highway.

Less research is found on-road using moving platforms. In [45] a front-faced stereo-camera is used and fused with radar data to predict lane changes in front of the ego-vehicle up to 2.2 seconds ahead. Fully vision-based examples include [37] and [46], learning behaviors in an unsupervised manner in a rear and front perspective, respectively. In this work we expand the framework of Morris and Trivedi [35] from looking at single perspectives to a full surround, resulting in longer and more comprehensive trajectories. Both an unsupervised clustering, and a supervised classification are evaluated, together with an online classification example.



(a) Dense Traffic **(b)** Free-flow Traffic

Fig. 10.2: Typical maneuvers found on-road in a front perspective at both dense and free-flow traffic. With dense traffic only the adjacent lanes are visible due to occlusion, and a short distance to the vehicle in front. With free-flow traffic more lanes are visible, and the dynamics are shown using O to mark the beginning, and X to mark end position, showing overtaking vehciles on the left and receeding vehicles on the right of the ego-vehicle. [46].

10.1 Unsupervised Learning

Clustering trajectories includes several challenges. The main challenge is the variation in both length and start/stop positions. Normalization is an option for the offline classification task, but less suitable in active application, and might lose the temporal information contained in the trajectory. Common clustering algorithms e.g. K-means, is used to cluster points and

feature vectors, where the trajectories contain a temporal aspect as well. This is why spectral clustering is found suitable since this method relies on eigen-decomposition of a similarity matrix. The similarity matrix can be constructed as it would using points or features, by using a similarity measure suitable for trajectories. The spectral clustering is first tested on simulated data and then applied on real data.

10.1.1 Spectral Clustering

The similarity measure used to construct the similarity matrix is reported to be of high importance for a successful clustering [4, 36]. In general the longest common subsequence (LCS) is found useful for trajectory comparison using only positions.

The LCS can be seen as an extended version of the dynamic time warping (DTW). In DTW all elements are matched, even outliers, which is not a requirement using the LCS. LCS is best explained using strings, even though the principle directly applies to the trajectory domain as well. The main difference between comparing strings and trajectories is the matching criteria of the LCS algorithm in Equation 10.1. The first case is a sanity check, and returns zero if either of the strings length is zero. A matrix is instantiated of size $T_i + 1 \times T_j + 1$, as seen in Table 10.1, where T_i and T_j are the lengths of the two strings. An extra row and column is inserted to ensure the generality of the algorithm. The matrix is traversed in a double loop starting row-wise. First, the 't' in trajectory is compared to the 't' in tractor. This is a match, and one is added to the entry plus the value of the top-left cell. Next, the 't' in trajectory is compared with the 'r' in tractor. This is no match, and the maximum of the left and top cell is stored. This is repeated throughout the entire matrix, resulting in an $O(n \times m)$ computational complexity. The end results is seen in Table 10.2 where the LCS of 'trajectory' and 'tractor' is found to be seven.

$$LCS(F_i, F_j) = \begin{cases} 0 & if\ T_i = 0 | T_j = 0 \\ 1 + LCS(F_i^{T_i-1}, F_j^{T_j-1}) & if\ d_E(f_{i,T_i}, f_{j,T_j}) < \epsilon \quad \& \quad |T_i - T_j| < \delta \\ max(LCS(F_i^{T_i-1}, F_j^{T_j}), LCS(F_i^{T_i}, F_j^{T_j-1})) & \text{otherwise} \end{cases} \tag{10.1}$$

The LCS definition for trajectories allow the matching to stretch in time. An acceptable time window of δ is allowed. A match between two points in

Table 10.1: LCS initialisation between the two strings 'trajectory' and 'tractor'.

		t	r	a	c	t	o	r
	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0

Table 10.2: At the end of the algorithm the longest common subsequence between the two strings '*trajectory*' and '*tractor*' is found to be seven.

		t	r	a	c	t	o	r
	0	0	0	0	0	0	0	0
t	0	1	1	1	1	1	1	1
r	0	1	2	2	2	2	2	2
a	0	1	2	3	3	3	3	3
j	0	1	2	3	3	3	3	3
e	0	1	2	3	3	3	3	3
c	0	1	2	3	4	4	4	4
t	0	1	2	3	4	5	5	5
o	0	1	2	3	4	5	6	6
r	0	1	2	3	4	5	6	7
y	0	1	2	3	4	5	6	7

two trajectories is further constrained based on the Euclidean distance being less than ϵ. Both the ϵ and δ threshold is pre-defined and application dependent. For matching trajectories in the road plane an ϵ of half a lane width is used, and a δ of 150 samples. An illustration of LCS is found in Figure 10.3 with two trajectories.

The LCS is used in a distance function Equation 10.2, which normalizes using the length of the shortest of the two trajectories. The distance function is wanted low, where the LCS is wanted high and is therefore inverted. The distance function is thereby comparable with other similarity measures.

$$D_{LCS}(F_i, F_j) = 1 - \frac{LCS(F_i, F_j)}{\min(T_i, T_j)} \tag{10.2}$$

The distance is used in a Gaussian kernel function, Equation 10.3, to make the similarity easily adjustable. The trajectory neighborhood σ is used to control how rapidly the similarity falls off. A higher σ gives higher similarity scores.

$$s_{ij} = e^{-D^2_{LCS}(F_i,F_j)/2\sigma^2} \epsilon [0, 1] \tag{10.3}$$

The previous described steps are performed for each trajectory against all other trajectories and stored in similarity matrix S with entries of Equa-

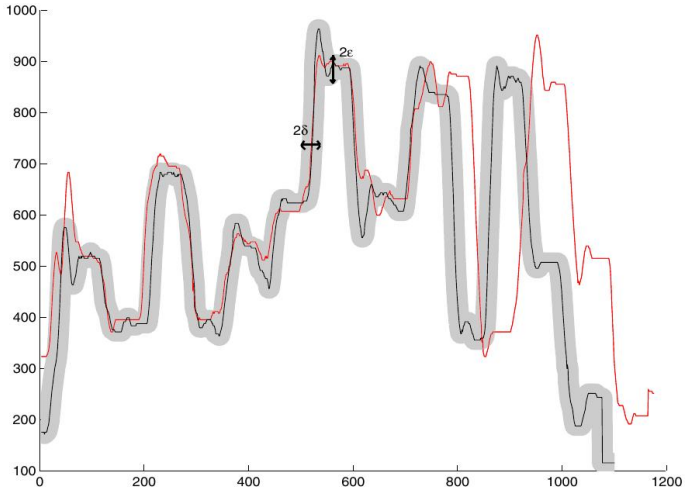


Fig. 10.3: Illustration of LCS between two trajectories [56].

tion 10.3. Given the trajectories easily contain a few hundreds of points, this is a considerable time consuming process. The Laplacian matrix is formed as Equation 10.4, where D is the diagonal degree matrix with entries being row sum of S.

$$L = I - D^{-1/2} S D^{-1/2} \qquad \text{where} \quad D_{ii} = \sum_j S_{ij} \tag{10.4}$$

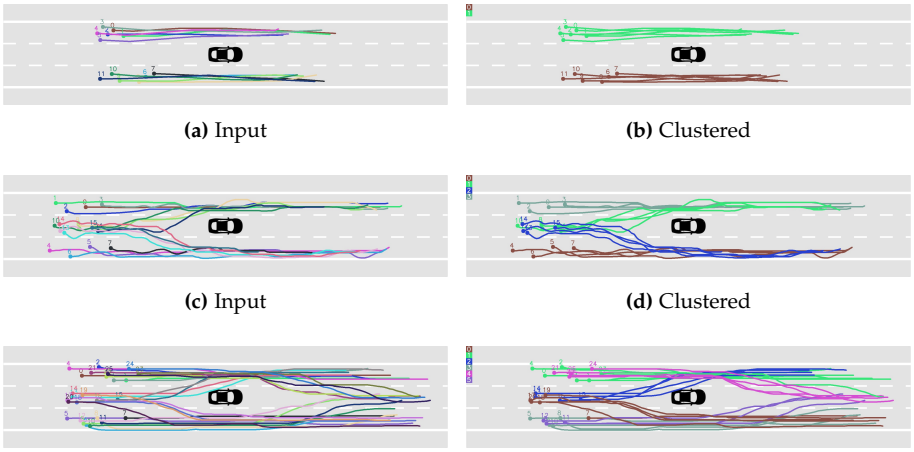
The K largest eigenvalues of L are used as column feature vectors in a new matrix, Q, serving as input to the clustering algorithm. The fuzzy C-means [1] (FCM) clustering algorithm is used, instead of the traditional K-means, to utilize soft cluster membership to minimize the effects of outliers. Both of these clustering algorithms takes the number of clusters as input. As the actual number of clusters is unknown, the number of clusters used is scanned in a range. The FCM toolbox furthermore outputs a metric telling how clean the clustering is, called the fuzzy partition coefficient (FPC), defined in the range from 0 to 1, with 1 being best.

10.1.2 Clustering Simulated Data

The spectral clustering is first tested on simulated data. The simulated data is generated by drawing the trajectories by hand using a simple customized

[1] `https://pypi.python.org/pypi/scikit-fuzzy`

GUI. The trajectories have unequal length, and the start and stop locations can be variated as wanted. The trajectories are drawn as if they belong to a pre-defined cluster and the correct number clusters is thereby known. Some example cases are shown in Figure 10.4.



(a) Input **(b)** Clustered

(c) Input **(d)** Clustered

(e) Input **(f)** Clustered

Fig. 10.4: Clustering simulated trajectories using spectral clustering and fuzzy C-means. The trajectories are numbered by id and the start is marked with a dot to indicate direction.

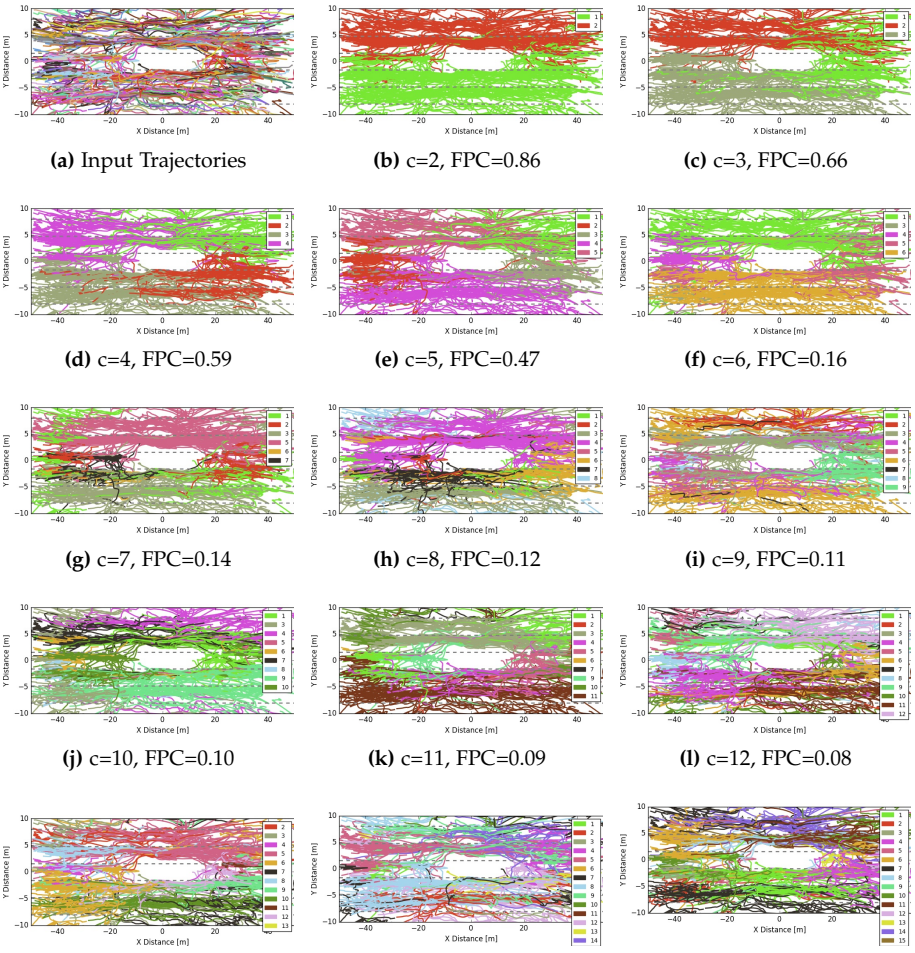
The clustering is in general well separated using the simulated data, especially with the two first cases. Some errors are found in the last case. This is seen for ID number 7, belonging to a wrong cluster, as with a green trajectory in front doing a cut-in maneuver.

10.1.3 Clustering Real-Data

The spectral clustering with FCM seems to overall perform well and the method is therefore applied to real data as seen in Figure 10.5. A range of different cluster numbers, c, is shown since the number of clusters are unknown. A tendency to cluster trajectories in the diagonal directions seen from the ego-vehicle, which might be caused by these areas have a higher amount of fragmented trajectories. Trajectories in these areas will have a high similarity measure. To prioritize long trajectories it could be considered to normalize using the maximum length of the two compared trajectories instead of the original minimum length. One could fear the clustering to be divided as the cameras coverage areas as this would be a sign of error prone

between perspective association. This is though not the case as seen from Figure 10.5d, which somewhat verifies the robustness of multi-perspective association.

A remarkable drop is seen in FPC score from five to six clusters, which can also be seen by the clustering changing pattern. It is noted how some clusters do not have any members and therefore not included in the figure legend. This indicates the data is not easily separable or a bad initialization of clusters.



(a) Input Trajectories **(b)** c=2, FPC=0.86 **(c)** c=3, FPC=0.66

(d) c=4, FPC=0.59 **(e)** c=5, FPC=0.47 **(f)** c=6, FPC=0.16

(g) c=7, FPC=0.14 **(h)** c=8, FPC=0.12 **(i)** c=9, FPC=0.11

(j) c=10, FPC=0.10 **(k)** c=11, FPC=0.09 **(l)** c=12, FPC=0.08

(m) c=13, FPC=0.07 **(n)** c=14, FPC=0.07 **(o)** c=15, FPC=0.06

Fig. 10.5: Spectral clustering using fuzzy C-means on real-data. The number of clusters, c, is specified along with a clustering score FPC where higher is better.

Several improvements can be made to the unsupervised clustering which might give better results, but they are not tested in this study. A more careful parameter tuning might give better trajectory similarities. Note it takes approximate three hours to calculate the LCS similarity matrix for each iteration using a couple of hundreds trajectories, without utilizing dynamic programming. As seen Morris and Trivedi [35] the simulated data is clustered into an excess number of clusters, followed by an agglomerative merging of similar clusters. This does however not seem too promising.

The unsupervised clustering does not give the wanted result since the clusters are not directly comparable with the maneuvers a human observer would specify. Any further analysis of the trajectories using the clustering of the current state is questionable and a supervised method is examined instead.

10.2 Supervised Learning

Even though an unsupervised learning approach may seem appealing due to the ability to adjust to different scenarios, a supervised learning approach might be sufficient if the road structure is known beforehand, as often found on highways being highly structured. For the purpose of trajectory analysis, we are often interested in human defined maneuvers, where an unsupervised method might define different clusters as seen in the previous section. A supervised approach is therefore examined in the remainder of this section.

Trajectories are found in a wide variety surrounding the ego-vehicle as seen from Figure 10.6. It is nearly impossible to define well defined classes to everyone of them.

A case scenario is therefore defined, only evaluating vehicles approaching from the rear with respect to the ego-vehicle. This is seen as a potential dangerous situation because of multiple blindspots, and since a drivers primary focus is in front of the ego-vehicle. In this case scenario five classes are defined as illustrated in Figure 10.9. The trajectory dataset is manually traversed and classified functioning as ground truth. These selected trajectories are later divided into training and test data, with the aim of training a method to classify test trajectories correctly.

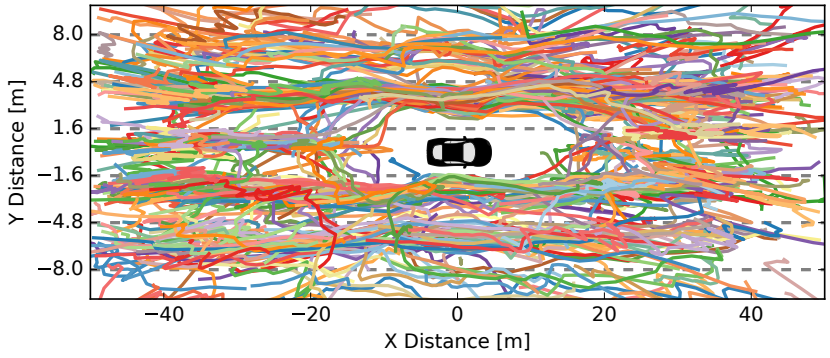


Fig. 10.6: All trajectories found in the dataset.

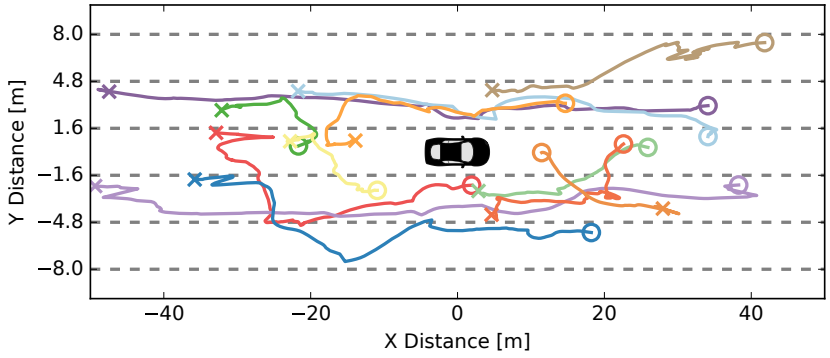


Fig. 10.7: Few selected trajectories of which considered simple consiting of either staying in lane or single lane changes.

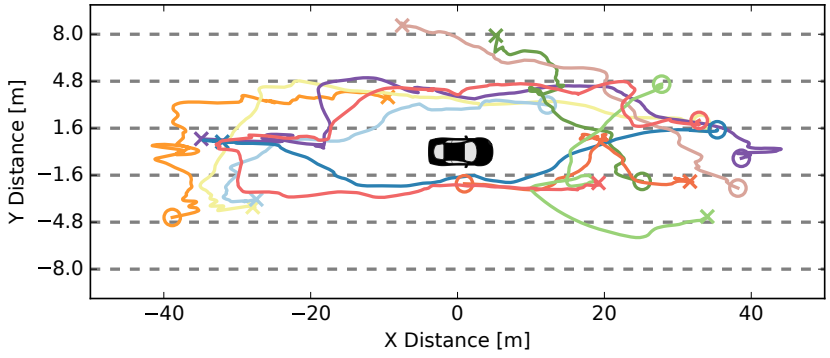
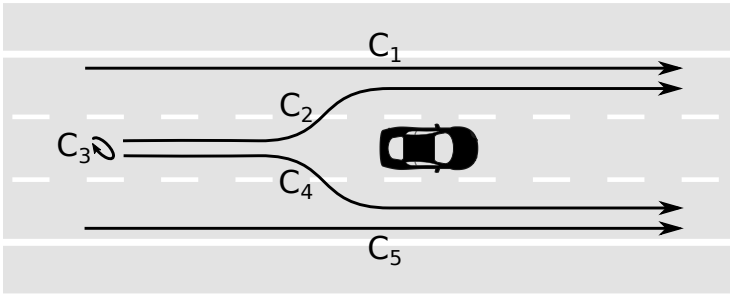


Fig. 10.8: Few selected trajectores advanced trajectories with multiple lane changes.



(a) Trajectory classes

Fig. 10.9: Classes used for the classification of vehicles approaching from behind. C_1) Overtaking on the left. C_2) Left lane change followed by an overtaking on the left. C_3) Staying in ego-lane behind ego-vehicle, also known as tailgating. C_4) Right lane change followed by an overtaking on the right. C_5) Overtaking on the right.

10.2.1 Hidden Markov Models

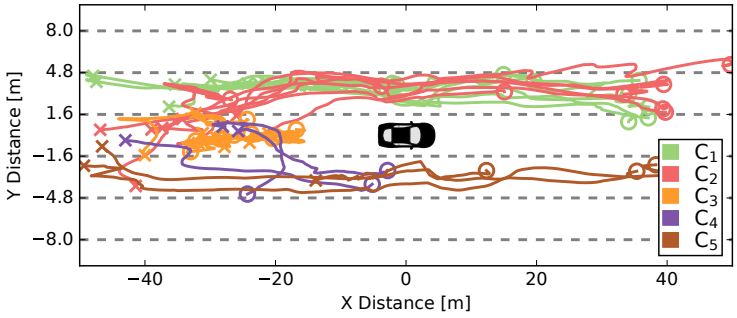
In this work the Hidden Markov Model (HMM) is chosen for its applications in both active and passive safety. The passive safety applications include statistical counts of events based on a classification task. The HMM is further suitable for online classification with real-time capabilities for active safety systems. This also gives a framework predicting vehicle locations ahead of time for early collision warnings.

The annotated trajectories are divided into training and test data. The training set is further expanded by using mirrored trajectories. As for the class of C_2, some cut-in trajectories found in front of the ego-vehicle are used by flipping the x-axis, and reversing the ordering of points in the trajectory. This trick is only used for training the models and not as test trajectories. The training trajectories can be seen in Figure 10.10.

A HMM is trained for each class using the corresponding training trajectories. Each point in a trajectory consists of a feature vector with position in the road surface (x_{road},y_{road}) and velocity ($\dot{x}_{\text{road}}$,$\dot{y}_{\text{road}}$). Note these are the Kalman filtered points from the previous module.

$$\boldsymbol{x}_k = \begin{bmatrix} x_{\text{road}} & y_{\text{road}} & \dot{x}_{\text{road}} & \dot{y}_{\text{road}} \end{bmatrix}^{\text{T}} \tag{10.5}$$

The native MATLAB implementation is using discrete observations, where we have feature vectors with a temporal aspect. Typically, one can either decode the feature vector, or model the feature vector as Gaussian mixtures. The latter approach is implemented in this work using the Probabilistic Modeling



(a)

Fig. 10.10: Training trajectories.

Toolkit (PMTK3)[2]. A HMM is trained by estimating the state transition probability matrix, $\boldsymbol{A}$, emission probability matrix, $\boldsymbol{B}$, and its initial state probability vector, π, as seen in Equation 10.6. The matrices are of size $Q \times Q$, with Q being the number of hidden states used to model each trajectory.

$$\lambda_C = (\boldsymbol{A}_C, \boldsymbol{B}_C, \pi_C) \quad \text{with } C = \{C_1, C_2, C_3, C_4, C_5\} \tag{10.6}$$

The positional distribution of a HMM model using three hidden layers is seen in Figure 10.11. This gives a visual impression of how trajectories can be modeled as transitions between hidden states. A left-right HMM is often applied in surveillance applications, limiting the HMM to only transition from one way to another, not allowing transitioning back to a state once it have left. This could as well be applied in this specific scenario, but is left out to allow vehicles going back and forward in their lane, a case that is not seen in traditional surveillance perspectives i.e. monitoring vehicles on a highway from a static bird point of view will not change direction throughout its trajectory.

A new test trajectory is classified by comparing it to each of the pre-trained models, choosing the one with the highest likelihood as seen from Equation 10.7 and 10.8. The likelihood is calculated using forward algorithm, multiplying the likelihood of an observation to originate from each state. This chain of multiplications gives a very small number, given the likelihood is between zero and one. Longer observation sequences result in smaller numbers. The logarithmic likelihood is therefore used to avoid underflow in

[2] `https://github.com/probml/pmtk3`

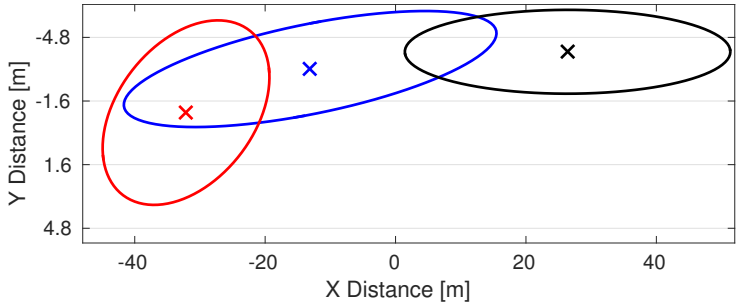


Fig. 10.11: Lillipad plot of the spatial gaussian mixture of C_2 with three hidden layers.

the machine precision.

$$LL_C = \log\ P(O|\lambda_C) \tag{10.7}$$

$$\hat{C} = \arg\max_C LL_C \tag{10.8}$$

The HMM classified test trajectories are shown in Figure 10.12 and evaluated in Table 10.3. Both precision and recall is generally high in all five classes in this simplified scenario with limited number of trajectories. The most noticeable error is the number of false positives for C_2, which is expected to be improved with more training data.

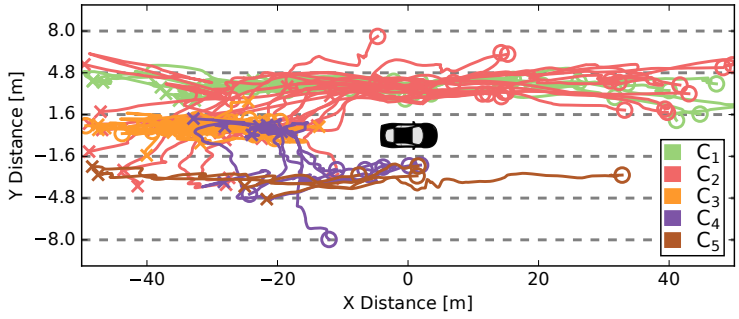


Fig. 10.12: HMM classified test trajectotries.

One downside of the HMM classification approach is found in the overlap between classes. This is seen especially evident for C_3 overlapping with both C_2 and C_4. This gives a higher risk of misclassification since a trajectory of C_3 is easily associated with one of the leftmost hidden states of either C_2 or C_4. One way to combat this in a offline application is to include the requirement to visit all states for a trajectory to be classified. A similar solution could be to

Table 10.3: Classification score for the HMMs on a 2.5 hour drive with selected sequences.

Class	TP/GT	Recall	FP	Precision
C_1	17/21	81%	0	100%
C_2	16/16	100%	5	76%
C_3	18/19	95%	0	100%
C_4	6/6	100%	1	86%
C_5	4/5	80%	0	100%
Total	61/67	91%	6	91%

apply a second classifier on the hidden states visited, or to rely on additional features in spatial overlapping regions.

Given the high classification scores, we can continue with the analysis of the HMM classified trajectories. In the application of NDS analysis one might be interested in reducing the drive into events. These events are almost given when the classes are defined in a supervised manner. The class of C_2 is in this study reduced to two events being a lane change to the left followed by a pass on the left. The event reduced drive is summarized in Table 10.4. It is clear it is more common to overtake on the left than on the right. Overtaking on the right is generally found to be bad driving behavior, but still considerable number is found during 2.5 hour drive, which testifies the need to check for overtaking on the right as well.

Table 10.4: Trajectory analysis results.

Description	Value
Passes - left/right	33/10
Lane changes - left/right	16/6
Tailgating	18
Average velocity of ego-vehicle	102kph
Average velocity of C_1/ego	115kph / 104kph
Average velocity of C_2/ego	113kph / 99kph
Average velocity of C_3/ego	105kph / 104kph
Average velocity of C_4/ego	109kph / 106kph
Average velocity of C_5/ego	109kph / 97kph

Having access to the velocity of each class further enriches the analysis. The ego-vehicle is overall found to be driving within the speed limits of 105

kph (65 mph), where vehicles approaching from behind is found to be driving faster than the ego-vehicle, except tailgating vehicles. Vehicles on the left is found to drive faster than those on the right, which might be caused by the higher awareness of overtaking vehicles on the common left side.

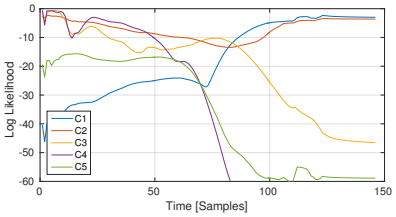
Further work includes more data for training and testing, testing the number of states together with other features.

10.3 Online Classification

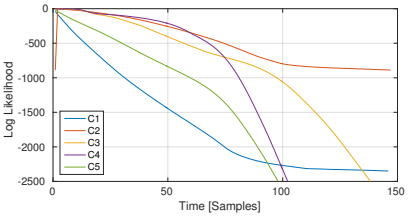
Analyzing trajectories are not only interesting classifying complete trajectories for passive safety in NDSs, but certainly also in real-time applications for active safety as in ADAS. In this work the same HMMs from previously are used. The input can either be the last sample only, the last n samples, or the samples available up to the current time. This functions as a compromise between an early classification and certainty.

A scenario is shown with an approaching vehicle from the rear as seen in Figure 10.13e and Figure 10.13f as it is about to change lane following C_2. The challenge is to determine a transition from one model to another as soon as possible i.e. when are the vehicle changing lane. The likelihood for each increasingly available sample is shown in Figure 10.13a using only the last available sample, and Figure 10.13b using all the available samples up to the current time. It is noticed how using only the last sample gives a more sporadic likelihood curve with several transitions, where using all available samples gives a more smooth curve. The trajectory of the vehicle seen in the road plane is shown in Figure 10.13c and Figure 10.13d marked with red at the transition to C_2, and corresponding frame at Figure 10.13e and Figure 10.13f. It is not obvious from these images that a lane change is about to take place. The decision making can be assumed to be based heavily on the velocity, which will be higher in both the x and y-direction during an overtake.

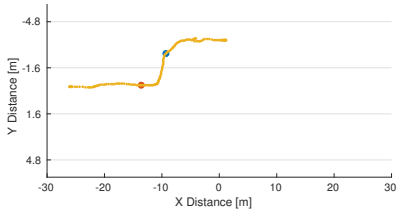
One thing is to be able to recognize behaviors, but likewise importantly is the detection of abnormal behaviors, which is considered a higher risk to the ego-vehicle. Using the HMMs, abnormal maneuvers would be detected if the maximum likelihood among all classes is below a certain threshold. This threshold is not easily determined with trajectories of different lengths, and either a window function could be used, or a threshold as a function of trajectory length.



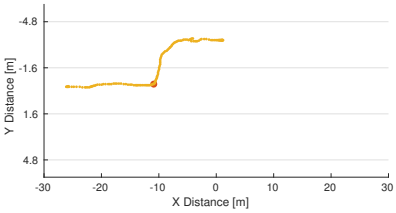
(a) Log likelhihood to originate from each of the five classes using only the last available sample. Transition to C_2 at sample 47.



(b) Log likelyhood to originate from each of the five classes using all available samples up the current time. Transition to C_2 at sample 60.



(c) Trajetory in road plane with marked transition to C_2 shown in red using only the last sample.



(d) Trajectory in road plane with marked transition to C_2 shown in red using all available samples.

(e) Corresponding frame of the transition to C_2 using the last sample.

(f) Corresponding frame of the transition to C_2 using all available samples.

Fig. 10.13: Online classification comparison between using only the last available sample and alle currently available samples.

Chapter 11

Concluding Remarks

It is in this part proved that a vision-based framework is able to track vehicles all the way around the ego-vehicle. Full surround is achieved and overlap is achieved using four GoPro cameras mounted on the rooftop rails of the testbed. The data is post-processed correcting for synchronization and distortion. A larger data collection process have been initiated with a novel multi-perspective dataset captured at 2.7K resolution being higher than any previous traffic datasets. Over 4000 vehicles are annotated including occlusion and truncation tags.

A DPM detector trained on the KITTI dataset is evaluated and compared to the state-of-the-art SubCat detector. Overall the detectors score acceptable with SubCat detecting with the highest precision. With computational times taken into consideration the SubCat dominates the DPM. Both side perspectives score considerably lower showing there is still room for improvements. The detection of truncated vehicles is found of increased importance with the aim of detecting vehicles in multiple adjacent perspectives. Truncated vehicles have previously been of less importance since these are often only evaluated with little to no truncation.

The MDP tracker is used as it is found to be one of the highest scoring on the KITTI tracking evaluation. The freely available MDP code is originally aimed at pedestrian detection, where several modifications are made to track vehicles and optimized for multi-perspective tracking. These modification include re-routing the MDP states to avoid directly tracking all detections thereby reducing the number of false positives. The consequence is the tracking starting a few samples later, which can be problematic for the

side perspectives where vehicles might only be present for a limited number of frames. Another challenging case is vehicles staying in the overlapping regions. These vehicles are truncated in both views and often lack detections and thereby difficult to track. Tracks are prolonged near the image border by prediction using the last couple of tracked samples to assist the association.

IPM is used to project the middle of bottom bounding box to a common road plane. IPM is used under the assumption of a level surface which is suitable for the data used. Points in the road plane are assigned to Kalman filters thereby associating. No common metrics are used in the evaluation of tracking in 3D coordinates. It is proposed to re-define the MOTP to use an euclidean distance threshold instead of bounding box overlap. The tracker using SubCat detections were found to have an significantly higher MOTA and precision compared to DPM. The association were tested scoring 92% correct, showing the robustness of the association.

The estimated trajectories are analyzed using both a supervised and unsupervised approach. The unsupervised method using spectral clustering performed well on simulated data, but unable to find satisfying clusters in real data. The real data is shown to be complex with various types of maneuvers. A simplified scenario is defined consisting of five classes of trajectories found for vehicles approaching from the rear. An HMM was used to classify full length trajectories with 91% precision and recall. Lastly an online classification task is examined, recognizing a lane change as early as possible using only the last sample or all available samples. A compromise is found between an early recognition and certain decision.

Part IV

Closing

Chapter 12

Conclusion

This master's thesis is the product of a seven months visit to the Computer Vision and Robotics Research Laboratory located at University of California, San Diego. The research is furthermore documented in an accepted conference paper, a journal paper submitted for review, and a conference paper intended for submission. The focus of the study is to observe vehicles from a camera-equipped test vehicle. Overall topics of the research include data collection and evaluation, computer vision, and machine learning.

Several large-scale naturalistic driving studies have through the years sought to answer why fatal vehicle crashes happen. Most of these focus on the awareness of drivers, which suggests that the state of the drivers is an important factor in crashes and near crashes. Because of limits to human sensing and the fight for attention, drivers are not always able to maintain an adequate overview of surrounding vehicles.

This thesis describes how multiple visual cameras can be used to achieve a full surround view of a vehicle, potentially enabling intelligent warning or assistance of drivers in dangerous situations. Throughout the study, data that have been collected on highways in the San Diego area, are used to develop and evaluate methods on realistic and challenging scenarios. Two different setups, using four and six cameras respectively, are designed and used for collection of data. Additional to data collection, the thesis consists of two essential topics – multi-perspective vehicle trajectory estimation and automatic analysis of trajectories.

The estimation of vehicle trajectories from multiple visual cameras is of main concern in this work, as a central goal is to prove the competitiveness

of visual cameras in the field of full surround sensing, where lidar and radar are currently the leading technologies. Indeed, this work confirms that it is possible to estimate trajectories, not only in a front perspective, but all the way around a vehicle. To this end, the system takes advantage of two major fields within computer vision – detection and tracking. By imposing certain constraints and assumptions it is shown possible from the four monocular perspectives to estimate trajectories in 3D, thus taking the trajectories from images to real world coordinates.

The trajectories have a wide range of potential uses, and contain valuable information to both naturalistic driving studies and advanced driver assistance systems. In this work, it is shown how supervised learning of different types of trajectories from vehicles approaching from behind, can be used to classify new trajectories, and potentially for predicting which paths vehicles will follow. The latter is in particular of interest for future applications, since long-term predictions can be a great tool for safer driving.

Chapter 13

Directions of Future Work

This study has proved promising results of using a multi-perspective camera setup to observe surrounding vehicles. In order to bring the concept closer to a real world application, several ideas can be investigated.

The execution time of the methods have not been prioritized in this work, and thus the system does not run in real-time. However, development within faster detection and tracking methods, and the fast evolution of hardware specifications, suggest that it will be possible to run the system in real-time in the near future. Real-time execution is a unavoidable requirement for ADASs. NDSs on the other hand, can benefit from the findings of this study directly.

By expanding the dataset with more data possibly from other locations with different weather settings at different times of the day (e.g. nighttime driving), a more profound evaluation would be possible. Note that this study deals with data specifically selected to expose strengths and weaknesses of methods for multi-perspective tracking. Thus, the data might not be representable for general driving. An expansion to more scenarios will strengthen the contribution, and addition of for example urban environments would greatly broaden the usability. Urban environments would further require an automatic understanding of the surrounding road structure.

A very appealing next step of the concept is a comparison with established technologies such as lidar and radar. By combining the strengths of each modality in a multi-modal fusion framework, a potentially superior system would be in possession of great tools for surround trajectory understanding. One of the disadvantages of the active sensors (lidar and radar) is that the

data are not intuitive to human interpretation. Visual cameras on the other hand, allow for a simple human-computer-interface. Thus, it also needs to be considered how the driver is included in the loop.

Bibliography

[1] American Association of State Highway and Transportation Officials (AASHTO) [2005], 'A Policy on Design Standards Interstate System'.

[2] Bernardin, K. and Stiefelhagen, R. [2008], 'Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics', *Journal on Image and Video Processing* .

[3] Bertozzi, M. and Broggi, A. [1998], 'GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection', *IEEE Transactions on Image Processing* **7**(1), 62–81.

[4] Besse, P. C., Guillouet, B., Loubes, J.-M. and Royer, F. [2016], 'Review and perspective for distance-based clustering of vehicle trajectories'.

[5] Bouguet, J.-Y. [2000], 'Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker Description of the algorithm'.

[6] Boyle, L. N., Hallmark, S., Lee, J. D., McGehee, D. V., Neyens, D. M. and Ward, N. J. [2012], Integration of Analysis Methods and Development of Analysis Plan, Technical report, Transportation Research Board of the National Academies.

[7] California Department of Transportation (Caltrans) [2014], 'California Manual on Uniform Traffic Control Devices for Streets and Highways'.

[8] Dalal, N. and Triggs, B. [2005], Histograms of Oriented Gradients for Human Detection, *in* '2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 1.

[9] Dingus, T. A., Klauer, S., Neale, V., Petersen, A., Lee, S., Sudweeks, J., Perez, M., Hankey, J., Ramsey, D., Gupta, S. et al. [2006], The 100-

Car Naturalistic Driving Study, Phase II-Results of the 100-Car Field Experiment, Technical report.

[10] Dollár, P. [n.d.], 'Piotr's Computer Vision Matlab Toolbox (PMT)', `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

[11] Dollár, P., Appel, R., Belongie, S. and Perona, P. [2014], 'Fast Feature Pyramids for Object Detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(8).

[12] Ehlgen, T., Pajdla, T. and Ammon, D. [2008], 'Eliminating Blind Spots for Assisted Driving', *IEEE Transactions on Intelligent Transportation Systems* **9**(4), 657–665.

[13] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. [2007], 'The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results', http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[14] Felzenszwalb, P. F., Girshick, R. B. and McAllester, D. [n.d.], 'Discriminatively Trained Deformable Part Models, Release 4', http://people.cs.uchicago.edu/ pff/latent-release4/.

[15] Felzenszwalb, P. F., Girshick, R. B., McAllester, D. and Ramanan, D. [2010], 'Object Detection with Discriminatively Trained Part Based Models', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**.

[16] Ferryman, J. and Shahrokni, A. [2009], Pets2009: Dataset and challenge, *in* 'IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)'.

[17] Garcia, F., Cerri, P., Broggi, A., de la Escalera, A. and Armingol, J. M. [2012], Data Fusion for Overtaking Vehicle Detection based on Radar and Optical Flow, *in* 'IEEE Intelligent Vehicles Symposium'.

[18] Geiger, A., Lauer, M., Wojek, C., Stiller, C. and Urtasun, R. [2014], '3D Traffic Scene Understanding From Movable Platforms', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(5).

[19] Geiger, A., Lenz, P., Stiller, C. and Urtasun, R. [2013], 'Vision meets Robotics: The KITTI Dataset', *The International Journal of Robotics Research* .

[20] Geiger, A., Lenz, P. and Urtasun, R. [2012], Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, *in* 'IEEE Conference on Computer Vision and Pattern Recognition'.

[21] Girshick, R. [2012], From Rigid Templates to Grammars: Object Detection with Structured Models, PhD thesis, University of Chicago.

[22] Girshick, R. B., Felzenszwalb, P. F. and McAllester, D. [n.d.], 'Discriminatively Trained Deformable Part Models, Release 5', `http://people.cs.uchicago.edu/~rbg/latent-release5/`.

[23] Hartley, R. and Zisserman, A. [2003], *Multiple View Geometry in Computer Vision*, 2 edn, Cambridge University Press.

[24] Jensen, M. B., Philipsen, M. P., Møgelmose, A., Moeslund, T. B. and Trivedi, M. M. [2016], 'Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives', *IEEE Transactions on Intelligent Transportation Systems* **PP**(99).

[25] Kalal, Z., Mikolajczyk, K. and Matas, J. [2010], Forward-backward error: Automatic detection of tracking failures, *in* 'International Conference on Pattern Recognition'.

[26] Kalal, Z., Mikolajczyk, K. and Matas, J. [2012], 'Tracking-Learning-Detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34**(7).

[27] Kumar, R. K., Ilie, A., Frahm, J.-M. and Pollefeys, M. [2008], Simple Calibration of Non-overlapping Cameras with a Mirror, *in* 'IEEE Conference on Computer Vision and Pattern Recognition'.

[28] Leal-Taixé, L., Milan, A., Reid, I., Roth, S. and Schindler, K. [2015], 'MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking', *arXiv:1504.01942 [cs]* .

[29] Lucas, B. D. and Kanade, T. [1981], 'An Iterative Image Registration Technique with an Application to Stereo Vision', *International Joint Conference on Artificial Intelligence* **81**.

[30] Luettel, T., Himmelsbach, M. and Wuensche, H. J. [2012], 'Autonomous Ground Vehicles - Concepts and a Path to the Future', *Proceedings of the IEEE* **100**(Special Centennial Issue).

[31] Martin, S., Ohn-Bar, E. and Trivedi, M. M. [2015], Automatic Critical Event Extraction and Semantic Interpretation by Looking-Inside, *in* 'IEEE Intelligent Transportation Systems Conference'.

[32] Milan, A., Leal-Taixé, L., Reid, I. D., Roth, S. and Schindler, K. [2016], 'MOT16: A Benchmark for Multi-Object Tracking', *CoRR* **abs/1603.00831**.

[33] Milan, A., Schindler, K. and Roth, S. [2013], Challenges of Ground Truth Evaluation of Multi-target Tracking, *in* 'IEEE Conference on Computer Vision and Pattern Recognition Workshops'.

[34] Møgelmose, A., Liu, D. and Trivedi, M. M. [2015], 'Detection of U.S. Traffic Signs', *IEEE Transactions on Intelligent Transportation Systems* **16**(6).

[35] Morris, B. T. and Trivedi, M. M. [2011], 'Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(11).

[36] Morris, B. and Trivedi, M. [2009*a*], Learning trajectory patterns by clustering: Experimental studies and comparative evaluation, *in* 'IEEE Conference on Computer Vision and Pattern Recognition'.

[37] Morris, B. and Trivedi, M. [2009*b*], Unsupervised Learning of Motion Patterns of Rear Surrounding Vehicles, *in* 'IEEE International Conference on Vehicular Electronics and Safety'.

[38] Neale, V. L., Dingus, T. A., Klauer, S. G., Sudweeks, J. and Goodman, M. [2002], An Overview of the 100-Car Naturalistic Study and Findings, Technical report, VTTI.

[39] Ng, A. Y. and Russell, S. [2000], 'Algorithms for Inverse Reinforcement Learning', *International Conference on Machine Learning* .

[40] Nieto, M., Salgado, L., Jaureguizar, F. and Cabrera, J. [2007], Stabilization of Inverse Perspective Mapping Images based on Robust Vanishing Point Estimation, *in* 'IEEE Intelligent Vehicles Symposium'.

[41] Ohn-Bar, E. and Trivedi, M. M. [2014], Fast and Robust Object Detection Using Visual Subcategories, *in* 'IEEE Conference on Computer Vision and Pattern Recognition Workshops'.

[42] Oliveira, M., Santos, V. and Sappa, A. D. [2015], 'Multimodal inverse perspective mapping', *Information Fusion* **24**, 108–121.

[43] Satzoda, R. K. and Trivedi, M. M. [2015], 'Drive Analysis Using Vehicle Dynamics and Vision-Based Lane Semantics', *IEEE Transactions on Intelligent Transportation Systems* **16**(1).

[44] Satzoda, R. K. and Trivedi, M. M. [2016], 'Looking at Vehicles in the Night: Detection & Dynamics of Rear Lights', *IEEE Transactions on Intelligent Transportation Systems* .

[45] Schlechtriemen, J., Wedel, A., Hillenbrand, J., Breuel, G. and Kuhnert, K. D. [2014], A Lane Change Detection Approach using Feature Ranking with Maximized Predictive Power, *in* 'IEEE Intelligent Vehicles Symposium'.

[46] Sivaraman, S., Morris, B. and Trivedi, M. [2011], Learning Multi-Lane Trajectories using Vehicle-Based Vision, *in* 'IEEE International Conference on Computer Vision Workshops'.

[47] Sivaraman, S. and Trivedi, M. [2013], 'Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis', *IEEE Transactions on Intelligent Transportation Systems* **14**(4).

[48] Takeda, K., Hansen, J. H. L., Boyraz, P., Malta, L., Miyajima, C. and Abut, H. [2011], 'International Large-Scale Vehicle Corpora for Research on Driver Behavior on the Road', *IEEE Transactions on Intelligent Transportation Systems* **12**(4).

[49] Tawari, A., Chen, K. H. and Trivedi, M. M. [2014], Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation, *in* 'IEEE Intelligent Transportation Systems Conference'.

[50] U.S. Department of Transportation [2009], '2009 National Household Travel Survey'.

[51] U.S. Department of Transportation [2014], 'Highway Statistics 2014'.

[52] U.S. Department of Transportation, National Highway Traffic Safety Administration [2013], 'Fatality Analysis Reporting System (FARS)'.

[53] U.S. Department of Transportation, National Highway Traffic Safety Administration [2014], 'Lives Saved in 2012 by Restraint use and Minimum Drinking Age Laws'.

[54] Victor, T., Dozza, M., Bärgman, J., Boda, C.-N., Engström, J., Flannagan, C., Lee, J. D. and Markkula, G. [2015], SHRP 2 Report S2-S08A-RW-1. Analysis of Naturalistic Driving Study Data: Safer Glances, Driver Inattention, and Crash Risk, Technical report, Transportation Research Board of the National Academies.

[55] Viola, P. and Jones, M. J. [2004], 'Robust Real-Time Face Detection', *International journal of computer vision* **57**(2).

[56] Vlachos, M., Kollios, G. and Gunopulos, D. [2002], Discovering similar multidimensional trajectories, *in* 'International Conference on Data Engineering'.

[57] Wang, C., Fang, Y., Zhao, H., Guo, C., Mita, S. and Zha, H. [2016], 'Probabilistic Inference for Occluded and Multiview On-road Vehicle Detection', *IEEE Transactions on Intelligent Transportation Systems* **17**(1).

[58] Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M., Qi, H., Lim, J., Yang, M. and Lyu, S. [2015], 'DETRAC: A New Benchmark and Protocol for Multi-Object Tracking', *CoRR* **abs/1511.04136**.

[59] Wu, B. and Nevatia, R. [2006], Tracking of Multiple, Partially Occluded Humans based on Static Body Part Detection, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', Vol. 1.

[60] Xiang, Y. [2015], 'MDP_Tracking Code', Available Online: `https://github.com/yuxng/MDP_Tracking`.

[61] Xiang, Y., Alahi, A. and Savarese, S. [2015], Learning to Track: Online Multi-Object Tracking by Decision Making, *in* 'IEEE International Conference on Computer Vision'.

[62] Yan, J., Lei, Z., Wen, L. and Li, S. [2014], The Fastest Deformable Part Model for Object Detection, *in* 'IEEE Conference on Computer Vision and Pattern Recognition'.

[63] Yebes, J. J., Bergasa, L. M., Arroyo, R. and Lazaro, A. [2014], Supervised learning and evaluation of KITTI's cars detector with DPM, *in* 'IEEE Intelligent Vehicles Symposium'.

[64] Zhang, B., Appia, V., Pekkucuksen, I., Liu, Y., Batur, A. U., Shastry, P., Liu, S., Sivasankaran, S. and Chitnis, K. [2014], A surround view camera solution for embedded systems, *in* 'IEEE Conference on Computer Vision and Pattern Recognition Workshops'.

[65] Zhang, H., Geiger, A. and Urtasun, R. [2013], Understanding High-Level Semantics by Modeling Traffic Patterns, *in* 'International Conference on Computer Vision'.

[66] Zou, W. and Li, S. [2015], 'Calibration of Nonoverlapping In-Vehicle Cameras With Laser Pointers', *IEEE Transactions on Intelligent Transportation Systems* **16**(3).

Part V

Papers

Paper A

Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies

Miklas S. Kristoffersen, Jacob V. Dueholm, Ravi K. Satzoda,
Mohan M. Trivedi, Andreas Møgelmose, Thomas B. Moeslund

The paper has been accepted for
IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016.

Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies

Miklas S. Kristoffersen[1,2], Jacob V. Dueholm[1,2], Ravi K. Satzoda[2],
Mohan M. Trivedi[2], Andreas Møgelmose[1,2], and Thomas B. Moeslund[1]
[2] University of California, San Diego, USA. rsatzoda@eng.ucsd.edu,mtrivedi@ucsd.edu
[1] Aalborg University, Denmark. {mskr11,jdueho11}@student.aau.dk {am,tbm}@create.aau.dk

Abstract

This paper proposes the use of multiple low-cost visual sensors to obtain a surround view of the ego-vehicle for semantic understanding. A multi-perspective view will assist the analysis of naturalistic driving studies (NDS), by automating the task of data reduction of the observed sequences into events. A user-centric vision-based framework is presented using a vehicle detector and tracker in each separate perspective. Multi-perspective trajectories are estimated and analyzed to extract 14 different events, including potential dangerous behaviors such as overtakes and cut-ins. The system is tested on ten sequences of real-world data collected on U.S. highways. The results show the potential use of multiple low-cost visual sensors for semantic understanding around the ego-vehicle.

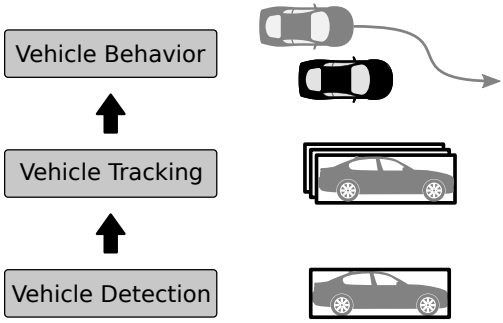


Figure 1. The ascending levels of vehicle interpretation in a vision-based application. At the lowest level is vehicle detection, which locates visible vehicles on a single-camera and single-frame basis. One level up, detections are associated between frames and views, in order to track vehicles on a multiple-camera and multiple-frame basis. At the highest level, the spatio-temporal trajectories are used to classify behaviors of vehicles.

1. Introduction

Trajectories of surrounding vehicles are essential to the extraction of higher-level semantics. Recent scientific progress in visual vehicle detection and tracking allows for robust trajectories [19] that enables us to automate exploration of vehicle behaviors, which has previously been a time-consuming manual hand-labeling process. However, until now, visual cameras have not been used to cover full surroundings of a vehicle with the purpose of estimating trajectories of surrounding vehicles and analyzing maneuvers. In this study we show how existing methods for monocular vehicle detection and tracking adapts to a multi-perspective framework with the purpose of reaching a higher level understanding of surrounding vehicle maneuvers and behaviors as shown in Fig. 1. If successful, these trajectories contain information, which is valuable to naturalistic driving studies (NDS) that seek to answer how drivers behave and why, in order to understand circumstances of crashes and near-crashes. By learning how surrounding trajectories develop over time, it is possible to predict which route the vehicles will probably follow in the near future. The prediction of trajectories is an integral part of path-planning in advanced driver assistance systems (ADAS).

The leading technologies in terms of sensing vehicular surroundings are LiDAR and radar. A lot of research has been conducted in the field using three-dimensional point clouds, consequently enabling autonomous vehicles to successfully drive public roads without causing accidents. However, by introducing low-cost passive visual cameras it is possible to add a level on top of the already existing solutions that rely purely on spatiotemporal positions and shapes. The visual modality contains appearance cues that can help improve the performance, e.g. by detecting brake lights, estimating orientation of vehicles, and recognizing traffic signs and signals. Thus, by using multi-perspective visual cameras together with existing ADAS, it is possible to achieve rich information of surroundings.

The main contributions of this paper can be summarized

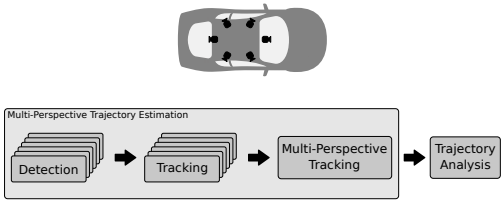


Figure 2. The top image displays the placement of the six synchronised cameras. The bottom image shows the flow of the system from the input of six video sequences to the output of a trajectory analysis.

as follows: (1) Using six cameras, we develop a framework for estimating vision-based multi-perspective trajectories on a moving platform. The method has three steps: Vehicle detection in six different perspectives, vehicle tracking between frames in the six perspectives, and multi-perspective tracking that connects the trajectories across perspectives; (2) The multi-perspective trajectories are analyzed for semantics of surrounding vehicular events. We show how the combination of six perspectives, a top-down visualization of trajectories, and a list of events that have occurred, can be used as a powerful tool to interpret higher-level semantics of the surrounding vehicular maneuvers; (3) A vehicle equipped with six cameras is used to capture several hours of free-flow highway driving. We show a real-world study of 10 sequences chosen to prove the potential of the system.

2. Related Work

High-level semantics have previously been analyzed, identifying maneuvers as overtakes, lane-changes, cut-ins, cut-outs, or simply staying in lane. Early examples [9] use simulated data, while recently, real data are used in a front view of a moving platform [17, 12, 21], classifying up to 27 maneuvers regarding lane-changes. In [18], both a mono and a stereo camera are used to obtain trajectories in front of the ego-vehicle. The behaviors of the obtained trajectories are then learned using an unsupervised learning approach. A similar approach is seen in [16] with vehicles behind the ego-vehicle. Trajectories are furthermore used to infer traffic patterns in intersections using stereo vision [24, 7]. Estimating trajectories from vision-based sensors can be divided into classic computer vision disciplines as detection and tracking of vehicles. These are well researched fields with public available databases with common benchmarks. Multi-target vehicle tracking is mainly found in KITTI [8] and DETRAC [22], where multi-perspective tracking is mainly found for pedestrian tracking, as seen in Pets2009 [5] with overlapping views, MOT Challenge [14], and MCT Challenge [1] with non-overlapping views. In comparison to trajectories observed from pedestrians with static cameras [13], vehicle trajectories discovered with a multi-camera setup on a moving platform are subject to additional difficulties [18], such as effects of relative motion. Non-overlapping perspectives require the use of re-identification, which is traditionally used in surveillance applications [10]. In the application of tracking surround vehicles, the re-identification problem between perspectives is considerably simplified, since only a limited number of candidates exist, depending on the traffic density.

(a) Front left (b) Front (c) Front right

(d) Rear left (e) Rear (f) Rear right

Figure 3. Sample images captured from the synchronized multi-perspetive setup. Note the challenges of e.g. glare, shadows, and distortion.

Previous studies have detected and tracked vehicles using multi-camera setups. An early example is seen in [6], where an omnidirectional camera together with a pan-tilt-zoom camera are used to detect and classify vehicles. In [3] surrounding vehicles and pedestrians are detected and tracked in a simple low-velocity parking environment. In [20] vehicles are detected around the ego-vehicle in a highway scenario using a method based on the deformable parts model (DPM) [4]. These studies focus on the low-level aspects of detecting and tracking in surround view applications, whereas we in this work furthermore show the potential use of the resulting trajectories as a tool for analyzing the behaviors of surrounding vehicles.

The challenge of associating trajectories between perspectives is studied in [15], where four cameras are used with partial overlap. Trajectories are extracted from each individual camera and projected to a common plane, where trajectories are associated. A similar approach is seen in [2], finding local trajectories and projecting to a common plane and linked if both the spatio-temporal features match.

3. System Overview

The synchronized data used in this work are collected on U.S. highways in California. The vehicle used for data collection is equipped with six Point Grey cameras and a

GPS tracker. Furthermore, data are logged from the controller area network (CAN) bus. The six cameras are placed strategically around the vehicle, as shown in Fig. 2, in order to achieve a full surround view as seen in Fig. 3. The front and rear cameras are considered the most important in the process of estimating the multi-perspective trajectories, for which reason they are capturing with a resolution of 1280×960. The two cameras use low-distortion lenses with horizontal field of view of $70°$ and $80°$, respectively. The four side view cameras are captured at a lower resolution of 640×480, to achieve a frame rate of 15 frames per second (FPS) for the synchronized data collection. The side view cameras are mounted with wide angle lenses with a horizontal field of view of $135°$, to ensure a full surround coverage with overlapping views, at the cost of a higher degree of distortion.

A flow diagram of the system is shown in Fig. 2. Vehicle detection is performed for each of the six inputs of the cameras. The detections in each perspective are used by the vehicle tracker, which associates the detections between frames for each of the six perspectives. The trajectories are connected between perspectives, and finally an analysis of the multi-perspective trajectories is performed.

4. Multi-Perspective Trajectory Estimation

In the following section we present the methods designed for estimating trajectories of vehicles present in surroundings of the ego-vehicle using six different visual perspectives.

4.1. Vehicle Detection

Visual vehicle detection is a well researched topic that has seen recent scientific progress, but is not yet considered a solved problem. In this work we use six different perspectives from the same location on a moving platform, and are thus subject to variances in capturing such as the viewpoint of vehicles, lighting, shadows, and glare. An example of these challenges is shown in Fig. 3. The side views are especially challenging with lower resolutions and severe distortion caused by the wide angle lenses. The multi-perspective challenges require the vehicle detection to be either one versatile detector, or to use a separate detector optimized for each perspective.

In this work we use the model-based Deformable Parts Model (DPM) detector [4] in a two-stage implementation presented in [24, 7]. The implementation includes a pretrained vehicle model trained on the KITTI dataset [8], which is used for all six perspectives. The first stage is a regular DPM detector, while the second stage detects vehicles in an upscaled version of the image in an area around the horizon. The horizon is specified for each of the perspectives. Detections for both stages are combined in a non-maxima suppression.

We have a set of captured video sequences in the time interval T, which is $\boldsymbol{V}^T = \{\boldsymbol{V}_1^T, \boldsymbol{V}_2^T, \cdots, \boldsymbol{V}_K^T\}$ for K cameras. A video sequence for one camera is a subset, $\boldsymbol{V}_k^T \subset \boldsymbol{V}^T$. Each video sequence has F images, thus $\boldsymbol{V}_k^T = \{I_1, I_2, \cdots, I_F\}$. We use the two-stage DPM detector to find a set of detections $\boldsymbol{D}^T = \{\boldsymbol{D}_1^T, \boldsymbol{D}_2^T, \cdots, \boldsymbol{D}_K^T\}$ for K cameras in the time interval T. Furthermore, the set of detections in camera k over time T has a length of N and is $\boldsymbol{D}_k^T = \{d_1, d_2, \cdots, d_N\}$. Each detection is $d_n = [t, x_1, y_1, x_2, y_2, s]$ where t is the time index/frame number, x_1 is the horizontal coordinate of the top left corner of the bounding box with respect to the top left corner of the input image, y_1 is the vertical coordinate of the top left corner, x_2 and y_2 are the bottom right corner of the bounding box, and s is a score.

4.2. Vehicle Tracking

Just like visual vehicle detection, the topic of visual vehicle tracking has received a lot of attention in scientific research. The challenge of tracking vehicles in six different perspectives over longer time periods is mainly difficult due to three things; sudden changes in capturing conditions, similar appearance of vehicles, and inter-vehicle occlusions. Despite these challenges, the visual vehicle tracking methods have reached an accuracy that allows for higher-level understanding of trajectories in a scene.

We use the online tracking method presented in [23] in a tracking-by-detection manner for each perspective in order to track vehicles between frames. It uses Markov decision processes (MDP) in combination with the widely used Tracking-Learning-Detection (TLD) tracker [11].

The tracker is originally designed for tracking pedestrians, for which reason, it is optimized for tracking vehicles in this study. The first change is the aspect ratio of the template, which is chosen based on typical vehicle aspect ratios in the annotations of the KITTI dataset [8] as shown in Fig. 4. Note that the aspect ratio of vehicles varies with the orientation at which they are observed. From this follows that vehicles observed in the side views will have a larger aspect ratio than vehicles observed in the rear and front views. We use an aspect ratio of 1.5, which is the mean of the annotated bounding box aspect ratios of the KITTI dataset. The second change is the state transition parameters of the MDP, which has been trained for vehicles. The MDP is trained on a sequence from the KITTI dataset [8] using available ground-truth annotations and detections computed by the DPM detector.

We find a set of associations of detections between frames $\boldsymbol{A}^T = \{\boldsymbol{A}_1^T, \boldsymbol{A}_2^T, \cdots, \boldsymbol{A}_K^T\}$ for K cameras in the time interval T. The kth set of associations has a length of M and is $\boldsymbol{A}_k^T = \{a_1^k, a_2^k, \cdots, a_M^k\}$. Each association is $a_m^k = [ID, d_n]$ where ID is a unique vehicle identification number.

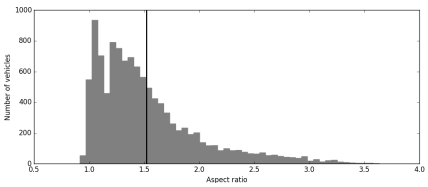


Figure 4. Histogram of annotated vehicle aspect ratios in the KITTI dataset [8]. The mean is shown with the vertical line at approximately 1.5.

4.3. Multi-Perspective Tracking

The final step of the multi-perspective trajectory generation is the connection of trajectories between cameras. Stationary setups have shown reliable performance, but in this study we have six perspectives on a moving platform, which makes the challenge of correctly associating trajectories non-trivial.

The trajectories are associated between perspectives, by assigning the same identification number to trajectories belonging to the same vehicle across perspectives. The association is done directly in the image planes, where stationary multi-perspective setups often perform the trajectory association in a common ground-plane. Since the camera views are known to overlap, predefined overlap regions are determined for each view denoted $\Omega^k = [\Omega_L^k, \Omega_R^k]$. Each trajectory is only evaluated once, in the first frame it appears. The bounding box of the new trajectory is firstly examined to be positioned in either the left or right overlapping region. Secondly, the corresponding adjacent view is examined for possible candidates to be associated with. Associated trajectories between cameras are described as $\boldsymbol{B}^T = \{\boldsymbol{B}_{k,k\pm1}^T\}$ for $k \in [1, 2, \cdots, K]$ in the time interval T, with K being the number of cameras. Note that k wraps around, such that k_1 and k_K are adjacent perspectives. Each set of associations between two cameras k and $k \pm 1$ is $\boldsymbol{B}_{k,k\pm1}^T = \{b_1, b_2, \cdots, b_L\}$ where $b_l = [a_m^k, a_{m'}^{k\pm1}]$ is the lth association.

$$b_l = \begin{cases} [a_m^k, a_{m'}^{k-1}] & \text{if } \Omega_R^{k-1} < a_{m'}^{k-1}(x_2) \text{ and } a_m^k(x_1) < \Omega_L^k \\ [a_m^k, a_{m'}^{k+1}] & \text{if } \Omega_R^k < a_m^k(x_2) \text{ and } a_{m'}^{k+1}(x_1) < \Omega_L^{k+1} \end{cases}$$

As an example, see Fig. 7(b), where the leftmost car just appeared, and is being associated with the rightmost car in Fig. 7(a). A similar association is made between Fig. 7(f) and Fig. 7(e). In the case with multiple possible matches in the adjacent view, a constraint is added, where an ID only can exists once in each view, or else the closest match is chosen.

This association scheme is seen to fail at high density scenes, or at late detections, when the vehicle has already passed the overlapping part of the image, resulting in two trajectories not being associated. The simple association method is found sufficient in free-flow highway scenarios.

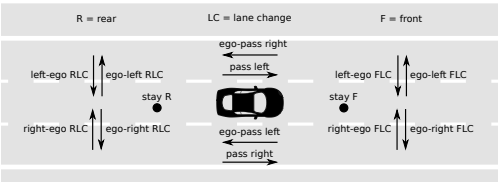


Figure 5. The 14 events detected in the trajectory analysis.

One advantage of using multiple views, is the ability to remove short-lived faulty trajectories, since the trajectories of interest are considered as long tracks in order to describe an event. All trajectories with a length less than a certain threshold measured in frames are removed. The threshold has been determined experimentally to 75 frames, corresponding to 5 seconds with 15 FPS, for the results presented in this work.

5. Trajectory Analysis

A map or a list of the dynamics and behaviors of surrounding vehicles is an integral part of understanding what is happening around the ego-vehicle, and why something is happening. In this section we present how the multi-perspective trajectories are transformed to a common framework and analyzed for events and certain behaviors. The system output is thus two-fold; a visualization of trajectories in the road surface enabling an in-depth analysis and a list with events that allows for fast interpretation.

5.1. Visualization of Trajectories

The visualization enables NDS to describe why events at certain time instances are happening. Combined with the actual video feeds, this is a powerful tool for studying on-road vehicle behaviors in a way that has not been presented previously.

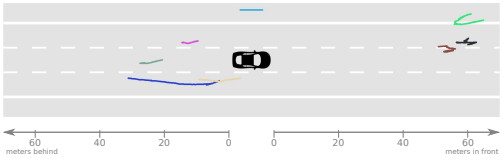
The multi-perspective trajectories are mapped to a common framework being the road surface. This is achieved by inverse perspective mapping (IPM) the front and rear perspectives, and using the middle of the bottom of the bounding box as a position of tracked vehicles. The trajectories are filtered using the average of the last n positions in order to achieve smooth tracks. The side views are used as discrete positions for rear left, rear right, front left, and front right. Furthermore, a simple lane estimator is used to show in which lane vehicles are positioned when they are on the side of the ego-vehicle. As the road might have a curve or slope, the IPM can not be expected to be accurate at larger distances. Vehicles are therefore tracked up to a distance of approximately 70 meters behind and in front of the ego-vehicle.

5.2. Data Reduction

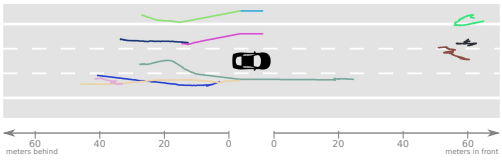
Visualizations are valuable for analyzing vehicle dynamics, but they contain a lot of data that are not easily interpretable. This problem can be solved by reducing the amount of information presented to the end-user. Furthermore, It allows for NDS to be automated.

The top-view trajectories are used to compute which events are occurring. We detect 14 different events as shown in Fig. 5. The method is currently limited to detecting lane changes in the front and rear perspectives, and only for adjacent lanes. Passing vehicles are found for all available lanes. For example, if a vehicle moves from a rear left to a front left position it is passing the ego-vehicle on the left. Likewise, if a vehicle moves from a front left to a rear left position the ego-vehicle is passing it on the right.

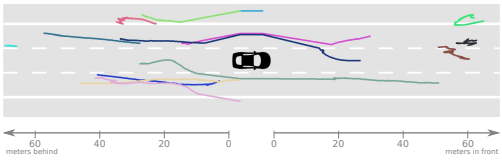
A combination of events can be grouped into semantics allowing for a higher-level understanding of vehicular maneuvers. For example, if a vehicle stays in front of the ego-vehicle within a certain distance over a time period, it can be concluded that the ego-vehicle is tailgating the vehicle in front. Another example is a vehicle that changes from ego-lane to left lane to pass the ego-vehicle on the left. This is defined as an overtake. If a passing vehicle changes lane to the ego-lane close to the front of the ego-vehicle, it is called a cut-in. A behavior that is potentially dangerous.



(a) Frame 100



(b) Frame 350



(c) Frame 600

Figure 6. Top-view trajectories of Seq2 at three time instances. As seen over time, the ego vehicle is being overtaken multiple times, where one vehicle furthermore makes a potential dangerous cut-in.

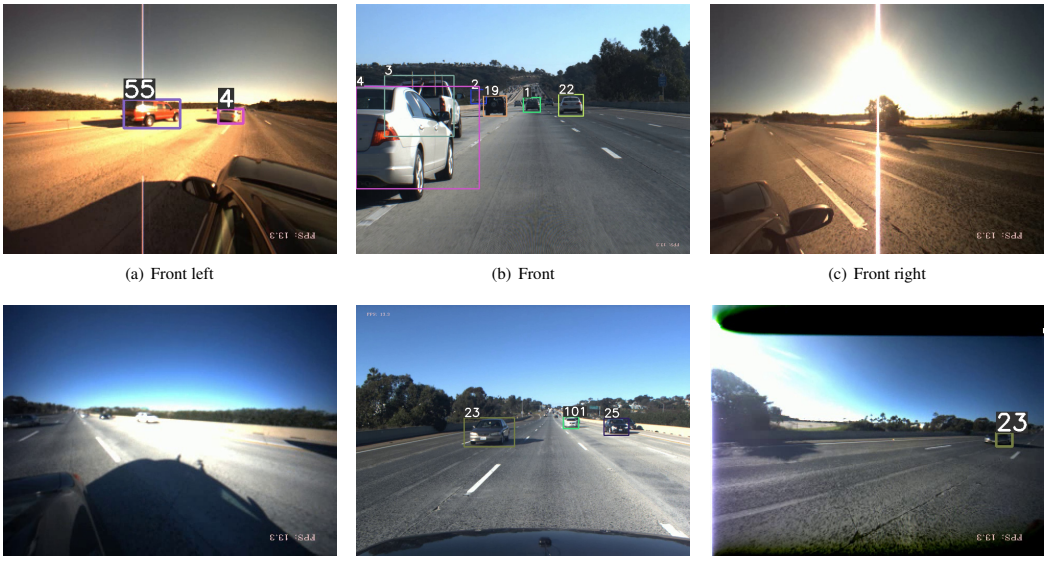
6. Experimental Evaluation

In this section we evaluate the performance of the system based on ten highway sequences ranging from 10 seconds to 40 seconds. The sequences are chosen from several hours of captured data in free-flow traffic, where interesting events are observed, to prove the potential of the system. These events include overtaking, tailgating, cut-ins, and cut-outs. In order to gain further insight in the performance, we show a detailed evaluation of one of the sequences.

It would be time consuming for NDS to analyze the events from six different perspectives. Our visualization allows for a top-down view of the scene, helping to get an overview of the different events. Fig. 6 shows the visualized trajectories at three time instances of a 40 seconds sequence (Seq2). In this way it is possible to see what is happening in the sequence over time. At the first time instance Fig. 6(a), the ego-vehicle has two receding vehicles in the rear right lane, one approaching vehicle in the rear ego-lane, one approaching vehicle in the rear left lane, one vehicle on the left side, and three vehicles in the lanes in front. At the second time instance Fig. 6(b), one of the vehicles has chosen to overtake on the right side of the ego-vehicle, which is probably caused by the vehicle overtaking on the left that has a lower velocity. Also, a new vehicle is approaching in the rear left lane. Note that this vehicle has a higher velocity than the vehicle currently overtaking on the left. This might be the reason why the very same vehicle at the third time instance Fig. 6(c), cuts into the ego-lane after overtaking the ego-vehicle on the left and starts to overtake the slower moving vehicle on the right. Fig. 7 displays the six perspectives at the time instance where the vehicle cuts into the ego-lane.

The list of events for Seq2 shown in Table 1 reduces the information further. With three detected left side passes and one right side pass, it can be concluded that the ego-vehicle drives slower than the surrounding traffic. However, as a vehicle stays in front of the ego-vehicle, it is likely that the ego-vehicle drives with a velocity similar to that vehicle. The combination of the visualization and the list of events is a powerful tool that allows for fast interpretation of behaviors occurring in a scene.

Table 1 summarizes the number of occurrences of each event for all ten sequences compared to the ground-truth obtained by manual inspection of each sequence. An overview of the ten sequences is shown in Fig. 8 along with all the trajectories from all ten sequences plotted in Fig. 8(k). This demonstrates the variety in the sequences of vehicles overtaking on both left and right, lane changes, and a few potential dangerous cut-ins. The system shows approximately the same tendencies as the ground-truth throughout all the ten sequences. This is also confirmed by the precision $TP/(TP + FP)$ and recall $TP/(TP + FN)$, where TP



(a) Front left (b) Front (c) Front right

(d) Rear left (e) Rear (f) Rear right

Figure 7. The six perspectives of Seq2 at frame 506. The multi-perspective tracked vehicles are shown by their latest detection in colored bounding boxes with corresponding identification number. Note some vehicles can be seen in multiple perspectives due to overlap, thus assigned the same identification number.

is true positives, FP is false positives, and FN is false negatives. The most frequent event is found to be vehicles passing the ego-vehicle on the left, while there was no one going from the ego-lane to the right-lane in front of the ego-vehicle. This indicates a passive driver, not forcing any of the cars in front to make a lane change. Also noteworthy is the event of a vehicle changing from ego-lane to left lane in front of the ego-vehicle, having a precision and recall of zero. This is partly explained by the false positives caused by the inaccuracy at far distances as seen in Fig.8(a). The inaccuracy is mainly caused by a road surface that is not completely flat or curved, which will make the IPM inaccurate, or the fact that only a small number of pixels are available the further away the vehicle is. The two false negatives seen in sequence three and seven respectively, may be caused by the filtering of trajectories, resulting in the trajectories coming up short, as the trajectories direction indicate a lane change, according to Fig.8(c) and Fig. 8(g).

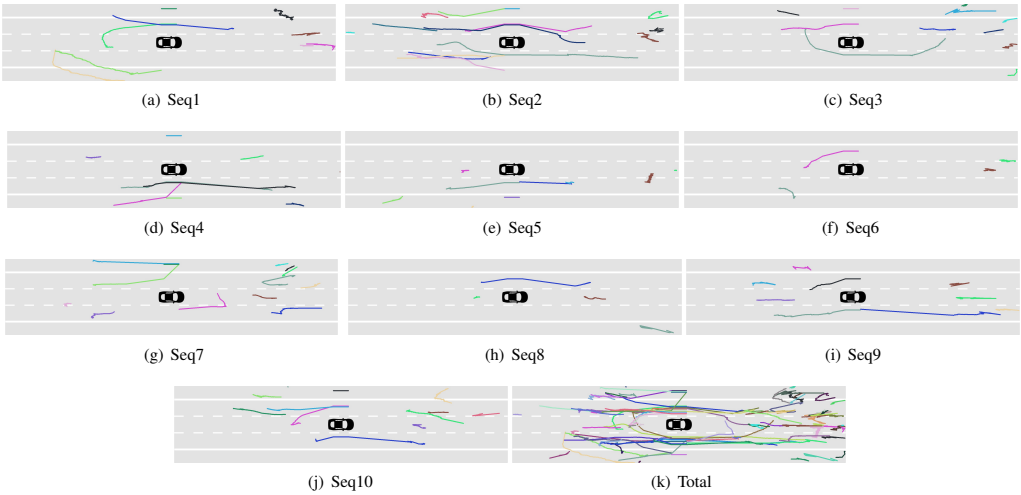
As seen in Fig. 8(k), the system is primarily tracking vehicles in the ego-lane and adjacent lanes. This is primarily due to frequent occlusions of vehicles in other lanes, but also the fact that they need a bigger distance to the ego-vehicle before appearing in the front and rear perspectives. The result is that vehicles in outer lanes have a higher probability of causing false negatives, which also reflects in the result for left passes in Table 1. Also, the association between views has difficulties if two vehicles pass on the same side simultaneously. Including more features than position may solve this problem, e.g. by using appearance cues. Furthermore, instead of using the overlap restriction, vehicles can be associated between views by allowing them to appear in other views within a certain time frame. This is however more a task of vehicle re-identification than overlap-association.

7. Concluding Remarks

This work developed a multi-perspective framework for analyzing on-road vehicle behavior in real-world highway data. The usage of multiple overlapping cameras proves useful for estimating persistent trajectories in full surrounding of the ego-vehicle. The multi-perspective framework successfully enables in-depth analysis despite the challenges introduced in the visible domain such as variances in point of view, glare from the sun, shadows of different sizes and shapes, and distortion (see Fig. 7 and Fig. 9 for examples), and is efficiently removing short-lived false trajectories. Furthermore, by using low-cost passive sensors in the visible spectrum the system allows for an interface that is easily understandable by humans, which is an important property in terms of human-computer-interaction

Table 1. Events detected by the system for all ten sequences compared to ground-truth (GT) [System/GT].

Event	**Seq1**	**Seq2**	**Seq3**	**Seq4**	**Seq5**	**Seq6**	**Seq7**	**Seq8**	**Seq9**	**Seq10**	**Precision**	**Recall**
Stay front	2/1	0/1	1/1	1/0	0/1	1/1	1/1	1/1	1/1	1/1	0.78	0.78
Stay rear	0/0	0/0	0/0	0/0	1/1	0/0	0/0	1/1	1/1	0/0	1.00	1.00
Pass on left	3/4	3/4	1/2	1/1	0/0	1/1	3/3	1/1	0/0	3/5	1.00	0.76
Pass on right	0/0	1/1	1/1	0/0	1/1	0/0	0/0	0/0	0/0	1/1	1.00	1.00
Ego-pass on left	0/0	0/1	0/1	4/4	1/1	0/0	0/0	0/0	1/1	0/0	1.00	0.75
Ego-pass on right	0/0	0/0	1/1	0/0	0/0	0/0	0/0	0/0	1/1	0/0	1.00	1.00
In front, left to ego-lane	1/0	2/1	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1/1	0.50	1.00
In front, right to ego-lane	0/0	0/0	1/1	0/1	1/1	0/0	1/1	0/0	0/0	0/0	1.00	0.75
In front, ego-lane to left	1/0	0/0	0/1	0/0	0/0	0/0	0/1	0/0	0/0	0/0	0.00	0.00
In front, ego-lane to right	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1.00	1.00
In rear, left to ego-lane	0/0	0/0	1/1	0/0	0/0	0/0	0/0	0/0	0/1	0/0	1.00	0.50
In rear, right to ego-lane	1/1	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1.00	1.00
In rear, ego-lane to left	1/1	0/0	0/0	0/0	0/0	1/1	0/0	0/0	0/0	1/1	1.00	1.00
In rear, ego-lane to right	0/0	1/1	0/1	0/0	0/0	0/0	1/0	0/0	0/0	0/0	0.50	0.50
Precision	0.7	0.88	1.0	0.83	1.0	1.0	0.83	1.0	1.0	1.0		
Recall	0.88	0.7	0.6	0.83	0.8	1.0	0.83	1.0	0.8	0.78		



(a) Seq1 (b) Seq2 (c) Seq3

(d) Seq4 (e) Seq5 (f) Seq6

(g) Seq7 (h) Seq8 (i) Seq9

(j) Seq10 (k) Total

Figure 8. Visualization of the ten sequences along with all the trajectories in total. Evaluated in Table 1.

(HCI). This makes the system an attractive addition to the sensor suite of intelligent vehicles.

The potential of the system is not limited to highway driving. More complex scenarios are a logical next step for example in urban areas as shown in Fig 9. In this specific scenario, the vehicle is stopped at an intersection with vehicles coming from the front right, and going through multiple perspectives, before disappearing in the rear left perspective. This is only one scenario among many. Applications able to model scenes by utilizing the surround view allow for sophisticated understanding of events and behavior. The obtained information can be used for both NDS and ADAS, ultimately answering questions such as: Why did this vehicle make a cut-in? Is it safe to make a left turn now?

A more comprehensive study of semantics from the detected events would include classification of e.g. safe and aggressive lane changes. Thus, a movement towards understanding high-risk semantics that need the attention of the driver or the ADAS. Also, by using a data-driven learning approach instead of the heuristic rule-based event classification, it will be possible to model typical trajectories allowing for future predictions of dynamics and behaviors in the scene.

(a) Front left (b) Front (c) Front right

(d) Rear left (e) Rear (f) Rear right

Figure 9. Six perspectives at an intersection in an urban scenario.

Acknowledgment

The authors would like to thank their colleagues at the Laboratory for Intelligent and Safe Automobile (LISA), University of California, San Diego, for assisting with the data gathering and their invaluable discussions and comments.

References

[1] Multi-Camera Object Tracking (MCT) Challenge. [Online]. Available: http://mct.idealtest.org.

[2] N. Anjum and A. Cavallaro. Trajectory Association and Fusion across Partially Overlapping Cameras. In *IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009*.

[3] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti, and P. Versari. 360 Detection and tracking algorithm of both pedestrian and vehicle using fisheye images. In *IEEE Intelligent Vehicles Symposium, 2015*.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 2010.

[5] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009*.

[6] T. Gandhi and M. Trivedi. Video Based Surround Vehicle Detection, Classification and Logging from Moving Platforms: Issues and Approaches. In *IEEE Intelligent Vehicles Symposium, 2007*.

[7] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3D Traffic Scene Understanding From Movable Platforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5), 2014.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[9] T. Gindele, S. Brechtel, and R. Dillmann. A Probabilistic Model for Estimating Driver Behaviors and Vehicle Trajectories in Traffic Environments. In *IEEE Conference on Intelligent Transportation Systems, 2010*.

[10] T. Huang and S. Russell. Object Identification in a Bayesian Context. In *IJCAI*, volume 97, 1997.

[11] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7), 2012.

[12] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel. Object-Oriented Bayesian Networks for Detection of Lane Change Maneuvers. *IEEE Intelligent Transportation Systems Magazine*, 4(3), 2012.

[13] M. S. Kristoffersen, J. V. Dueholm, R. Gade, and T. B. Moeslund. Pedestrian Counting with Occlusion Handling Using Stereo Thermal Cameras. *Sensors*, 16(1):62, 2016.

[14] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv:1504.01942 [cs]*, 2015.

[15] M. J. Mirza and N. Anjum. Association of moving objects across visual sensor networks. *Journal of Multimedia*, 7(1), 2012.

[16] B. T. Morris and M. M. Trivedi. Unsupervised Learning of Motion Patterns of Rear Surrounding Vehicles. In *IEEE International Conference on Vehicular Electronics and Safety, 2009*.

[17] R. K. Satzoda and M. M. Trivedi. Drive Analysis Using Vehicle Dynamics and Vision-Based Lane Semantics. *IEEE Transactions on Intelligent Transportation Systems*, 16(1), 2015.

[18] S. Sivaraman, B. Morris, and M. Trivedi. Learning Multi-Lane Trajectories using Vehicle-Based Vision. In *IEEE International Conference on Computer Vision Workshops, 2011*.

[19] S. Sivaraman and M. Trivedi. Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4), 2013.

[20] C. Wang, Y. Fang, H. Zhao, C. Guo, S. Mita, and H. Zha. Probabilistic Inference for Occluded and Multiview On-road Vehicle Detection. *IEEE Transactions on Intelligent Transportation Systems*, 17(1), 2016.

[21] G. Weidl, A. Madsen, D. Kasper, and G. Breuel. Optimizing Bayesian Networks for Recognition of Driving Maneuvers to Meet the Automotive Requirements. In *IEEE International Symposium on Intelligent Control, 2014*.

[22] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. DETRAC: A New Benchmark and Protocol for Multi-Object Tracking. *CoRR*, abs/1511.04136, 2015.

[23] Y. Xiang, A. Alahi, and S. Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. In *IEEE International Conference on Computer Vision, 2015*.

[24] H. Zhang, A. Geiger, and R. Urtasun. Understanding High-Level Semantics by Modeling Traffic Patterns. In *International Conference on Computer Vision, 2013*.

Paper B

Trajectories and Behaviors of Surrounding Vehicles Using Panoramic Camera Arrays

Jacob V. Dueholm, Miklas S. Kristoffersen, Ravi K. Satzoda,
Thomas B. Moeslund, Mohan M. Trivedi

The paper has been submitted for
IEEE Transactions on Intelligent Vehicles, 2016.

Trajectories and Behaviors of Surrounding Vehicles Using Panoramic Camera Arrays

Jacob V. Dueholm, Miklas S. Kristoffersen, Ravi K. Satzoda, Thomas B. Moeslund, and Mohan M. Trivedi

***Abstract*—Vision-based research for intelligent vehicles have traditionally focused on specific regions around a vehicle, such as a front looking camera for e.g. lane estimation. Traffic scenes are complex and vital information could be lost in unobserved regions. This paper proposes a framework that uses four visual sensors for a full surround view of a vehicle in order to achieve an understanding of surrounding vehicle behaviors. The framework will assist the analysis of naturalistic driving studies (NDSs) by automating the task of data reduction of the observed trajectories. To this end, trajectories are estimated using a vehicle detector together with a multi-perspective optimized tracker in each view. The trajectories are transformed to a common ground plane, where they are associated between perspectives and analyzed to reveal tendencies around the ego-vehicle. The system is tested on sequences from 2.5 hours of drive on U.S. highways. The multi-perspective tracker is tested in each view as well as for the ability to associate vehicles between views with a 92% recall score. A case study of vehicles approaching from the rear shows certain patterns in behavior that could potentially influence the ego-vehicle.**

***Index Terms*—behavior analysis, naturalistic driving studies, automatic data reduction, computer vision, multi-perspective, surround view, tracking.**

I. Introduction

AUTOMATIC identification and understanding of vehicle maneuvers is a useful tool for both advanced driver assistance systems (ADASs) and naturalistic driving studies (NDSs). NDSs aid in developing technologies that improve the safety of road users. Unlike controlled experimental studies, NDSs involve uncontrolled, yet naturalistic driving data that are analyzed for understanding driving styles and behaviors [1]. Most typical naturalistic driving data such as SHRP2 [1] include large volumes of data from multiple sensors such as visual data from multiple perspectives, in-vehicle sensor data, GPS and IMU (inertial motion unit) data, and active sensor data such as radars and lidars. Analysis of such data into specific events that could possibly lead to crashes or near-crashes is termed as data reduction. Such a process is usually conducted by human reductionists but there have been efforts to automate this process in more recent works such as [2], [3], [4]. There are a number of events and variables that are used as a reference to extract possible conflicting events during the trips from NDS data. While existing works on NDS have focused on detecting specific events such as lane changes, lane drifts etc., most existing works do not capture the spatiotemporal dynamics of the surrounding vehicles during the data reduction process, which are captured in the trajectories of the surrounding vehicles.

Trajectory analysis in the context of active safety and prediction using active sensors is well studied in the intelligent vehicles research community [5]. Surround sensing using multiple radars and lidars is commonly used to estimate trajectories of vehicles around the ego-vehicle [6]. However, cameras have become the cheapest sensing modality in recent times [5], and are being deployed in vehicles extensively to sense the surroundings. While front facing cameras are most extensively used in vehicles, surrounding visual inspection is less studied, albeit the increasing pervasiveness of cameras in vehicles. More specifically, trajectory analysis of the surrounding vehicles using multi-perspective visual data is not addressed as extensively as active sensor based trajectory analysis. Furthermore, data reduction in NDSs using the spatiotemporal dynamics of the surrounding vehicles by investigating their trajectories is also less addressed in existing literature.

Although active sensors are more effective in detecting physical parameters such as relative distances and velocities of surrounding vehicles, which are vital for trajectory analysis and estimation, visual sensor data provide a complementary modality while also providing a visual ground truth. This is especially critical in NDSs, which insist on visual inspection of the data to ascertain specific events leading to crashes and near-crashes. However, a manual inspection by human reductionists of the dynamics and trajectories of surrounding vehicles can be challenging especially when there are multiple perspectives. This is illustrated in a sample drive segment shown in Fig. 1, where two vehicles are making maneuvers around the ego-vehicle. Understanding such maneuvers is critical for developing automated techniques for driver behavior analysis using NDS data. Furthermore, this surround scene analysis can also be directly deployed for real-time driver assistance systems and inference engines and controllers in automated vehicles.

In this paper, we introduce surround vehicle trajectory analysis tools for NDS data analysis and reduction using multi-perspective visual data. This is particularly significant because cameras are being increasingly used in recent times to capture the 360° surroundings of the ego-vehicle. However, to the best of our knowledge, there are no works reported in literature that analyze surrounding visual data for trajectories.

We summarize the main contributions of the paper as follows: 1) We present a complete framework for estimating trajectories in full surround from four cameras mounted on a

J.V. Dueholm and M.S. Kristoffersen are with the Visual Analysis of People Lab, Aalborg University, Denmark, and the Laboratory for Intelligent and Safe Automobiles (LISA), University of California San Diego, USA.

T.B. Moeslund is with the Visual Analysis of People Lab, Aalborg University, Denmark.

R.K. Satzoda and M.M. Trivedi are with the Laboratory for Intelligent and Safe Automobiles (LISA), University of California San Diego, USA.

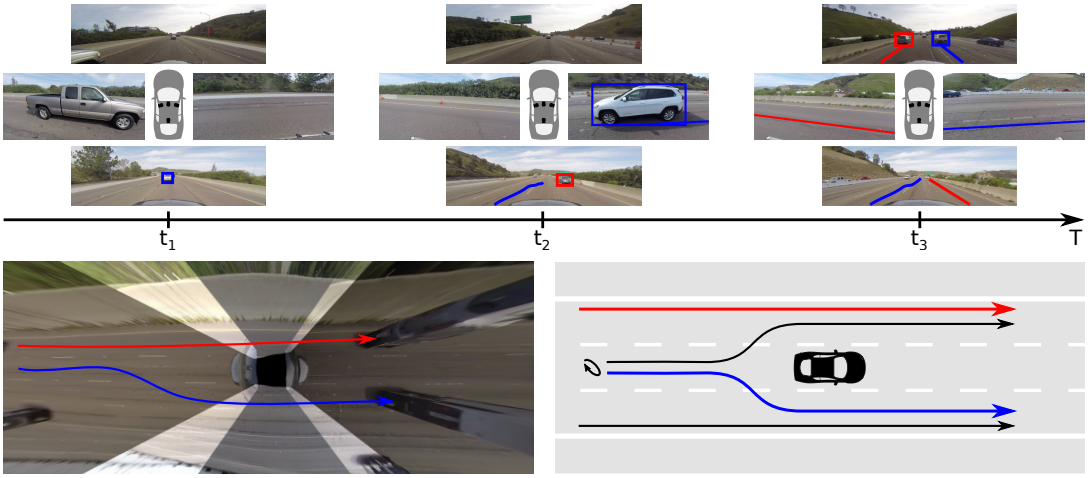


Fig. 1. Automatic driver behavior analysis using a multi-perspective camera setup include the objectives of vehicle detection, tracking, association between perspectives, and reduction of data.

TABLE I
RELATED WORK TOWARDS VISION-BASED FULL SURROUND VEHICLE BEHAVIOR ANALYSIS. ALL ON REAL-DATA USING A MOVING PLATFORM.

Reference	Sensors Used	Full Surround	Vehicle Detection	Vehicle Tracking	Trajectory Estimation	Behavior Analysis
Esparza et al. [7]	Eight fisheye cameras	✓				
Wang et al. [8]	Ladybug3 – 360° system	✓	✓			
Bertozzi et al. [9]	Four fisheye cameras	✓	✓	✓		
Proposed	Four GoPro HERO3+	✓	✓	✓	✓	✓

moving platform. 2) We modify the MDP tracker to be optimized for multi-perspective tracking by prolonging trajectories near the image boundary for better association between perspectives, and by reducing the number of false positives by rearranging the MDP state structure. 3) We present the estimated trajectories found around the ego-vehicle and analyze a simple scenario of vehicles approaching from behind by reducing a drive into events. Finally, we also discuss how the proposed techniques and their variants could be extended to analyze, in real-time, the dynamics of the surrounding vehicles for driver assistance using on-board computing systems.

The remaining paper is organized as follows: Section II describes related work. An overview of the framework used in this work is found in Section III, and in more detail in Section IV. The estimated trajectories found surrounding the vehicle are exposed in Section V. Section VI evaluates the proposed framework. Lastly, the concluding remarks are to be found in Section VII.

II. RELATED WORK

In this section we review recent studies that have contributed within the fields of obtaining surround view, trajectory estimation, and behavior analysis.

Comprehensive NDS have been conducted [10], [11], [1], but relies on manual labeling in order to reduce the raw sensor data to higher level inferences of behavior. Various studies have proposed automatic exploration of certain events within NDS [2], [12], [13]. High-level semantics of vehicle dynamics surrounding the ego-vehicle have previously been studied [5] with the purpose of identifying maneuvers such as overtakes and lane changes. Early examples [14] used simulated data, while recently, real data have been used in a front view of a moving platform [15], [16] on highways, classifying up to 27 maneuvers regarding lane changes. A fully unsupervised learning approach is seen applied to simulated trajectories in an intersection [17], in a frontal view [18], and in rear view in [19]. Each behavior is modelled by a HMM to be able to isolate abnormal trajectories. Trajectories have furthermore been used to infer traffic patterns in intersections using stereo vision [20], [21].

Vehicle trajectories are typically estimated by active sensors such as radars and lidars, passive camera sensors, or a fusion hereof. A full surround using four radars is achieved by [22], detecting parked vehicles at a parking lot. A full surround using four lidars is found in [23], [24]. The 3D point clouds give a natural framework to combine several active sensors by adding more points in the overlapping areas. Trajectories are estimated in a highway setting in both an online and offline manner. Both radars and lidars are used in [25] to recommend

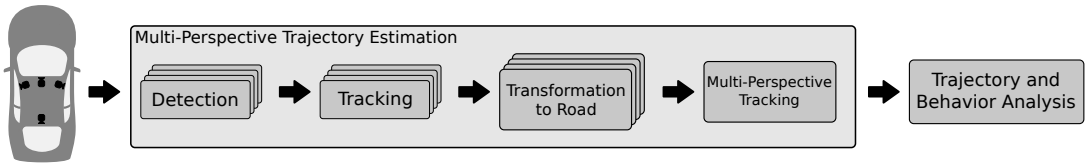


Fig. 2. The flow of the system from the input of four video sequences to the output of a multi-perspective trajectory and behavior analysis. The four synchronized cameras are placed as shown on the vehicle in the left side.

Fig. 3. Sample images captured from the synchronized multi-perspetive setup.

safe lane changes or lane merges based on surrounding vehicles. High-level fusion is seen in [26], fusing radar, lidar, and camera data based on existence probabilities in both partial and full overlapping regions. A similar system [27] was tested to recommend safe behaviors of the ego-vehicle for ADAS and autonomous driving applications in highway settings. Trajectory estimation from vision-based sensors can be divided into the two classic computer vision disciplines being detection and tracking of vehicles. These are well researched fields with public available databases with common benchmarks [5], [28], [29].

While most previous vision-based works use only one camera, it has been studied how to cover the full surroundings of a vehicle using multiple cameras. A summary of the studies working towards vision-based full surround vehicle behavior analysis is listed in Table I. Most surround view research consider the task of assisting a driver in avoiding obstacles e.g. when parking the vehicle. They do so by visualizing the close surroundings in a top-down view [30], [31], [32], [33] or by fully automating the process of parking [34], [35]. Also, multiple commercial systems have emerged with surround view parking assistance. These works are however focusing on the very near surroundings and are thus not included in Table I. In [7] eight cameras are used to reconstruct the surrounding scene in 3D. Two omni-directional cameras placed at the side mirrors are used in [36] to detect and track vehicles at the side and in front of the ego-vehicle. In [8] a full surround view is used to detect vehicles in a highway scenario with the omni-vision Ladybug3 system. Geometric models are learned for four dominant viewpoints to describe the configuration of vehicle parts and their spatial relations in probabilistic representations. A full surround detection and tracking method is proposed in [9]. Both vehicles and pedestrians are detected and tracked in a low velocity parking scenario using four fisheye cameras. The challenge of associating trajectories between perspectives is also studied in [37] for non-vehicle applications, where four cameras are used with partial overlap. Trajectories are extracted from each individual camera and projected to a common plane, where trajectories are associated. A similar approach is seen in [38], finding local trajectories, projecting to a common plane, and linking if the spatiotemporal features match.

III. System

The synchronized data used in this work are collected on U.S. highways in California. The vehicle used for data collection is equipped with four GoPro HERO3+ cameras. Furthermore, kinematics of the ego-vehicle are logged from the controller area network (CAN) bus. The four cameras are placed strategically around the vehicle, as shown in Fig. 2, in order to achieve a full surround view with slight overlaps between perspectives as seen in Fig. 3. The number of cameras is a trade-off between the amount of overlap, cost, and the computations needed for processing. With four cameras it is possible to achieve a 360° field of view, while keeping the amount of data for processing at an acceptable level. The cameras are capturing with a resolution of 2704x1440 at 12 frames per second (FPS). The cameras are calibrated to obtain intrinsic parameters in order to undistort the input images.

A flow diagram of the system is shown in Fig. 2. Vehicle detection is performed for each of the four inputs of the cameras. The detections in each perspective are used by the vehicle tracker to associate detections between frames for each of the four perspectives. The positions of the tracked vehicles are transformed to the road surface, where the trajectories are connected between perspectives. Finally, the data are reduced to analyze the surrounding vehicle behaviors.

IV. Multi-Perspective Trajectory Estimation

In the following section we present the methods designed for estimating trajectories of present vehicles surrounding the ego-vehicle using four different visual perspectives.

A. Vehicle Detection

Visual vehicle detection is a well researched topic that has seen recent scientific progress in both accuracy and computational speed, but is not yet considered a solved problem. Using multiple perspectives sets high demands for detecting vehicles at changing viewpoints, while the different perspectives will vary in lighting, shadows, and sun glare. An example of these challenges is shown in Fig. 3. As the ego-vehicle is a moving platform, detection methods utilizing background subtraction is less effective and a model based detector is preferred. Detections for each perspective can either be found using the same general trained model, or by training a model specifically for each view [8]. In this study, the trained KITTI model is experimentally found to be sufficient in all four perspectives. In this study the discriminatively trained deformable part models (DPM) is used [39] with a pre-trained vehicle model on the KITTI dataset [20], [21]. The DPM detects objects at different scales using a feature pyramid with the features being a variation of HOG features. Each detected object is assigned a confidence score based on the level and position of the pyramid. Overlapping detections are eliminated by non-maximum suppression.

Each detection in camera, k, is described using $d_n^k = [t, x_1, y_1, x_2, y_2, s]$ where t is the time index/frame number, x_1 is the horizontal coordinate of the top left corner of the bounding box with respect to the top left corner of the input image, y_1 is the vertical coordinate of the top left corner, x_2 and y_2 are the bottom right corner of the bounding box, and s is a confidence score.

B. Vehicle Tracking

Tracking vehicles in different perspectives over time is mainly challenging due to three things; sudden changes in capturing conditions, similar appearance of vehicles, and inter-vehicle occlusions. Despite these challenges, the visual vehicle tracking methods have reached an accuracy that motivates the estimation of multi-perspective trajectories. In this study we design a vehicle tracker with the purpose of associating the trajectories between multiple perspectives.

We seek to find a set of associations of detections between frames. Each association is $a_m^k = [ID, d_n]$ where ID is a unique vehicle identification number. To achieve this we use the online tracking-by-detection method presented in [40] for each perspective in order to track vehicles between frames. It uses Markov decision processes (MDP) in combination with the widely used Tracking-Learning-Detection (TLD) tracker [41].

The tracker is originally designed for tracking pedestrians, for which reason, it is optimized for tracking vehicles in this study. The first change is the aspect ratio of the template used for associating vehicles between frames, which is chosen based on typical vehicle aspect ratios in the annotations of the KITTI dataset [28] and of one annotated sequence collected in this work as shown in Fig. 4. Note that the aspect ratio of vehicles varies with the orientation at which they are observed. From this follows that vehicles observed in the side views will have a larger aspect ratio than vehicles observed in the rear or front view in a highway scenario, where the driving direction is mainly straight forward. Thus, optimally the aspect ratio should be optimized for each of the four perspectives. We have, however, experimentally found an aspect ratio of 1.5 to be sufficient for all four views, which is the mean of the annotated bounding boxes aspect ratios. The second change is the state transition parameters of the MDP shown in Fig. 5, which have been trained for vehicles. We have trained the MDP on a sequence captured on a U.S. highway with free-flow traffic, using ground-truth annotations and detections computed by the DPM detector.

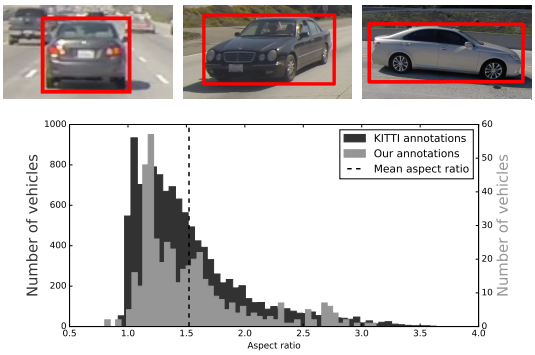


Fig. 4. Histogram of annotated vehicle aspect ratios in the KITTI dataset [28] shown with the dark color and histogram of annotated vehicle aspect ratios in our data shown with the light color. The means are almost equal and shown with the vertical line at approximately 1.5. Above are some examples of aspect ratios increasing from left to right.

The variation in appearance is less prominent for vehicles compared to pedestrians, and typical motion does not see abrupt changes as with pedestrians. This allows us to further constrain the creation of new trajectories as seen in Fig. 5. Thus, when a new vehicle is detected, it starts in an *Active* state, from where it can transition to a *Lost* state via action a_1 if it is determined to be a correct detection, or to an *Inactive* state via action a_2 if it determined to be a false detection. This stands in contrast to the original implementation where a new vehicle is able to transition directly to a *Tracked* state. The purpose of this re-routing in the MDP is to reduce the number of false positive trajectories caused by spurious detections that incorrectly transition from the *Active* state to the *Tracked* state instead of the *Inactive* state. The trade-off is that correctly tracked vehicles are slower at reaching the *Tracked* state, since they are kept in the *Lost* state until they have been observed β times. The action a_3 is thus not only the result of no association between a lost track and new detections, but also of a track that is too young to transition to the *Tracked* state.

In this study the tracking in each perspective has to take into account the purpose of linking trajectories between slightly overlapping perspectives. Thus, trajectories need to prolong as close to the image borders as possible. A situation that meets several challenges such as missing detections caused by truncation and severe appearance variations as the observation angle changes rapidly in the proximity of the ego-vehicle. An

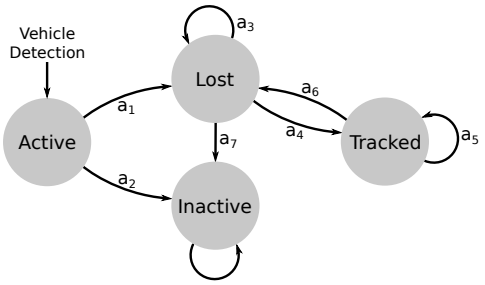


Fig. 5. The MDP states and actions designed for tracking vehicles. A new vehicle is not allowed to transition directly from the *Active* state to the *Tracked* state as in the original implementation [40].

example of these challenges is shown in Fig 3. Note from the figure that the truncated vehicles are located in regions where they might not be detected in any of the views due to truncation. One solution, that does not require a change of setup, is to design the vehicle detector for detecting truncated vehicles. A topic that has previously received attention, but remains an unsolved problem [5]. In this study we instead focus on extending the trajectories despite the missing detections. This is achieved by predicting bounding boxes for lost trackers. Let α be a set of associations that have the same ID and are sorted with increasing age of detections. Thus, the newest detection is at the first entry, α_1, and the first detection is at the last, α_M. An average change of the bounding box parameters is computed using:

$$\dot{\omega} = \frac{1}{L}\sum_{m=1}^{L}\frac{\omega|_{\alpha_m} - \omega|_{\alpha_{m+1}}}{t|_{\alpha_m} - t|_{\alpha_{m+1}}} \tag{1}$$

Where ω defines all four parameters of the bounding box, x, y, w, h, and L is the number of bounding boxes used in the estimation of the average change, $\dot{\omega}$. The bounding box prediction is then:

$$\hat{\omega}_{t|_{\alpha_1}+\tau} = \omega|_{\alpha_1} + \dot{\omega}\tau \tag{2}$$

Where τ is the time from the last available detection to the current time at which we want to predict the bounding box. The bounding box is used as a guess of where the vehicle has moved, which is tested using the iterative Lucas-Kanade method with pyramids to obtain the optical flow from previous detections of the vehicle (see [40] for a more detailed description). The Forward-Backward (FB) error of the optical flow is used as a measure of the stability of the prediction. If the median of the FB errors is below a certain threshold, T_1, the prediction is accepted as a valid vehicle detection. If that is not the case, an optical flow for the left and right half of the bounding boxes are investigated, since parts of the vehicle might have left the image. If the median FB error shows to be below a more strict threshold, T_2, for either the left or the right half, the prediction is assumed to be a true detection. The procedure is shown in Fig. 6.

The extension of the track is performed until τ gets larger than a predefined value or more than half of the bounding box is outside the image.

1: Predict bounding box using (1) and (2) for x, y, w and h
2: FBE $\leftarrow$ compute median of FB error
3: **if** FBE $< T_1$ **then**
4: Prediction is accepted
5: **else**
6: FBE_left $\leftarrow$ median of FB error for left half
7: FBE_right $\leftarrow$ median of FB error for right half
8: **if** FBE_left $< T_2$ or FBE_right $< T_2$ **then**
9: Prediction is accepted
10: **end if**
11: **end if**

Fig. 6. Algorithm for extension of trajectories.

Lastly, each track is modified to include normalized color histograms that are used for association between perspectives in Section IV-D. This allows us to base the association on both spatial, temporal and appearance information.

C. Transformation to Road Surface

It is difficult to analyze and associate trajectories in the image planes as they are not easily transfered to a common understanding of the surrounding road structure [42]. Furthermore, the four cameras have limited overlapping regions and are thus not simple to robustly calibrate extrinsically. A common solution is to use a top-down view, also known as a bird's eye view of the scene in order to achieve distances between surrounding vehicles and the ego-vehicle. The top-down view is achieved using Inverse Perspective Mapping (IPM), which is a transformation from image plane to road surface. The IPM estimates a homography matrix, $\boldsymbol{H}$, that maps a point in the image plane, $\boldsymbol{p}_{\text{image}}$, to a point in the road surface, $\boldsymbol{p}_{\text{road}}$, such that:

$$\boldsymbol{p}_{\text{road}} = \boldsymbol{H} \cdot \boldsymbol{p}_{\text{image}} \tag{3}$$

Note that the points are in homogeneous coordinates. A homography matrix for each perspective is estimated. However, the homography matrices are estimated such that they map to a global road surface and not four locally defined road surfaces. The ego-vehicle is placed in the origin, and the x-axis is the direction of the road such that the y-axis is orthogonal to the driving direction as seen in Fig. 7.

Since the mapping is for points in the road surface, it is needed to determine the location of surrounding vehicles on the road from their bounding boxes. This is solved by using the middle of the bottom of the bounding box:

$$\boldsymbol{p}_{\text{image}} = \begin{bmatrix} x + \frac{w}{2} & y + h & 1 \end{bmatrix}^{\mathrm{T}} \tag{4}$$

Where (x, y) is the top-left corner and (w, h) is the width and height of the bounding box. Note that the mapped position thus relies heavily on the performance of the vehicle detector to select bounding box parameters. Also, with varying observation angle the selected point will change position on the vehicle. As an example see Fig. 4, where the leftmost vehicle will be mapped at the rear whereas the rightmost vehicle will be mapped at the side. Furthermore, as a nature of IPM, the transformation is only accurate in level environments.

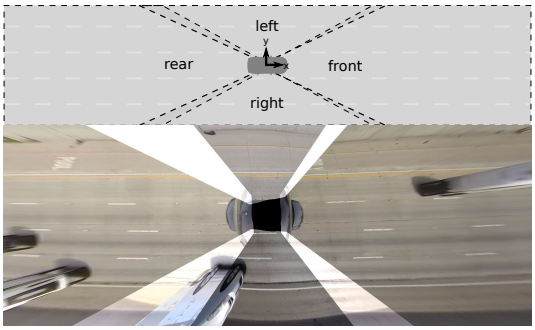


Fig. 7. Areas covered by the four cameras in the road surface. The bright areas in the bottom image are the regions in which two cameras overlap each other.

From this follows that any bumps or similar road irregularities will result in inaccurate and possibly erroneous mappings. In order to handle these inaccuracies, the mapped points are filtered using an average of the last α points in the trajectory. An alternative would be to use stereo or an active sensor such as lidar or radar to obtain the information needed for reconstructing the scene.

Since the cameras are overlapping, a vehicle might be tracked in two views simultaneously resulting in two points in the road surface. These points can be merged to one point using the spatial property that two points belonging to the same vehicle are placed close to each other in the road surface. This is essentially a form of vehicle association between perspectives, but the purpose is to make sure that each vehicle is only represented by one point in the road surface.

D. Multi-Perspective Vehicle Tracking

With the points in the road surface, the last task is to stitch the trajectories between perspectives. For this purpose, a Kalman filter is used for each trajectory. The filter simply models the (x, y) position and the $(\dot{x}, \dot{y})$ velocity of a vehicle in the road surface. Thus, the state vector, $\boldsymbol{X}_k$, is of the form:

$$\boldsymbol{X}_k = \begin{bmatrix} x_{\text{road}} & y_{\text{road}} & \dot{x}_{\text{road}} & \dot{y}_{\text{road}} \end{bmatrix}^{\text{T}} \tag{5}$$

The state transition matrix, $\boldsymbol{A}$, and the measurement transition matrix, $\boldsymbol{C}$, are defined as:

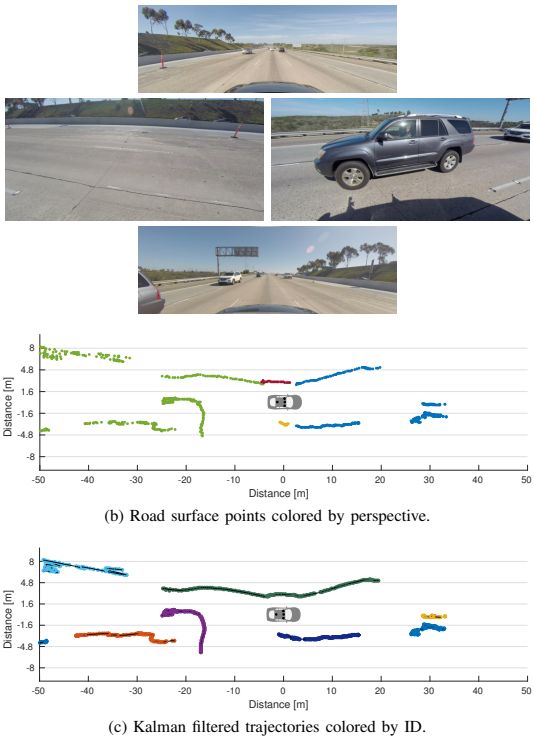
$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6}$$

The filters associate points in the road surface using spatiotemporal information. This allows us to track vehicles regardless of which perspectives they are tracked in. An example is shown in Fig. 8.

To further improve the robustness of the association of trajectories between perspectives, each time a new tracker is initialized the surrounding is investigated. If a lost tracker is present near to the new tracker, a similarity of the normalized color histograms for each tracker is used to determine if the two trackers belong to the same vehicle. Specifically, a χ^2 (chi-squared) distance, D, is computed [43], [44]:

$$D = \frac{1}{2} \sum_{b=1}^{B} \frac{(h_i(b) - h_j(b))^2}{h_i(b) + h_j(b)} \tag{7}$$

Where h_i and h_j denote the B-bin normalized histograms. If the χ^2 distance is below a threshold, T_3, the newly initialized tracker is deleted, and the old tracker is associated with the point.



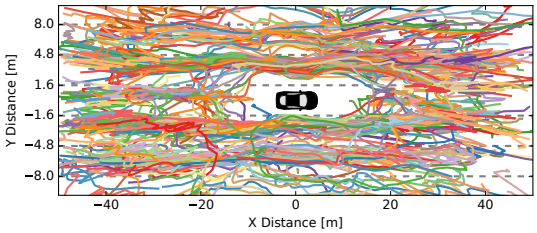
(b) Road surface points colored by perspective.

(c) Kalman filtered trajectories colored by ID.

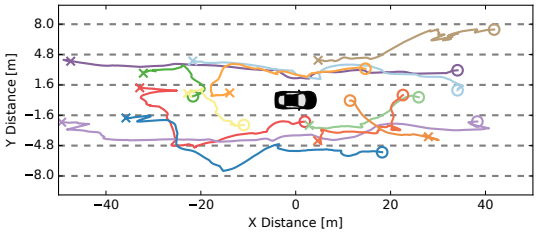
Fig. 8. Example of a sequence after 150 frames with the transformed points (b) and the Kalman filtered points (c).

V. Trajectory Analysis

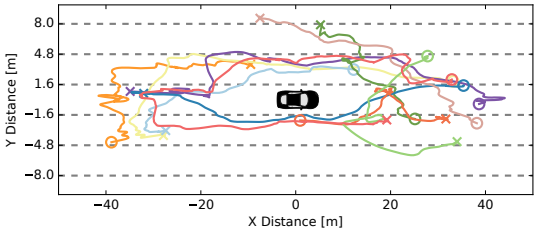
The following section investigates the estimated trajectories found around the ego-vehicle in a real-world highway setting using the proposed four camera framework. A total of 2.5 hours of driving on U.S. highways are gathered spread out on four days. The data are divided into 50 sequences to ignore scenarios with lane changes of the ego-vehicle and severe road curvatures which are unaccounted for. The sequences contain more than 15,000 frames with vehicles performing maneuvers around the ego-vehicle. The spatial properties of all the trajectories can be seen in Fig. 9a. This reveals the usage of the road and its lanes and a larger safety distance in the driving direction compared to the adjacent lanes.



(a) Surrounding trajectories estimated using the proposed framework.



(b) Simple trajectories consiting of staying in lane, or single lane changes.



(c) Abnormal trajectories.

Fig. 9. Estimated trajectories from real-world data. The trajectories are marked with a start position X and an end position O to indicate relative velocity to the ego-vehicle.

A vast variety of trajectories are found surrounding the ego-vehicle. A number of selected trajectories are shown in Fig. 9b. These contain what is considered simple maneuvers consisting of single lane changes or staying in lane. Variety is also seen within each type of maneuver e.g., lane changes at different distances to the ego-vehicle, different relative velocity of maneuvers, and trajectories of different length and different starting and end positions. These variations make analysis of the data challenging, since they complicate tasks such as grouping trajectories into general trajectory types.

Trajectories surrounding the ego-vehicle are not limited to simple maneuvers. More complex trajectories are shown in Fig. 9c, and include e.g. multiple lane changes and substantial changes in relative velocity. For example, one vehicle is observed to move all the way around the ego-vehicle (the same vehicle is also shown in Fig. 1). These trajectories occur less frequently in the collected data, and can be considered abnormal behaviors.

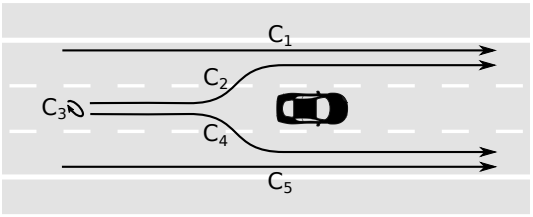


Fig. 10. Classes used for the classification of vehicles approaching from behind. C_1) Overtaking on the left. C_2) Left lane change followed by an overtaking on the left. C_3) Staying in ego-lane behind ego-vehicle. C_4) Right lane change followed by an overtaking on the right. C_5) Overtaking on the right.

A. Trajectory Analysis for NDS

The trajectories around the ego-vehicle can be used for multiple purposes in both active and passive applications. As an example, we here describe it in the case of NDS, classifying trajectories approaching from the rear. This is an interesting scenario since it is difficult for the ego-driver to be fully aware of the situation behind the ego-vehicle at all times. This analysis of typical behaviors of vehicles approaching the vehicle from behind is a necessary step before on-road prediction of how these vehicles will behave in the future, which might affect the ego-vehicle. However, as explained in the previous section, the set of trajectories include many different types of trajectories that are not easily separable. For this reason we study a simplified scenario with five classes of trajectories as previously seen in Fig. 10. All other types of trajectories are ignored in the analysis performed in this work.

The trajectories are classified using a data-driven learning approach, namely the Hidden Markov Model (HMM) [45]. The Markov model describes the spatiotemporal aspect of trajectories transitioning from state to state based on previous observations. The observations are vectors of features with four entries, being the Kalman filtered x and y positions and velocities in the road plane as seen in (5). The states themselves are not observable i.e. hidden, and not directly of interest for the classification task.

A HMM is fully described by its state transition probability matrix $\boldsymbol{A}$, emission probability matrix $\boldsymbol{B}$, and its initial state probability vector π as seen in (8). The matrices are of size $Q \times Q$, with Q being the number of hidden states used to model each trajectory.

$$\lambda_C = (\boldsymbol{A}_C, \boldsymbol{B}_C, \pi_C) \quad \text{with } C = \{C_1, C_2, C_3, C_4, C_5\} \quad (8)$$

The HMM parameters are trained for each of the five classes in a supervised manner with the manually annotated trajectories shown in Fig. 11. In this study all HMMs are trained with the same number of hidden states and initial states more likely to initialize in the hindmost states.

The likelihood of a new observed sequence O to originate from each class is calculated, and classified according to maximizing the likelihood as seen from (9) and (10) using the full length of the trajectories. The likelihood of the exact sequence

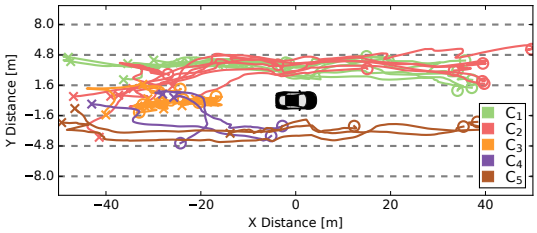


Fig. 11. Trajectories used to train the five classes.

to be observed is small, for which reason the logarithmic likelihood is used to avoid computational underflow.

$$LL_C = \log\ P(O|\lambda_C) \tag{9}$$

$$\hat{C} = \arg\max_C LL_C \tag{10}$$

Note how the classes each consist of a sequence of events, such as lane changes and passings. These are useful measures for the trajectory analysis, and can be used to get a quantitative overview of the behavior of vehicles behind the ego-vehicle. We extract five events: Passes on the left side of the ego-vehicle (C_1 and C_2), passes on the right side (C_4 and C_5), lane changes from ego-lane to left lane (C_2), lane changes to right lane (C_4), and tailgating (C_3). This list is not exhaustive and could be extended to include more types of events. However, by combining the classes and events with information of the velocity of the ego-vehicle from the CAN bus, the analysis enables automatic extraction of valuable information.

VI. Experimental Evaluation

In this section we present results showing the capability of multi-perspective trajectory estimation and analysis using several sequences from real highway data. The system is evaluated in three tests. The first test is a single-perspective tracking evaluation in four different perspectives compared with the performance of two additional trackers. This evaluation uses common metrics for multiple-object tracking. The second test evaluates the ability of the trackers to associate trajectories between perspectives in the road surface in 11 sequences. Finally, a NDS study analyzes the resulting trajectories of five types of maneuvers behind the ego-vehicle.

The two trackers used for comparison are the original online MDP tracker [40] and the offline Tracking-By-Detetion (TBD) [21] method, both publicly available. The only changes made to the original MDP are the template sizes, since it is originally made for pedestrian tracking, and the training is performed on the same single annotated training sequence as the modified MDP. The TBD method is based on the Hungarian algorithm, first associating detections into tracklets in an online approach, while Kalman filtering the bounding boxes. This is followed by a second Hungarian algorithm, to associate tracklets into trajectories. This second step is done in an offline batch manner, and the performance is thus not directly comparable to those of the two online trackers, but serves as a simple baseline. The method proposed in this paper uses the parameter values listed in Table II.

TABLE II
Parameter list.

Description	Parameter	Value
Number of observations needed to transition from *Lost* to *Tracked*.	β	3
Number of bounding boxes used in prediction of bounding box.	L	10
FB error threshold for whole, and left and right halfs of bounding box.	T_1,T_2	3,2
Number of points used for filtering.	α	5

(a) Before (b) During (c) After

Fig. 12. Example of an occlusion case, where the TBD makes a fragmentation and the two MDP methods track correctly. There is a one second offset between each of the samples.

A. Single-Perspective Tracking

The three different trackers are compared on a single sequence consisting of 400 frames (33 seconds). The sequence contains several inter-vehicle occlusions and vehicles moving between multiple perspectives. Each tracker is using the same detections with positive scores from the DPM detector, and is evaluated on all four individual perspectives. The trackers are evaluated using the CLEAR MOT metrics [46] and the definition of fragmentations (Frag.) and ID switches (IDS) from [47]. A fragmentation thereby occurs when a ground-truth (GT) trajectory is presented as two separate trajectories by the tracker. An ID switch occurs when a tracker ID switches from one ground-truth trajectory to another.

The ground-truth bounding boxes for the evaluation are obtained by manually annotating every tenth frame. Vehicles in the far distance are ignored by setting a constraint on the width of the detected vehicle to be at least 35 pixels, which corresponds to a distance of roughly 50 meters from the ego-vehicle. Furthermore, vehicles driving in the opposite direction on the other side of the crash barriers are also ignored.

The results of the single-perspective tracking comparison are shown in Table III. The MDP-Modified is shown to outperform the MDP for vehicle tracking, despite it being designed for multi-perspective association. Note, the difference in precision scores of the two, which proves the re-routing of the MDP states to effectively remove false positives at the cost of slightly lower recall scores. The gap in recall scores would be larger if the MDP-Modified did not prolong the tracks as close to the image border as possible.

The overall tendency shows the offline TBD to have better accuracies but more fragmentations. However, in the context of generating trajectories of surrounding vehicles, fragmentations cause the loss of all previous history data of that trajectory, and are thus severe mistakes. The MDP trackers both show superior in the number of fragmentations, since they are able to re-assign a trajectory, when being occluded in a short amount of time. There are multiple occlusion cases in the proximity of

TABLE III
SINGLE PERSPECTIVE TRACKING RESULTS. A COMPARISON BETWEEN THE THREE TRACKERS (TBD, MDP, AND MDP-MODIFIED), EVALUATED ON FOUR DIFFERENT PERSPECTIVES. TT IS THE NUMBER OF TRACKER TRAJECTORIES AND GTT IS THE NUMBER OF GROUND-TRUTH TRAJECTORIES.

Perspective	Tracker	TT/GTT	MOTA	MOTP	IDS	Frag	Recall	Precision
Front								
	TBD (offline)	12/10	0.55	0.75	0	2	0.70	0.82
	MDP	11/10	0.33	0.74	0	0	0.74	0.65
	MDP-Modified	11/10	0.38	0.75	0	0	0.72	0.68
Rear								
	TBD (offline)	13/8	0.79	0.76	0	4	0.79	0.99
	MDP	10/8	0.70	0.76	1	1	0.83	0.87
	MDP-Modified	9/8	0.79	0.76	0	1	0.83	0.95
Left								
	TBD (offline)	8/4	0.67	0.81	0	1	0.67	0.96
	MDP	5/4	0.64	0.81	0	0	0.76	0.86
	MDP-Modified	3/4	0.67	0.79	0	0	0.73	0.92
Right								
	TBD (offline)	4/3	0.73	0.79	0	1	0.82	1.00
	MDP	2/3	0.45	0.80	0	0	0.45	1.00
	MDP-Modified	2/3	0.55	0.83	0	0	0.55	1.00
Total								
	TBD (offline)	37/25	0.69	0.78	0	8	0.74	0.94
	MDP	28/25	0.53	0.78	1	1	0.70	0.84
	MDP-Modified	25/25	0.60	0.78	0	1	0.71	0.89

TABLE IV
ASSOCIATIONS BETWEEN PERSPECTIVES BY THE SYSTEM FOR 11 SEQUENCES COMPARED TO GROUND-TRUTH (GT) [SYSTEM/GT].

	Seq1	Seq2	Seq3	Seq4	Seq5	Seq6	Seq7	Seq8	Seq9	Seq10	Seq11	Total	Recall
TBD (offline)	11/13	4/4	5/6	3/4	4/5	8/8	8/10	15/16	4/4	5/6	6/6	73/82	0.90
MDP	12/13	3/4	6/6	4/4	5/5	7/8	8/10	14/16	4/4	6/6	5/6	74/82	0.91
MDP-Modified	13/13	4/4	5/6	4/4	4/5	7/8	9/10	14/16	4/4	6/6	5/6	75/82	0.92

the ego-vehicle in the sequence evaluated. One occurs in the left perspective as shown in Fig. 12, where the TBD makes a fragmentation, while both MDP methods correctly track the vehicle.

B. Multi-Perspective Association

In this section we evaluate the ability of our multi-perspective tracker to associate trajectories between perspectives. The test is based on manually annotated associations between perspectives in 11 sequences, which are used to inspect the performance of each of the three tracking methods. The three trackers use the same transformation to road surface and Kalman filter parameters. The evaluation is limited to five lanes. Thus, only vehicles within two adjacent lanes are taken into account. Note that the associations often come in pairs, i.e. a situation where the ego-vehicle is being overtaken, where the overtaking car is first seen in the rear perspective, followed by the left perspective, and finally the front perspective, which summarizes to two associations between perspectives.

The results are listed in Table IV and three examples are shown in Fig. 13. Note that precision scores are not included as no false positives occurred. The modified MDP achieves the most associations with a success rate of 92%. Generally, the three trackers score approximately equally, which might be a result of the Kalman filter having a high impact in the task of associating between views.

The trackers are observed to fail if a vehicle is truncated in two perspectives and moves with a low relative velocity causing the tracker to deactivate because of missing assignments of points. Also, if a vehicle moves with a very high relative velocity it is only present for a limited number of frames in the side view, and might not be tracked at all in that view. This situation relies on the Kalman filter to predict the movement from rear to front, which can be troublesome.

Due to the novelty of multi-perspective tracking around a vehicle, no direct comparison can be made. A somewhat similar approach is applied in tracking basketball players with the use of four overlapping cameras [37]. Here an average recall of 85% over a 40 seconds sequence is achieved, showing generally high success rates in association for both applications.

C. NDS Analysis of a Sample Drive

The estimated trajectories are analyzed to reveal overall tendencies of surrounding road users in a highway setting. Selected trajectory types are classified and further divided into events to describe the drive in a data reduced manner. Note the drive is divided into sequences with manually selected trajectories of the types shown in Fig. 10 and thereby do not give a complete picture of the distribution of events.

An evaluation is made using recall and precision to validate the correctness of each classified trajectory. The results are

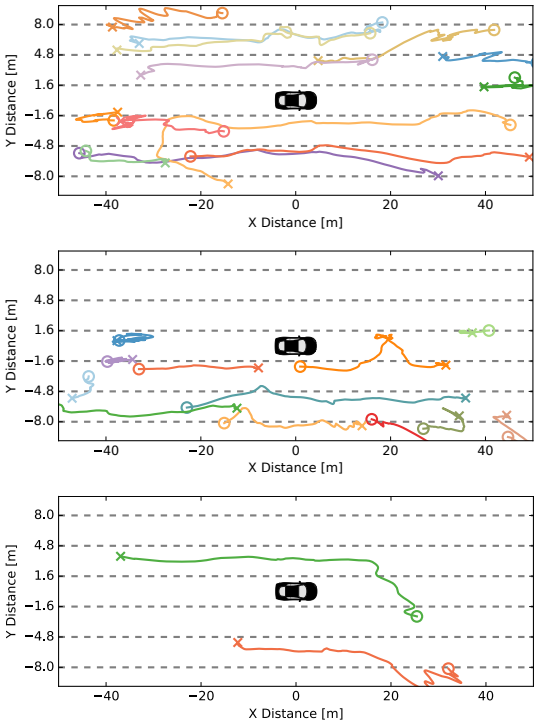


Fig. 13. Visualization of the resulting trajectories in three sequences.

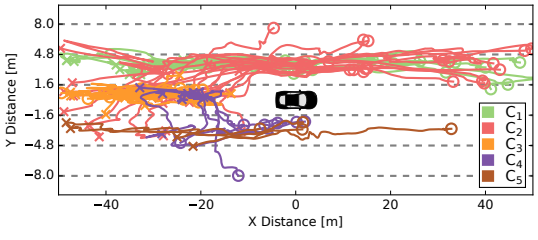


Fig. 14. Resulting trajectories used to test the five classes: C_1 in green, C_2 in red, C_3 in orange, C_4 in purple, and C_5 in brown.

shown in Table V for three hidden states in each HMM. The experiment was evaluated with varying number of hidden states with similar results. The classification percentages is in general found to be high with occasional misclassifications. Especially C_2 is found to be dominant with a high amount of false positives, which is caused by the overlap between the adjacent classes as seen from Fig. 11. Some trajectories are found to be more common than others as expected, but a decent number of overtakings on the right is found, which is considered as bad driving behavior.

The classified trajectories are used for the automatic data reduction, reducing the sample drive into a number of events e.g. a trajectory of C_2 is divided into two events, being a

TABLE V
CLASSIFICATION SCORE FOR THE HMMS ON A 2.5 HOUR DRIVE WITH SELECTED SEQUENCES.

Class	TP/GT	Recall	FP	Precision
C_1	17/21	81%	0	100%
C_2	16/16	100%	5	76%
C_3	18/19	95%	0	100%
C_4	6/6	100%	1	86%
C_5	4/5	80%	0	100%
Total	61/67	91%	6	91%

TABLE VI
TRAJECTORY ANALYSIS RESULTS.

Description	Value
Passes - left/right	33/10
Lane changes - left/right	16/6
Tailgating	18
Average velocity of ego-vehicle	102kph
Average velocity of C_1/ego	115kph / 104kph
Average velocity of C_2/ego	113kph / 99kph
Average velocity of C_3/ego	105kph / 104kph
Average velocity of C_4/ego	109kph / 106kph
Average velocity of C_5/ego	109kph / 97kph

left lane change followed by an overtaking on the left side of the ego-vehicle. The events used to describe the drive used in this work can be summarized as shown in Table VI, along with the average velocity of the classified trajectories with corresponding ego-vehicle velocity.

Looking at the estimated velocities also give rise to interesting findings. The ego-vehicle is found to be overtaken several times, although having an average velocity of 102 km/h, being just within the speed limit of 105 km/h. The ego-vehicle is even overtaken when slightly exceeding the speed limit as seen for C_4. The overtakes of C_2 and C_5 seems to be triggered by a slightly lower velocity of the ego-vehicle. Overtaking vehicles velocities are found to be close to the velocity of adjacent lanes going straight. Even though overtaking on the right is seen as bad behavior, the average velocity of the overtaking vehicle is found to less than overtaking on the left. Lastly, the trajectories of type C_3 is close to the velocity of the ego-vehicle as one would expect.

Even though the evaluations presented above are restricted to daytime highway conditions, we intend to extend the scenarios in which the system can be used to include nighttime driving [48] and urban environments [21].

VII. CONCLUDING REMARKS

This research is particularly focused on introducing a framework for estimating trajectories of surrounding vehicles in a highway setting using four cameras with slightly overlapping views to be used for data reduction for NDS.

The framework consist of the DPM detector and a modified version of the MDP tracker optimized for multi-perspective association of vehicles. The proposed method correctly associates 92% tracks between perspectives while also scoring

higher than the original MDP tracker in the single-perspective tracking evaluation. The framework is applied on sequences of a test drive showing the variety of trajectories found surrounding the ego-vehicle. Five types of trajectories are analyzed in the application of NDS data reduction. Trajectories are classified using trained HMMs, reducing the surrounding vehicle trajectories into events as lane changes and overtakes.

As presented, the system proves useful for NDSs as a powerful tool for understanding surrounding driver behaviors. However, the proposed framework is not limited to NDSs, but can be extended to ADASs. A possible ADAS is to predict surrounding vehicle trajectories in order to warn the driver of abnormal behaviors or of vehicles being on collision course. ADASs are subject to strict real-time requirements. Even though computation time has not been a priority in the choice of detector and tracker used in this work, the real-time capabilities are shortly examined. The DPM detector is found to be the bottleneck and is replaced with an off-the-shelf SubCat detector [49], detecting vehicles at approximately 1 Hz for each perspective without any loss in accuracy. Further optimization can be made by e.g. downsampling the 2704×1440 images, specifying regions of interest, and limiting the number of model orientations used for detecting in each perspective. To equip such a system in a vehicle would require it to run on an embedded platform as shown in [50]. To summarize, though several steps need to be taken, the framework allows for optimization of the individual modules potentially enabling it to emerge as a real-time surround trajectory and behavior tool for ADAS.

APPENDIX

Datasets will be available for evaluation and benchmarking in the public domain at Vision for Intelligent Vehicles and Applications[1] (VIVA) 2016. A few sequences of the dataset are added as a supplementary material for the review process.

ACKNOWLEDGMENT

The authors would like to thank their colleagues at the Laboratory for Intelligent and Safe Automobile (LISA) for assisting with the data gathering and their invaluable discussions and comments.

REFERENCES

[1] L. N. Boyle, S. Hallmark, J. D. Lee, D. V. McGehee, D. M. Neyens, and N. J. Ward, "Integration of Analysis Methods and Development of Analysis Plan," Transportation Research Board of the National Academies, Tech. Rep., 2012.

[2] R. K. Satzoda and M. M. Trivedi, "Drive Analysis Using Vehicle Dynamics and Vision-Based Lane Semantics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, 2015.

[3] S. Martin, E. Ohn-Bar, and M. M. Trivedi, "Automatic Critical Event Extraction and Semantic Interpretation by Looking-Inside," in *IEEE Conference on Intelligent Transportation Systems*, 2015.

[4] K. Takeda, J. H. L. Hansen, P. Boyraz, L. Malta, C. Miyajima, and H. Abut, "International Large-Scale Vehicle Corpora for Research on Driver Behavior on the Road," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, 2011.

[1] http://iv2016.org/workshops/#6

[5] S. Sivaraman and M. Trivedi, "Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, 2013.

[6] T. Luettel, M. Himmelsbach, and H. J. Wuensche, "Autonomous Ground Vehicles - Concepts and a Path to the Future," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, 2012.

[7] J. Esparza, M. Helmle, and B. Jähne, "Towards Surround Stereo Vision: Analysis of a New Surround View Camera Configuration for Driving Assistance Applications," in *IEEE Conference on Intelligent Transportation Systems*, 2014.

[8] C. Wang, Y. Fang, H. Zhao, C. Guo, S. Mita, and H. Zha, "Probabilistic Inference for Occluded and Multiview On-road Vehicle Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, 2016.

[9] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti, and P. Versari, "360 Detection and tracking algorithm of both pedestrian and vehicle using fisheye images," in *IEEE Intelligent Vehicles Symposium*, 2015.

[10] V. L. Neale, T. A. Dingus, S. G. Klauer, J. Sudweeks, and M. Goodman, "An Overview of the 100-Car Naturalistic Study and Findings," VTTI, Tech. Rep., 2002.

[11] T. A. Dingus, S. Klauer, V. Neale, A. Petersen, S. Lee, J. Sudweeks, M. Perez, J. Hankey, D. Ramsey, S. Gupta, *et al.*, "The 100-Car Naturalistic Driving Study, Phase II-Results of the 100-Car Field Experiment," Tech. Rep., 2006.

[12] O. Kumtepe, G. B. Akar, and E. Yuncu, "Driver aggressiveness detection via multisensory data fusion," *EURASIP Journal on Image and Video Processing*, 2016.

[13] M. P. Philipsen, M. B. Jensen, R. K. Satzoda, M. M. Trivedi, A. Møgelmose, and T. B. Moeslund, "Day and Night-Time Drive Analysis using Stereo Vision for Naturalistic Driving Studies," in *IEEE Intelligent Vehicles Symposium*, 2015.

[14] T. Gindele, S. Brechtel, and R. Dillmann, "A Probabilistic Model for Estimating Driver Behaviors and Vehicle Trajectories in Traffic Environments," in *IEEE Conference on Intelligent Transportation Systems*, 2010.

[15] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-Oriented Bayesian Networks for Detection of Lane Change Maneuvers," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 3, 2012.

[16] G. Weidl, A. Madsen, D. Kasper, and G. Breuel, "Optimizing Bayesian Networks for Recognition of Driving Maneuvers to Meet the Automotive Requirements," in *IEEE International Symposium on Intelligent Control*, 2014.

[17] B. T. Morris and M. M. Trivedi, "Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, 2011.

[18] S. Sivaraman, B. Morris, and M. Trivedi, "Learning Multi-Lane Trajectories using Vehicle-Based Vision," in *IEEE International Conference on Computer Vision Workshops*, 2011.

[19] B. Morris and M. Trivedi, "Unsupervised Learning of Motion Patterns of Rear Surrounding Vehicles," in *IEEE International Conference on Vehicular Electronics and Safety*, 2009.

[20] H. Zhang, A. Geiger, and R. Urtasun, "Understanding High-Level Semantics by Modeling Traffic Patterns," in *IEEE International Conference on Computer Vision*, 2013.

[21] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D Traffic Scene Understanding From Movable Platforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, 2014.

[22] R. Dubé, M. Hahn, M. Schütz, J. Dickmann, and D. Gingras, "Detection of Parked Vehicles from a Radar Based Occupancy Grid," in *IEEE Intelligent Vehicles Symposium*, 2014.

[23] H. Zhao, C. Wang, W. Yao, F. Davoine, J. Cui, and H. Zha, "Omni-directional detection and tracking of on-road vehicles using multiple horizontal laser scanners," in *IEEE Intelligent Vehicles Symposium*, 2012.

[24] H. Zhao, C. Wang, W. Yao, J. Cui, and H. Zha, "Vehicle Trajectory Collection Using On-Board Multi-Lidars for Driving Behavior Analysis," in *Knowledge Engineering and Management*. Springer, 2014.

[25] S. Sivaraman and M. M. Trivedi, "Dynamic Probabilistic Drivability Maps for Lane Change and Merge Driver Assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, 2014.

[26] M. Aeberhard, S. Paul, N. Kaempchen, and T. Bertram, "Object Existence Probability Fusion using Dempster-Shafer Theory in a High-Level Sensor Data Fusion Architecture," in *IEEE Intelligent Vehicles Symposium*, 2011.

[27] S. Noh, K. An, and W. Han, "High-Level Data Fusion Based Probabilistic Situation Assessment for Highly Automated Driving," in *IEEE Conference on Intelligent Transportation Systems*, 2015.

[28] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[29] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "DETRAC: A New Benchmark and Protocol for Multi-Object Tracking," *CoRR*, vol. abs/1511.04136, 2015.

[30] B. Zhang, V. Appia, I. Pekkucuksen, Y. Liu, A. U. Batur, P. Shastry, S. Liu, S. Sivasankaran, and K. Chitnis, "A Surround View Camera Solution for Embedded Systems," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

[31] S. Li and Y. Hai, "Easy Calibration of a Blind-Spot-Free Fisheye Camera System Using a Scene of a Parking Space," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, 2011.

[32] Y.-C. Liu, K.-Y. Lin, and Y.-S. Chen, "Bird's-eye view vision system for vehicle surrounding monitoring," in *Robot Vision: Second International Workshop*. Springer, 2008.

[33] T. Ehlgen, T. Pajdla, and D. Ammon, "Eliminating Blind Spots for Assisted Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 4, 2008.

[34] J. K. Suhr and H. G. Jung, "Sensor Fusion-Based Vacant Parking Slot Detection and Tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, 2014.

[35] C. Wang, H. Zhang, M. Yang, X. Wang, L. Ye, and C. Guo, "Automatic Parking Based on a Bird's Eye View Vision System," *Advances in Mechanical Engineering*, vol. 6, 2014.

[36] T. Gandhi and M. M. Trivedi, "Vehicle Surround Capture: Survey of Techniques and a Novel Omni-Video-Based Approach for Dynamic Panoramic Surround Maps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, 2006.

[37] M. J. Mirza and N. Anjum, "Association of moving objects across visual sensor networks," *Journal of Multimedia*, vol. 7, no. 1, 2012.

[38] N. Anjum and A. Cavallaro, "Trajectory Association and Fusion across Partially Overlapping Cameras," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2009.

[39] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, 2010.

[40] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track: Online Multi-Object Tracking by Decision Making," in *IEEE International Conference on Computer Vision*, 2015.

[41] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, 2012.

[42] M. S. Kristoffersen, J. V. Dueholm, R. K. Satzoda, M. M. Trivedi, A. Møgelmose, and T. B. Moeslund, "Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016.

[43] O. Pele and M. Werman, "The Quadratic-Chi Histogram Distance Family," in *Computer Vision – ECCV 2010*. Springer, 2010.

[44] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, 2002.

[45] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, 1989.

[46] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *Journal on Image and Video Processing*, 2008.

[47] Y. Li, C. Huang, and R. Nevatia, "Learning to Associate: HybridBoosted Multi-Target Tracker for Crowded Scene," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[48] R. K. Satzoda and M. M. Trivedi, "Looking at Vehicles in the Night: Detection & Dynamics of Rear Lights," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[49] E. Ohn-Bar and M. M. Trivedi, "Learning to Detect Vehicles by Clustering Appearance Patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, 2015.

[50] F. Lu, S. Lee, R. K. Satzoda, and M. M. Trivedi, "Embedded Computing Framework for Vision-Based Real-time Surround Threat Analysis and Driver Assistance," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016.

Jacob Velling Dueholm received the B.Sc. degree in Electronic Engineering with specialization in Signal Processing in 2014 from Aalborg University, Denmark. He is currently pursuing the M.Sc. degree in Vision, Graphics, and Interactive Systems at Aalborg University. He has been a Visiting Scholar with the LISA lab at the University of California San Diego in the Computer Vision and Robotics Research Laboratory. His research interests include computer vision and machine learning.

Miklas Strøm Kristoffersen received the B.Sc. degree in Electronic Engineering with specialization in Signal Processing in 2014 from Aalborg University, Denmark. He is currently pursuing the M.Sc. degree in Vision, Graphics, and Interactive Systems at Aalborg University. He has been a Visiting Fulbright Scholar with the LISA lab at the University of California San Diego in the Computer Vision and Robotics Research Laboratory. His research interests include computer vision and machine learning.

Ravi Kumar Satzoda received his B.Eng. (with First Class Honors), M. Eng. (by research) and Ph.D degrees from Nanyang Technological University (NTU), Singapore in 2004, 2007 and 2013 respectively. He is currently a Post Doctoral Fellow at University of California San Diego (UCSD), and associated with Laboratory for Intelligent and Safe Automobiles. His research interests include computer vision, embedded vision systems and intelligent vehicles.

Thomas Baltzer Moeslund received the M.Sc.E.E. and Ph.D. degrees from Aalborg University, Aalborg, Denmark, in 1996 and 2003, respectively. He is currently a Professor and the Head of the Visual Analysis of People Laboratory with Aalborg University. He has been involved in ten national and international research projects, as a Coordinator, Work Package leader, and Researcher. He is the author of more than 100 peer-reviewed papers. His research interests include all aspects of computer vision, with a special focus on automatic analysis of people. Prof. Moeslund has been a co-chair of four international workshops/tutorials and a member of the Program Committee for a number of conferences and workshops. He serves as an Associate Editor and as a member of the Editorial Board for four international journals. He received the Most Cited Paper Award in 2009, the Best IEEE Paper Award in 2010, and the Teacher of the Year Award in 2010.

Mohan Manubhai Trivedi received the B.E. (with honors) degree in electronics from Birla Institute of Technology and Science, Pilani, India, in 1974 and the M.S. and Ph.D. degrees in electrical engineering from Utah State University, Logan, UT, USA, in 1976 and 1979, respectively. He is currently a Professor of electrical and computer engineering. He has also established the Laboratory for Intelligent and Safe Automobiles, and Computer Vision and Robotics Research Laboratory, UCSD, where he and his team are currently pursuing research in machine and human perception, machine learning, intelligent transportation, driver assistance, active safety systems and Naturalistic Driving Study (NDS) analysis. He received the IEEE Intelligent Transportation Systems Society's highest honor, Outstanding Research Award in 2013.

Paper C

Vision for Intelligent Vehicles & Applications (VIVA): Multi-Perspective Vehicle and Trajectory Challenge

Jacob V. Dueholm, Miklas S. Kristoffersen, Ravi K. Satzoda,
Eshed Ohn-Bar, Thomas B. Moeslund, Mohan M. Trivedi

The paper is incomplete, but intended for submission to
IEEE Intelligent Transportation Systems Conference, 2016.

Vision for Intelligent Vehicles & Applications (VIVA): Multi-Perspective Vehicle and Trajectory Challenge

Jacob V. Dueholm[1,2], Miklas S. Kristoffersen[1,2], Ravi K. Satzoda[1], Eshed Ohn-Bar[1], Thomas B. Moeslund[2] and Mohan M. Trivedi[1]

***Abstract*— 200 words – Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis non mauris eros. Pellentesque in enim eget arcu tincidunt ultrices in ut est. In non justo sed nisl euismod rhoncus. Cras bibendum sapien a ligula pretium, quis varius ante iaculis. Sed blandit, tellus sed sagittis feugiat, enim magna laoreet tellus, vel accumsan dolor dui ac sapien. Aliquam velit justo, mollis non mauris egestas, mollis consequat lorem. Vestibulum lobortis id purus ac finibus.**

Morbi sit amet ligula enim. Donec in iaculis erat. Aenean bibendum maximus lorem, a vehicula nisi volutpat eget. Sed tempor sem quis diam scelerisque molestie. Praesent porta, est mattis faucibus vehicula, dui nisi posuere quam, nec cursus neque elit eget eros. Nam velit nibh, tristique at ligula congue, commodo molestie ipsum. Proin pharetra est a auctor iaculis. Mauris molestie nulla non nisi interdum pretium eget eu felis. Sed consequat mauris auctor, congue elit at, eleifend enim. Suspendisse sed nunc erat. Mauris enim magna, aliquet sit amet sem imperdiet, tincidunt auctor erat. Nam at justo et metus placerat placerat sed eget felis. Nullam nec enim mattis, maximus sapien nec, rhoncus enim. Aliquam vitae imperdiet sem. Morbi aliquam, ligula a feugiat fringilla, elit nunc rhoncus tortor, ut dapibus leo lorem ut.

I. Introduction

Detecting and tracking vehicles in full surround the ego-vehicle are a natural next step given the improved seen in monocular perspectives. This allows for the study of on-road behaviors [1], identifying behaviors [2], [3] and long-term prediction [4] for both active and passive safety applications. To this end, the research community has pushed forward the need for publicly available datasets and common benchmarks, as to strengthen the scientific methodology for development and evaluation of novel research.

Vision for Intelligent Vehicles & Applications (VIVA) [9] is a vision-based challenge set up to serve two major purposes. The first is to provide the research community with naturalistic driving data from looking-inside and looking-outside the vehicle, and thus to present the issues and challenges from real-world driving scenarios. The second purpose is to challenge the research community to highlight problems and deficiencies in current state-of-the-art approaches and simultaneously progress the development of future algorithms. Current challenges in VIVA include hands [10], faces [11], traffic signs [12], [13], traffic lights [14] and nighttime vehicles [15]. In this paper we introduce multi-perspective vehicle detection and tracking as a new part of the VIVA challenge, moving towards understanding trajectory based behaviors surrounding the ego-vehicle.

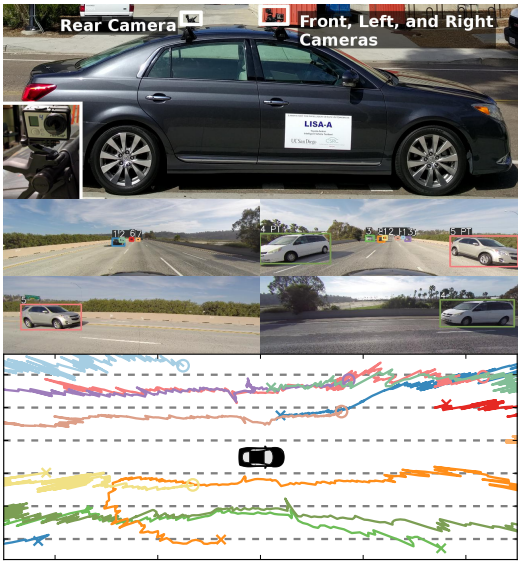


Fig. 1. Vehicle instrumented with four cameras (top), ground truth bounding boxes in the four perspectives (middle), and 3D ground truth trajectories (bottom).

Mention Active Sensors??

Delete trajectory column from table, or rename

The related datasets are summarized in Table I. Several benchmarks have emerged for vision-based vehicle **detection**. The Pascal VOC Challenge **ref??** is used to detect cars at various viewpoints in static images among 20 other object classes. The TME dataset [5] is continuous in a front faced perspective on Italian highways. Annotations are done automatically from lidar data. The PKU POSS dataset [6] used the omni-directional Ladybug3 to obtain full surround images which is divided into four images of equal size for vehicle detection at different viewpoints. Most notably, is the KITTI Vision Benchmark Suite [8] which comprises a large collection of on-road data, also from a single front faced perspective. KITTI furthermore annotated IDs which allows **Tracking**. Various work is found in both detecting and tracking of vehicles in traffic scenes seen from a surveillance perspective, as for the DETRAC [7] dataset. Trajectory

[1]Laboratory for Intelligent and Safe Automobiles, University of California, San Diego, USA

[2]Visual Analysis of People Laboratory, Aalborg University, Denmark

TABLE I

Dataset	Sensors	Detection Tracking Trajectory	Moving Platform	Full Surround	Environment
TME [5]	Two 1024×768 front faced color cameras at 20 Hz, IBEO 4 layer laser scanner	✓/✗/✗	✓	✗	Northern Italian Highways
PKU POSS [6]	Ladybug3 1 Hz	✓/✗/✗	✓	✓	Chinese Highways
DETRAC [7]		✓/✓/✗	✗		
NGSIM??		✓/✓/✓	✗		U.S. Highway??
KITTI [8]	lidar, front faced stereo camera 1392×512 at 10 Hz, GPS	✓/✓/✗	✓	✗	Germany, mid-size city, rural, highway
LISA Trajectory	Four GoPro with capturing 2704×1440 at 12/24 Hz	✓/✓/✓	✓	✓	U.S. Highways

databases as [16] estimated using both real and simulated data with the focus of clustering similar trajectories.

This vision based dataset consists of on-road full surround in high resolution at 12 FPS, suitable for both detection, tracking, and estimating trajectories around the ego-vehicle introduced as **Multi-Perspective 3D Tracking**. Contributions: 1) Full Surround 2) U.S. Highways, High resolution images at 12 FPS 3) Few fully annotated sequences 4) Propose metrics for full surround

II. Dataset and Challenges

A. Data Acquisition

The database is collected on U.S. Highways in southern California over 2.5 hours of driving. The drives are divided into sequences consisting of challenging behaviors found around the ego-vehicle including e.g. overtaking, cut-ins, and cut-outs. The data capturing vehicle is equipped with four GoPro HERO3+ achieving a full surround view with limited overlap as seen from Figure 1. Each camera is recording at a resolution of 2704×1440 at 12 Hz, and post-processed offline to synchronize and correct distortion. All four cameras are calibrated to a common road plane using homographies estimated from recordings at a parking lot, where the relative world positions of points used for the estimation are known.

Challenges??

CAN bus??

B. Ground Truth Annotation

The ground truth is obtained for each of the four perspectives by manually annotating bounding boxes in the format as seen in (2), where (x_1, y_1) denotes the top-left corner, and (x_2, y_2) denotes the lower-right corner.

$$[\textit{frame}, \textit{id}, \textit{occlusion}, \textit{truncation}, x_1, y_1, x_2, y_2] \quad (1)$$

Each vehicle is assigned an identification number, *id*, to evaluate tracking. Note the *id* is consistent between perspectives giving the option of multi-perspective tracking. The occlusion and truncation tags are both divided into three levels being *No*, *Partial*, and *Heavy*. Here, *No* equals 0%, *Partial* includes vehicles up 50%, while *Heavy* covers 50%+ for both occlusion and truncation. Three levels are chosen to simplify the annotation workload, while maintaining a certain division for analysis purposes. Annotation examples are shown in Fig. 2.

The bounding box annotations allow for evaluation of both detection and tracking. In order to evaluate 3D tracking, we use the homographies to transform each ground truth trajectory to the road plane. The middle of the bottom of the bounding box, $(x_1 + 0.5(x_2 - x_1), y_2)$, is used as vehicle position. The road plane annotations have four entries:

$$[\textit{frame}, \textit{id}, x, y] \quad (2)$$

Examples of ground truth trajectories in the road plane are shown in Fig. 3. Note that the annotations rely on the homography.

C. Evaluation Metrics

Explain how we handle multiple views?? Well-established metrics exist for both detection and single-perspective tracking. In **detection** the average precision (AP) is commonly used, calculated as the area under the precision-recall curve. For a ground truth bounding box to be matched with a detected bounding box, an overlap defined as the intersection over union is required to be at least 0.7 in this work.

The single-perspective multiple-object **tracking** is evaluated using the CLEAR MOT metrics (MOTA, MOTP) [17], together with metrics including fragmentations (Frag) and ID switches (IDS) [18], mostly tracked (MT), and mostly lost (ML) [19]. A ground-truth trajectory is counted as mostly tracked if it is associated more than 80% of the time by definition. Likewise is a ground-truth trajectory accounted as a ML if associated in less than 20% of the time. A fragmentation is added every time a ground-truth trajectory is split. An ID switch is added if a ground-truth trajectory is matched with another ID than the one that is currently associated.

The field of **multi-perspective 3D tracking** is less explored without any common metrics. The problem, however, is not so different from tracking in single perspectives. For this reason, we propose to use similar metrics in the road plane. Instead of association between ground truth

Fig. 2. Sample images from one of the sequences in the dataset with overlaid ground truth annotations at six different time instances. The bounding boxes are color coded by *id* and labled with *PO*, *HO*, *PT*, and *HT* denoting partial occlusion, heavy occlusion, partial truncation and heavy truncation, respectively.

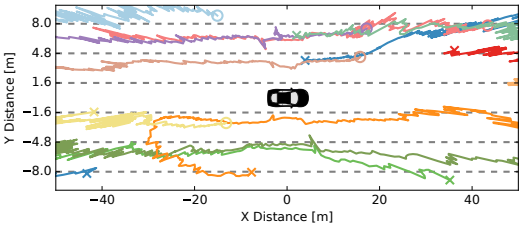


Fig. 3. Ground truth trajectories in the ground plane. The trajectories are marked with a start position X and an end position O to indicate relative velocity to the ego-vehicle.

bounding boxes and tracked bounding boxes, this will require association between points in the road plane. To this end, we use a weighted euclidean distance from ground truth trajectory points. This does however mean that the cost of matching a candidate to ground truth is not normalized (as the bounding box overlap definition), and thus MOTP is not well-defined ([20] suggests to normalize the distance with the matching threshold). The original definition of MOTP says that it is the average dissimilarity between all true positives and their corresponding ground truth targets. With bounding box overlap a score as close to 1.0 is ideal. Using euclidean distance changes the ideal score to be as close to zero as possible, as it is now defined as the average distance between true positives and their corresponding ground truth targets. To reduce the confusion, we refer to this score as multiple object tracking euclidean precision (MOTEP):

$$\text{MOTEP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{3}$$

Where c_t denotes the number of matches in frame t, and $d_{t,i}$ is the weighted euclidean distance between target i in frame t and the corresponding ground truth target. Another important aspect is the choice of matching threshold, which is traditionally chosen as either 0.5 or 0.7 bounding box overlap. A direct transfer to the road plane domain would be a static distance allowed from each ground truth trajectory point. However, as a nature of inverse perspective mapping small variations close to the ego-vehicle will not be as severe as small variations further away. We propose to make the matching criterion a function of the x-distance from the ego-vehicle. A target is matched to a ground truth target if it fulfills:

$$d_{t,i} < a|x| + b \tag{4}$$

where $|x|$ is the absolute x-coordinate of the ground truth target, a is the gradual increase in allowed distance, and b is the allowed distance at $|x| = 0$. Specifically, we use $a = 0.04$ and $b = 2$. It should however be noted that variations in the y-direction is more likely to cause erroneous matches. For this reason, we define the weighted euclidean distance as:

$$d_{t,i} = \sqrt{(gt_x - tr_x)^2 + 4\,(gt_y - tr_y)^2} \tag{5}$$

where $gt = [gt_x\ gt_y]^{\mathrm{T}}$ is the 2D ground truth position and $tr = [tr_x\ tr_y]^{\mathrm{T}}$ is the position of the target. Thus, distances in the y-direction have a double weight.

The metrics suggested above do however not encapsulate the importance of ID switches that happen between perspectives. For this reason, we intend to include trajectory similarity measures in the future, which could e.g. be longest common subsequence (LCSS).

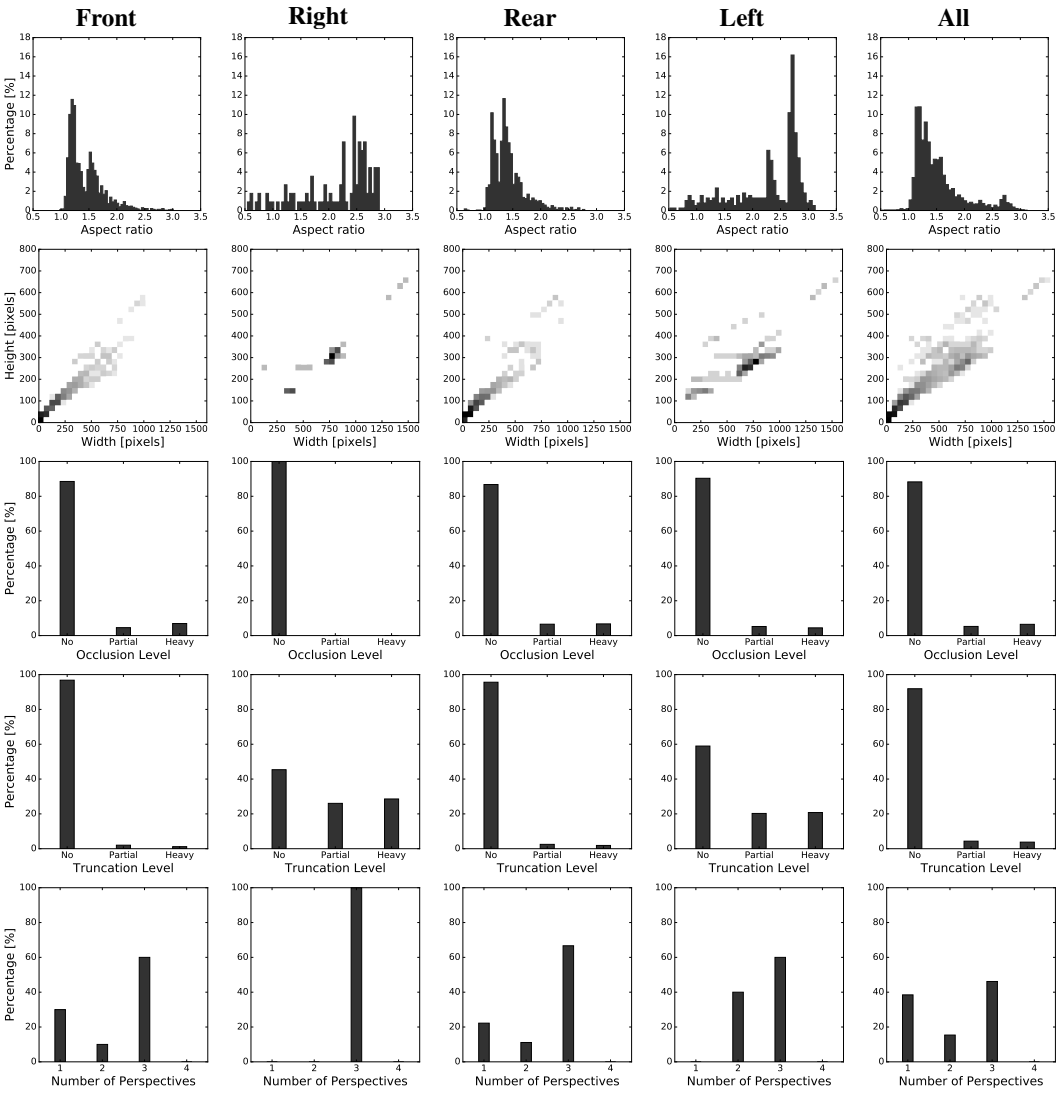


Fig. 4.

III. Experimental Evaluation

In this section we evaluate methods for each of the tasks; detection, tracking, and 3D tracking. Vehicle detection is performed for each of the four inputs of the cameras. The detections in each perspective are used by a vehicle tracker to associate detections between frames for each of the four perspectives. The positions of the tracked vehicles are transformed to the road surface, where the trajectories are connected between perspectives. A detailed description of the implementation has been submitted for a journal and is currently under review.

In this baseline a bounding box overlap criterion of 0.7 is used for both detection and tracking. A partial truncation level is used to evaluate up to 50% truncated vehicles. Heavily truncated vehicles are ignored i.e. not included even if it is correctly detected or not. Also, vehicles with a height less than 35 pixels are ignored, since these are far away from the ego vehicle (approximately 50 meters) and therefore not of interest. Lastly, an ignore region is defined to ignore oncoming traffic on the other side of the crash barrier.

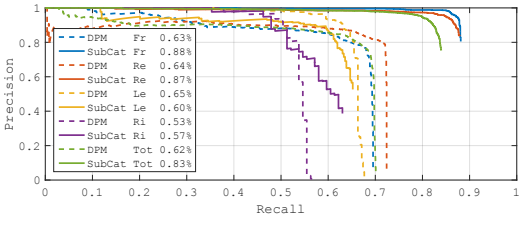


Fig. 5. Precision-Recall curve of DPM and SubCat detections with the average precision denoted in the label. Note the lower precisions in the side perspectives. Evalutated with a minimum bounding box overlap 0.7 and up to 50% occlusion and 50% truncation level.**Remake with new SubCat, and remake label**.

Both the DPM [21], [22], [23] and the SubCat [24] are tested on the proposed dataset for **detection** of vehicles in the four different perspectives. The detectors are evaluated using the average precision calculated as the area under the precision-recall curve found in Fig. 5. SubCat is found to outperform the DPM, especially in the front and rear perspectives, with mixed results for the side perspectives. This shows that the side views introduce new challenges compared to the traditional front and rear perspective. DPM employs parts, which implies more robustness to distortion in appearance due to side views. SubCat learns many models, so on normal settings it operates better on rigid or quasi-rigid objects like vehicles. DPM is found to run at 60 seconds per frame on full resolution images of 2704×1440, even though speed up can be achieved [25]. Compared to SubCat running at 1 second per frame being significantly closer to real-time.

The detections are used by a modified version of the MDP tracker presented in [26] to track the vehicles between frames. The **tracking** evaluation is presented in Table II.

Multi-Perspective 3D Tracking

SubCat + Modified MDP + 3D tracking

Table with MOTA and MOTP similar to the individual perspectives

MOT metrics with y distance has double cost (1m=2m). Matching using c<ax+b (a=0.04,b=2,x is gt distance from ego). MOTP does not have the same meaning anymore, since cost is not normalized. It is rather the average distance from TP to GT. So lower is better.

IV. Concluding Remarks

Acknowledgment

The authors would like to thank their colleagues at the Laboratory for Intelligent and Safe Automobile (LISA), University of California, San Diego, for assisting with the data gathering and their invaluable discussions and comments.

References

[1] S. Sivaraman and M. Trivedi, "Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, 2013.

[2] M. S. Kristoffersen, J. V. Dueholm, R. K. Satzoda, M. M. Trivedi, A. Møgelmose, and T. B. Moeslund, "Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016.

[3] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-Oriented Bayesian Networks for Detection of Lane Change Maneuvers," *Intelligent Transportation Systems Magazine, IEEE*, vol. 4, no. 3, pp. 19–31, Fall 2012.

[4] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, 2014.

[5] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas, "A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera," in *IEEE Conference on Intelligent Transportation Systems*, 2012.

[6] C. Wang, Y. Fang, H. Zhao, C. Guo, S. Mita, and H. Zha, "Probabilistic Inference for Occluded and Multiview On-road Vehicle Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, 2016.

[7] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "DETRAC: A New Benchmark and Protocol for Multi-Object Tracking," *CoRR*, 2015.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[9] Laboratory for Intelligent and Safe Automobiles, UCSD, "Vision for Intelligent Vehicles and Applications (VIVA)." [Online]. Available: http://cvrr.ucsd.edu/vivachallenge/

[10] N. Das, E. Ohn-Bar, and M. M. Trivedi, "On Performance Evaluation of Driver Hand Detection Algorithms: Challenges, Dataset, and Metrics," in *IEEE Conference on Intelligent Transportation Systems*, 2015.

[11] S. Martin, K. Yuen, and M. M. Trivedi, "Vision for Intelligent Vehicles and Applications (VIVA): Face Detection and Head Pose Challenge," in *IEEE Intelligent Vehicles Symposium*, 2016.

[12] A. Møgelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, 2012.

[13] A. Møgelmose, D. Liu, and M. M. Trivedi, "Detection of U.S. Traffic Signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, 2015.

[14] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, 2016.

[15] R. K. Satzoda and M. M. Trivedi, "Looking at Vehicles in the Night: Detection & Dynamics of Rear Lights," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[16] B. T. Morris and M. M. Trivedi, "Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, 2011.

[17] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *Journal on Image and Video Processing*, 2008.

[18] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," *CoRR*, vol. abs/1603.00831, 2016.

[19] B. Wu and R. Nevatia, "Tracking of Multiple, Partially Occluded Humans based on Static Body Part Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006.

[20] A. Milan, K. Schindler, and S. Roth, "Challenges of Ground Truth Evaluation of Multi-target Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013.

[21] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, 2010.

[22] H. Zhang, A. Geiger, and R. Urtasun, "Understanding High-Level Semantics by Modeling Traffic Patterns," in *IEEE International Conference on Computer Vision*, 2013.

TABLE II

SINGLE PERSPECTIVE TRACKING RESULTS.

Track/persp	MOTA↑	MOTP↑	IDS↓	Frag↓	MT↑	ML↓	Recall↑	Precision↑
Front								
SubCat-MDP	0.82	0.83	1	1	0.80	0.00	0.83	1.00
DPM-MDP	0.71	0.78	0	0	0.80	0.10	0.81	0.89
Rear								
SubCat-MDP	0.82	0.85	0	9	0.75	0.00	0.87	0.94
DPM-MDP	0.87	0.80	1	4	0.75	0.00	0.87	1.00
Left								
SubCat-MDP	0.76	0.77	0	1	0.40	0.20	0.76	1.00
DPM-MDP	0.77	0.80	0	1	0.40	0.40	0.77	1.00
Right								
SubCat-MDP	0.55	0.83	0	0	0.33	0.33	0.55	1.00
DPM-MDP	0.62	0.82	0	0	0.67	0.33	0.62	1.00
Total								
SubCat-MDP	0.81	0.84	1	11	0.65	0.08	0.83	0.97
DPM-MDP	0.79	0.79	1	5	0.69	0.15	0.83	0.95

TABLE III

MULTI-PERSPECTIVE TRACKING RESULTS.

Tracker	MOTA↑	MOTEP↓	IDS↓	Frag↓	MT↑	ML↓	Recall↑	Precision↑
SubCat-MDP-Kalman	0.64	1.23	5	93	0.46	0.15	0.79	0.85
DPM-MDP-Kalman	0.42	1.23	3	83	0.38	0.15	0.74	0.70

[23] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D Traffic Scene Understanding From Movable Platforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, 2014.

[24] E. Ohn-Bar and M. M. Trivedi, "Learning to Detect Vehicles by Clustering Appearance Patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, 2015.

[25] J. Yan, Z. Lei, L. Wen, and S. Li, "The Fastest Deformable Part Model for Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[26] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track: Online Multi-Object Tracking by Decision Making," in *IEEE International Conference on Computer Vision*, 2015.