# Vision-based Prediction of Human Driver Behavior in Urban Traffic Environments

Von der Technischen Fakultät der Universität Bielefeld zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation, vorgelegt am 11.02.2013

von Diplom-Informatiker
**Martin Heracles**
aus Linnich

Tag der mündlichen Prüfung: 31.10.2013

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

# Abstract

We address the problem of inferring the appropriate behavior of a human driver from visual information about urban traffic scenes. The visual information is acquired by an on-board camera that monitors the scene in front of the car, resulting in a video stream as seen by the driver. The appropriate behavior consists in the actions a responsible driver would typically perform in the depicted situations, including both longitudinal and lateral control. As solving the problem would enable a technical system to generate independent behavioral expectations, potential applications are in driver assistance and autonomous navigation.

While autonomous vehicles have mastered highway, off-road, and urban traffic environments by now, their perceptual basis has fundamentally shifted towards non-visual sensors. The same is true of driver assistance systems, which are in addition limited to specific functions like collision avoidance or lane keeping. Partly, the reason lies in the complexity of urban traffic scenes, being rich in visual information and often densely populated by other traffic participants. Moreover, their diversity complicates their relationship to driving behavior: Many situations require the same behavior while others allow for several alternatives.

In this context, we propose a novel framework based on scene categorization that approaches the problem from its behavioral side: Subdividing the behavior space induces visual categories for which dedicated classifiers are then learned. The visual complexity is handled by decomposing the traffic scenes into their constituent semantic entities and computing object-level features. While using known techniques, our linking them to actual human driver behavior is also novel. To validate our approach, we conduct experiments on video streams recorded in real urban traffic, including a detailed comparison to the state-of-the-art.

Our results give compelling evidence of the superior robustness of our system, compared to the filter-based representation of the current method. This finding is consistent with general results in scene categorization and emphasizes their importance for behavior prediction. Moreover, our scene categorization based behavior prediction framework offers exciting possibilities for future research. Examples include a route-planning layer on top of the proposed system to go beyond reactive behavior, multi-modal extensions by audio or tactile sensors to enrich the perceptual basis, and real-time applications in the automotive domain.

# Acknowledgments

First of all I would like to thank my advisors, Jannik Fritsch and Sven Wachsmuth, for their continued support, patience and guidance along the way. This work would not have been possible without them.

I would also like to express my gratitude to Gerhard Sagerer, my original advisor, for the encouraging discussions we had in the beginning of this research.

Further, I gratefully acknowledge the Honda Research Institute Europe GmbH for funding the research project underlying this thesis work.

Thanks also go to my colleagues at the Honda Research Institute Europe in Offenbach/Main, and to the other PhD students at the Research Institute for Cognition and Robotics in Bielefeld.

Finally, my sincere gratitude is with my family and friends, who have accompanied all the different stages of my PhD project with great patience and understanding. Especially Ellen, to whom I am deeply indebted, thank you for all your support!

# Contents

# Chapter 1

# Introduction

## 1.1 Objective

Human drivers perceive the traffic situations they encounter mainly by means of their visual senses. Based on the information thus acquired, they ultimately arrive at behavioral decisions that affect the course of the vehicle in a way that is appropriate for the traffic situations at hand. How this mapping from visual input to behavioral output is actually achieved, in terms of the underlying neurobiological processes, is still far from being fully understood. Nevertheless, the fact that human drivers successfully perform this task every day convincingly demonstrates that vision-based generation of appropriate driving behavior is indeed possible. Our goal is to enable a technical system to do the same (see Figure 1.1).

To this end, the system observes the scene in front of the car by means of an on-board camera. The resulting video stream closely matches the perspective of the human driver and depicts what he or she sees while driving. In addition, the system also observes what the driver is actually doing in terms of braking, accelerating, and steering. This information can be acquired from the CAN bus of the vehicle, or reconstructed from the video stream itself. Given the video stream and the behavior stream, the goal of the system is to learn how the two are correlated with each other. After learning, the system should be able to predict the behavior data from the visual stream alone. We thus refer to the problem as *(driving) behavior prediction*, and a formal definition is given in Chapter 3.

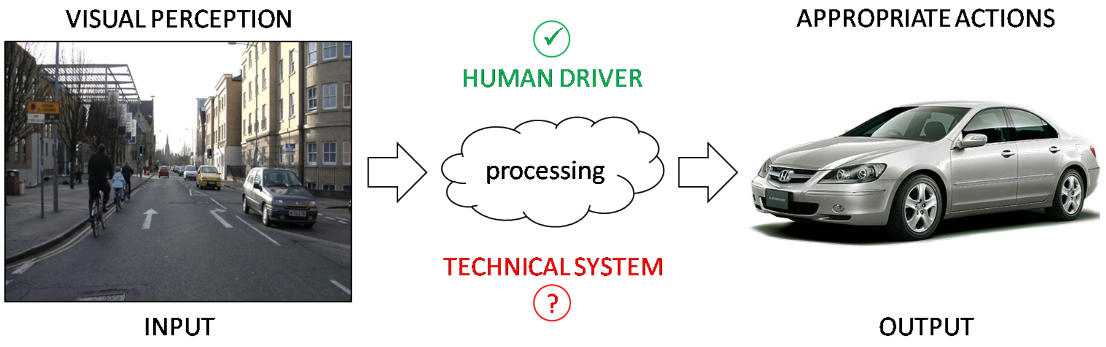


Figure 1.1: Human drivers essentially perform a mapping from visual information about the current traffic situation to appropriate behavioral commands affecting the course of the vehicle. Our goal is to enable a technical system to do the same.

## 1.2 Motivation

A technical system capable of interpreting visual information about traffic scenes such that it gains a basic understanding of what human drivers would typically do in these situations has potential applications in the automotive domain. Examples of such applications include driver assistance and autonomous navigation.

### Driver Assistance

Although humans are generally capable of driving a car with apparent ease, they are subject to occasional mistakes. Such mistakes are often caused by lack of attention in a critical moment, potentially leading to dangerous situations and traffic accidents. Failure of technical components, in contrast, can be ruled out nowadays by rigorous quality and maintenance standards. It is therefore useful to have a technical system observe the same traffic situations as the human driver and generate its own expectations of appropriate driving behavior. The system could then compare these expectations to the actual behavior of the human driver. Given a sufficiently high confidence of the system in its own predictions, any mismatch indicates a potential mistake by the human driver. In these cases, the system could react by providing a warning signal to notify the driver, or by directly executing the expected behavior (see Figure 1.2).

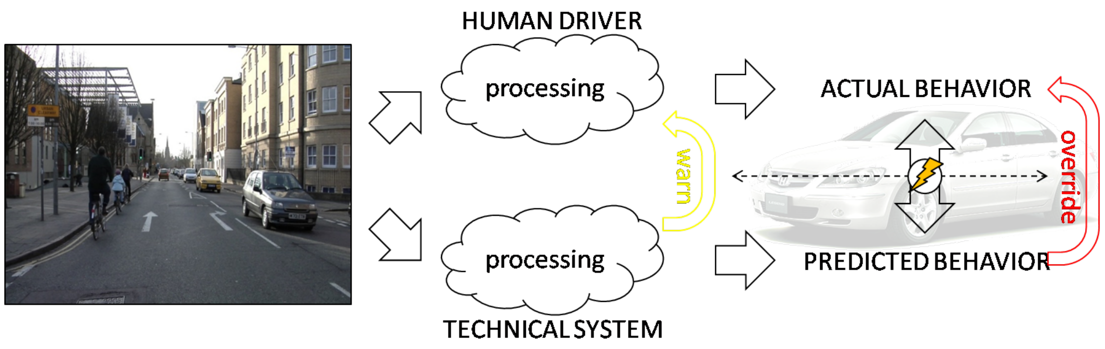


Figure 1.2: The trained system is able to generate its own behavioral expectations. By comparison to the actual behavior of the human driver, the system could detect mismatches and react – by warning the driver or by direct intervention.

### Autonomous Navigation

In the above example of driver assistance, the system is designed to remain passive unless it detects a mismatch between the actual behavior of the human driver and its own predictions. Nevertheless, the predictions made by the system actually form a continuous stream, and they are completely independent from what the human driver is actually doing, depending solely on the visual information about the given traffic situations. For this reason, the system could as well be designed to actively execute the predicted driving behavior for extended periods of time, thus constantly circumventing the human driver.

Traffic situations that allow for multiple behavioral alternatives, however, would require additional information to be resolved. Crossings are an example of such situations, as the choice whether to continue driving straight, turning left, or turning right depends on the intention of the human driver and cannot be told from the visual information itself. The required information could be provided by the human driver via operating the blinker of the vehicle, for example, or by a dedicated route-planning layer on top of the proposed system. The reason for this necessity is that the system has to make a decision on which driving behavior to execute at any given point in time, whereas in driver assistance the system may refrain from making a decision if the confidence in its own predictions is not high enough. Details on this confidence mechanism are given in Chapter 3.

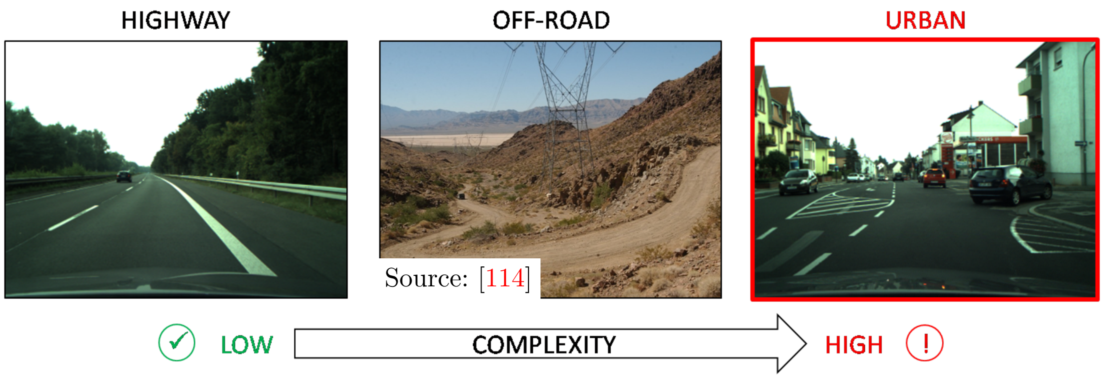


Figure 1.3: Highway, off-road, and urban traffic environments can be handled to the point of autonomous driving by now, but their increasing complexity requires extensive sensor technology. No purely vision-based solution exists to date.

## 1.3 Challenges

Despite the practical relevance of driving behavior prediction, a purely vision-based solution for urban traffic environments still eludes us. Major factors that contribute to the difficulty of this problem are the high visual complexity of urban traffic scenes, their non-trivial relation to appropriate driving behavior, and global scene conditions that fundamentally affect their visual appearance.

### Visual Scene Complexity

Urban scenes are more complex than other traffic environments (see Figure 1.3): For example, highway environments are primarily composed of the road itself, trees in the background, and the sky. Interaction with other traffic participants, predominantly with other cars, is relatively infrequent and usually involves a large safety distance due to the overall high velocity. Clear lane markings and the absence of crossings or junctions facilitate the visual processing of highway scenes. Urban environments, in contrast, are characterized by rich scene content that typically comprises the road, sidewalks, buildings, trees, and the sky. Other traffic participants include cars, bicyclists and pedestrians, with frequent interactions at often close range. Irregular lane markings, as well as the presence of crossings and junctions, further contribute to the complexity of urban traffic environments.

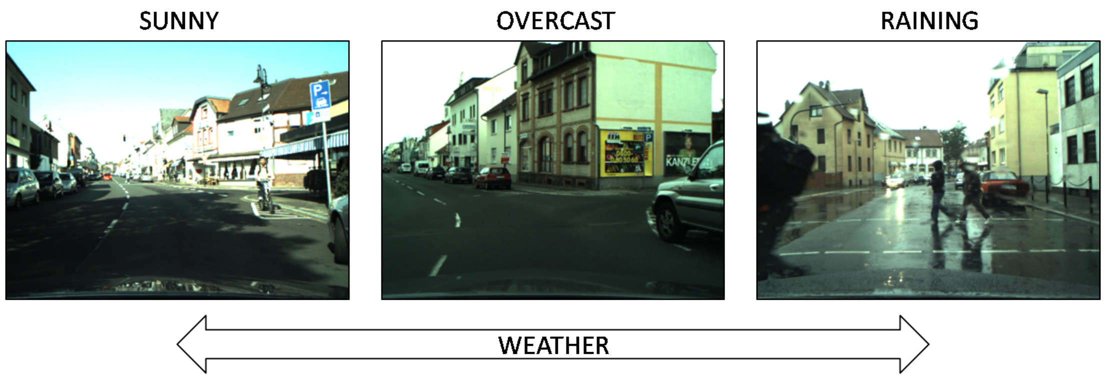


Figure 1.4: The robustness of visual processing is challenged by uncontrollable weather conditions. For example, the road may exhibit strong shadows (left), appear as homogeneous surface (center), or contain bright reflections (right).

### Changing Weather Conditions

Visual traffic scene understanding is complicated by the fact that global factors such as the weather can have a great effect on the visual appearance of a scene (see Figure 1.4). The resulting changes in visual appearance impose additional robustness requirements on vision-based approaches to driving behavior prediction, which are currently circumvented by the use of non-visual sensors (see Chapter 2).

### Appropriate Driving Behavior

While the appropriate driving behavior for highway environments largely consists in driving straight at a constantly high velocity, urban traffic environments require a much broader range of behaviors: Due to the frequent interactions with other traffic participants and the presence of crossings, junctions, and curves, driving in urban scenes involves considerable velocity changes and steering maneuvers. The difficulty for technical systems trying to learn the appropriate driving behavior lies in the complex dependencies of such actions on the visual scene content: On the one hand, we have already seen examples of traffic situations that allow for more than one possible driving behavior, depending on the driver's intentions (e.g., at crossings and junctions). On the other hand, many traffic situations that are visually dissimilar actually require the same driving behavior (see Figure 1.5).

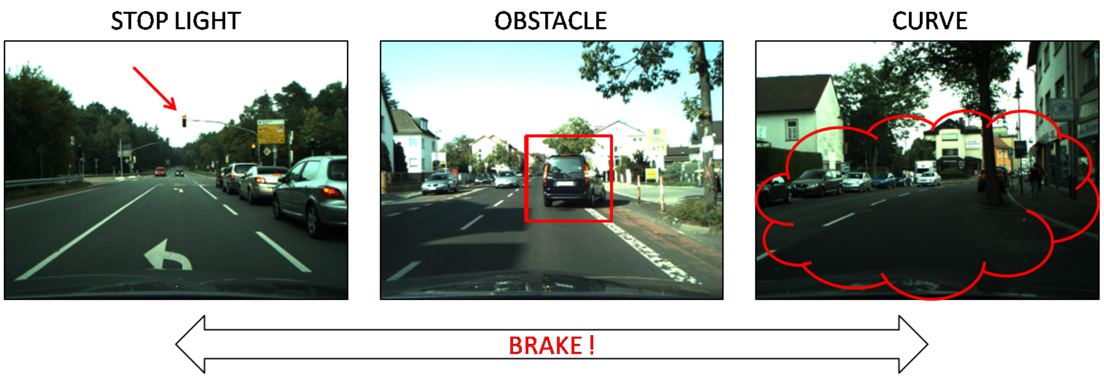


Figure 1.5: The complexity of urban traffic situations complicates the mapping between visual scene content and appropriate driving behavior. For example, braking can be due to a small stop light, an extended obstacle, or a diffuse curve.

## 1.4 Contributions

To overcome the above challenges, our dissertation work makes the following contributions to driving behavior prediction in urban traffic environments using visual perception alone.

### Scene Categorization Architecture

We propose a novel system architecture based on visual scene categorization that enables supervised learning of the appropriate driving behavior in reaction to the visual content of urban traffic scenes. To cope with the visual dissimilarity of traffic situations that are behaviorally equivalent (see Figure 1.5), we employ a behavior-driven approach that begins by subdividing the behavior space into discrete classes. These behavior classes induce image categories in the visual domain, consisting of all scenes in which the respective behavior was actually performed by a human driver. For each visual category, a dedicated image classifier is then trained in a one-versus-all manner, to distinguish its corresponding traffic scenes from all others. By applying these classifiers and deciding on the behavior class associated with the strongest response, the trained system is able to predict the appropriate behavior for previously unseen traffic situations. To our knowledge, we are the first [43] to propose a scene categorization approach for this problem.

### Visual Scene Decomposition

Our scene categorization architecture itself does not specify how to represent the traffic scenes for subsequent processing by the image classifiers. In order to break down the visual complexity of urban traffic scenes, we therefore propose a full decomposition into their constituent semantic entities. This decomposition is achieved by learning a dedicated object classifier per semantic entity type and integrating their response maps in a Conditional Random Field framework using energy minimization (see Chapter 4). While the underlying techniques and the resulting object-level features are well-known state-of-the-art methods, our contribution lies in explicitly linking them to human driver behavior and demonstrating their suitability for behavior prediction. Classical segmentation typically remains in the visual domain only, without addressing its usefulness for scene understanding in the context of a larger system. Driver assistance systems, in turn, usually focus on a manually selected subset of the above semantic entities as governed by their assumed behavioral relevance. In contrast, we consider the full range of semantic entities and let the system learn which of them are relevant.

### Comparative Performance Analysis

The segmentation-based representation of urban traffic scenes as described above essentially operates at the level of semantic entities and objects, which reflects a degree of abstraction that is generally associated with higher visual processing in the human brain. An alternative representation [84] has recently been proposed for driving behavior prediction that, unlike our approach, operates at the level of raw image filter responses and can be seen as a model of early visual processing prior to attention. As these representations correspond to fundamentally different ends of the cognitive spectrum, we conduct a comparative performance evaluation (see Chapter 5), which has not been done before. Our experiments include a principled investigation of the effect that different weather conditions have on the performance robustness, and also demonstrate the importance of stabilizing the single-frame predictions over time, in combination with a confidence-based reject option to temporarily suppress ambiguous predictions. Our results show the superior robustness of our approach over the best method currently known.

## 1.5 Thesis Outline

The rest of this dissertation is organized as follows.

**Chapter 2.** We begin with an overview of the current state-of-the-art in autonomous vehicles, driver assistance systems, and visual scene understanding. Our literature review shows that autonomous navigation in real urban traffic is possible by now, but generally requires non-visual sensor technology. In contrast, driver assistance systems often employ visual processing, but typically only for isolated support functions such as lane departure warning, for example. However, visual scene understanding has reached a maturity level that enables us to tackle driving behavior prediction in real urban traffic by visual perception alone.

**Chapter 3.** We proceed by describing our scene categorization architecture for driving behavior prediction. Our discussion starts with a problem formulation that includes a specification of the available data, the proposed subdivision of the behavior space, and how the resulting visual classes lead to scene categorization. We then explain the supervised learning that is enabled by our framework, from acquiring the ground truth behavior data over balancing the training examples to learning a multi-classifier for the actual prediction. A qualitative comparison to a related architecture for driving behavior prediction concludes this chapter.

**Chapter 4.** Our qualitative comparison of the two architectures continues with their different traffic scene representations. We first discuss the filter-based representation of the alternative approach, describing its oriented edge filters, pooling over image grid cells, and stabilization by Gaussian kernel weights. We then explain the segmentation-based representation used by our own approach, including its semantic object classifiers, the visual scene decomposition, and our feature vector computation. Two different implementations are given at the end.

**Chapter 5.** Finally, we report on the prediction accuracy achieved by our proposed approach, with detailed comparison to the state-of-the-art performance. After discussing public video datasets from the perspective of behavior prediction, we first conduct a stand-alone evaluation of the object-level representation. We then address the adequate definition of the behavior classes, evaluate the effect of different weather conditions as well as the temporal stabilization with confidences, and conclude our work with automatic weather recognition for model selection.

# Chapter 2

# State of the Art

As pointed out in the previous chapter, our work has potential applications in autonomous navigation and driver assistance, and we emphasize visual perception to infer the appropriate driving behavior in a given traffic situation. We therefore provide a brief overview of the current state-of-the-art in autonomous vehicles, discuss existing behavior prediction methods for driver assistance, and introduce relevant computer vision techniques for visual scene understanding.

Our review of the autonomous vehicles literature shows that driverless cars are now capable of successfully navigating in highway, off-road, and urban traffic. Their sensory processing, however, is largely based on extensive technical setups rather than on visual perception as in human drivers. The same is true of most driver assistance systems, which are specifically designed to implement a variety of individual support functions. In this context, driving behavior prediction is a well-established research field of its own, but its focus is more on recognizing the actual behavior of the human drivers to infer their intentions and predict their future trajectories. In contrast, we focus on the inference of the appropriate behavior in a given traffic situation from the visual scene content itself.

Computer vision techniques have sufficiently matured by now to be applicable to real-world urban traffic environments. While they generally do not go beyond the visual domain, however, our work establishes a direct link to driving behavior as part of a scene categorization approach (see Chapter 3). In doing so, we also promote a fundamental shift away from the non-visual sensors that, in our view, essentially reflect an apparent cognitive limitation in the systems of today.

## 2.1 Autonomous Vehicles

Although the field of mobile robotics dates back to the late 1960s [71, 72, 75], it was not until the late 1980s that such technology was successfully applied to commercially available cars. Early work in this direction, particularly in Europe, had a strong focus on autonomous navigation in highway environments [13, 26], due to their rather well-structured nature and low scene complexity. Importantly, these systems relied on visual perception alone, and demonstrated remarkable driving skills over large distances and at high velocities (see Section 2.1.1).

In contrast, research efforts on autonomous vehicles in the United States were overall tailored towards off-road navigation from the very beginning [16, 90, 101], corresponding to different military applications of this technology. Apart from a limited number of early approaches that were based on visual processing [83, 108], GPS and map data played a central role in this context, and non-visual sensors became a hallmark of the resulting vehicles. Impressive achievements were made particularly in unstructured and physically difficult terrain (see Section 2.1.2).

Given the success of autonomous vehicles in off-road environments, it was a natural transition to apply similar technology to urban scenarios as well [17]. While originally restricted to rather artificial settings with simplified conditions, recent efforts are clearly heading towards a deployment in real city traffic [62, 102]. As a consequence, sensors are now being integrated more seamlessly into the prototype vehicles than before, but the underlying technology is still based on non-visual perception such as given by 360° laser scanners (see Section 2.1.3).

In the following, we briefly discuss the most important milestones.

### 2.1.1 Highway Traffic

One of the leading pioneers of autonomous vehicles was Ernst Dickmanns from the Bundeswehr University Munich in Germany, who equipped a Mercedes-Benz van called *VaMoRs* with cameras in a stereoscopic arrangement, and also modified it such that steering and acceleration could be controlled electronically rather than by human mechanical operation [26]. Using computer vision techniques alone, *VaMoRs* was capable of autonomous road-following on public highways by 1987 at velocities of up to 96 km/h, but without any other traffic participants [28].

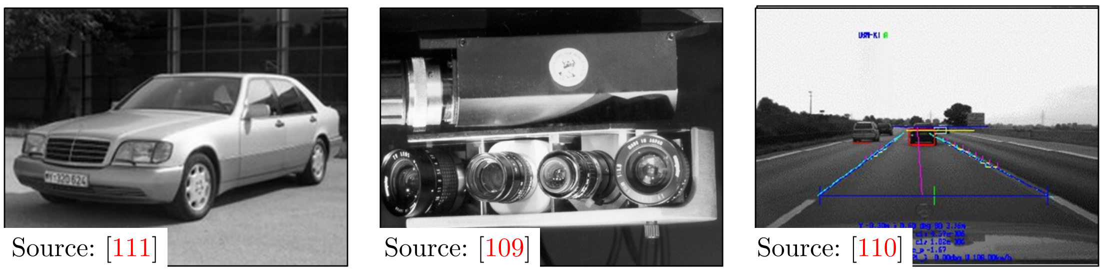


Figure 2.1: The *VaMP* car (left) was capable of autonomous driving on highways over large distances and at high velocities (right), solely based on visual perception as given by a movable array of video cameras (center).

In the following EUREKA Prometheus project of the European Union [9, 10], comparable in funding and participation to the later DARPA Grand Challenges as discussed in Section 2.1.2, the autonomous passenger car *VaMoRs-P* (or *VaMP*) was created in collaboration with Daimler-Benz (see Figure 2.1). Like *VaMoRs*, it was equipped with several cameras on a movable rig and relied on stereo vision. In 1994, as part of the final demonstration of the EUREKA Prometheus project, *VaMP* demonstrated its autonomous driving capabilities on a highway near Paris in France, including regular traffic and achieving up to 130 km/h velocity [27]. One year later, an improved version drove autonomously from Munich in Germany to Odense in Denmark, traveling a distance of over 1600 km on public highways at velocities of up to 175 km/h [25].

After the Prometheus project, Alberto Broggi from the University of Parma in Italy adapted a regular Lancia passenger car named *ARGO* to be capable of autonomously turning the steering wheel, by simply attaching an electric motor. Also, standard cameras were installed in a stereoscopic setup with large baseline to monitor the scene in front [13]. The emphasis was on low technical overhead, contrasting the highly engineered *VaMoRs* and *VaMP* with their sophisticated camera rigs. Although *ARGO* was capable of autonomous steering only, with a human driver having to operate the pedals, it demonstrated similar performance as *VaMP* when completing a 2000 km journey on Italian highways in 1998 [12]. Apart from occasional human intervention for safety reasons, the above examples all show that autonomous navigation in real highway traffic is largely solved.

### 2.1.2 Off-road Navigation

As we have seen in the previous section, visual processing was highly successful in providing control for autonomous vehicles. However, these approaches were all designed for operation in well-structured traffic environments with a known infrastructure such as highways, in which lane following and obstacle detection are the fundamental tasks. It is therefore no surprise that off-road environments, being largely unstructured and also imposing additional tasks like route planning, attracted non-visual sensor technology in addition.

An early example in this context is the DARPA-funded *Autonomous Land Vehicle (ALV)* project [90], of which the *NavLab* vehicle from Carnegie Mellon University was a promising candidate. The original *NavLab* was a large van-like mobile platform that was particularly suited for rough terrain, and its perception of the environment was based on cameras on the one hand as well as on 3D laser scanners on the other hand [101]. In the beginning, research on this vehicle included neural network approaches that relied solely on the cameras [82, 83], but these techniques were predominantly demonstrated in highway scenarios like their European counterparts, with similar results around 1990.

The extended period of *ALV*-related research ultimately led to the DARPA Grand Challenge, which was first held in 2004 and basically consisted in a large off-road race through the Mojave desert of California, USA [20]. The final race involved 15 autonomous vehicles and a previously unknown route spanning about 142 miles. None of the vehicles achieved this goal, however, mostly because of mechanical problems due to the rough terrain or because of getting stuck. The best vehicle, *Sandstorm* from Carnegie Mellon University, made it 7 miles.

Although unsuccessful, interest in the event was nevertheless increased, and the DARPA Grand Challenge was held again in the subsequent year [16]. The new route was comparable to the 2004 route, being a 132 miles parcours through the Mojave desert, in rugged terrain and with narrow paths (see Fig. 2.2, left). Specifically, it included three particularly narrow passages such as a tunnel, a gate, and a pass seamed by rocks on one side and a cliff on the other (see Fig. 2.2, center). Also, the total number of curves was much higher than in the year before. Only vehicles were admitted to the final race that demonstrated sufficient

Figure 2.2: The DARPA Grand Challenge required autonomous navigation in an off-road desert environment (left), characterized by unstructured terrain (center). It was successfully completed by *Stanley* (right) as well as four other vehicles.

capabilities of following paths, avoiding obstacles and collisions, and navigating through tunnels, so 23 teams participated in the final race. As before, the time limit for completing the course was 10 hours, and map data of the route was only made available shortly before the race. Five vehicles were able to complete the entire 132-miles trip, the fastest taking less than 7 hours (see Fig. 2.2, right). *Stanley* from Stanford University won the first place [104], followed by *Sandstorm* and *H1ghlander* from Carnegie Mellon University [122].

As pointed out earlier, it is important to note that the success of the Grand Challenge was not built on visual processing alone. In contrast to most European vehicles, the DARPA cars had a strong tendency towards relying on technical sensors such as ultrasonic, radar and lidar, or laser scanners in general. If at all, cameras were used as auxiliary sensors only. For example, *Stanley* explicitly made use of visual information for detecting the drivable road area, thus complementing the GPS and map information, whereas *Sandstorm* did not utilize visual information at all. Also, the use of map data with GPS waypoints was inherent to the Grand Challenge, as these defined the course to be followed. To a large extent, the actual challenge therefore consisted in properly localizing the vehicle on the map, typically by GPS in combination with inertial measurement units to account for odometry information, and matching the coarse map data to the actual terrain as perceived by the high-resolution laser scanners. This architectural paradigm remained strong in the subsequent approaches to autonomous driving in urban traffic environments, given its success in the off-road terrain.

### 2.1.3 Urban Environments

The same technology that enabled autonomous driving in off-road environments was put to the test again in the 2007 DARPA Urban Challenge [17]. For safety, an abandoned US airbase was chosen for the event rather than a real city, but it shared many characteristics with real urban environments such as paved roads, sidewalks, vegetation and buildings, as well as intersections and T-junctions with traffic signs. Although no pedestrians were present, the vehicles navigating in the scenery frequently encountered each other at the intersections, where they had to obey standard traffic rules such as precedence of driving. Compared to the Grand Challenge, the environment thus proved to be much more dynamic. Like before, a map of the site was provided shortly before the race, but each vehicle was assigned missions to be completed by driving autonomously along routes defined by GPS waypoints.

Six vehicles were able to complete their missions, partly within the time limit of 6 hours [3, 6, 34, 69, 70, 112]. Many other vehicles, however, either crashed into obstacles or were shut down because of dangerous behavior. Nevertheless, the event successfully demonstrated the ability of autonomous vehicles to handle constrained urban-like traffic environments. One important technical advance over the previous Grand Challenge cars was the use of a 360° LIDAR mounted on top. These high-velocity, high-resolution sensors create a detailed sensory representation of the surrounding traffic environment without having a blind spot, apart from a rather small radius in the immediate area around the vehicle. To compensate for this weakness, and to further enhance the sensory coverage of the environment, the classical SICK laser scanners from the earlier Grand Challenge are still heavily used, typically in combination with the 360-degree LIDAR. With the increasingly complex urban traffic environment, cameras have also become more important than before, as can be seen in *Boss* and *Odin* with their large double-cameras mounted on top. Nevertheless, visual sensors remained just one of several non-visual types in these vehicles.

After the success in constrained urban-like environments, the technology was gradually transferred to real city traffic. In continued collaboration with VW, Sebastian Thrun from Stanford University led the *Google Driverless Car* project,

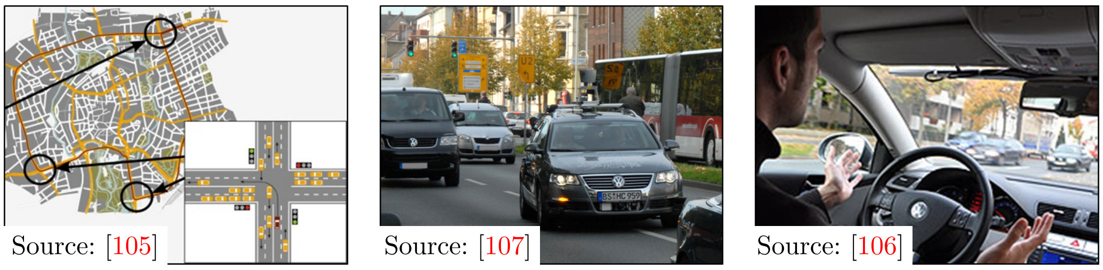


Figure 2.3: The autonomous vehicle *Leonie* of the Braunschweig *Stadtpilot* project is capable of driving in real-world urban traffic, but requires laser-based and radar-based perception with GPS information instead of visual sensors.

aiming for fully autonomous driving in urban traffic [102]. A fleet of seven vehicles was equipped with technical sensors similar to those used in the Urban Challenge, but they were more decently integrated. Specifically, each vehicle had a 360° LIDAR on top, three cameras inside the cabin to monitor the upcoming traffic scene, as well as laser scanners and radar sensors to the front. The sensors give rise to a detailed 3D representation of the vehicle's surroundings, and Google *Street View* information is combined with the sensor data. Like in the DARPA Urban Challenge, GPS is used for route planning and localization.

Another current successor of the Urban Challenge is the *Stadtpilot* project [62], in which the TU Braunschweig builds on their experience with the *Caroline* vehicle, one of the 11 Urban Challenge finalists. Their new vehicles *Leonie* and *Henry* are modified VW Passats and were equipped with a similar sensor setup like the Google Driverless Cars, including a 360-degree LIDAR on top as well as laser scanners and radar sensors (see Figure 2.3). In contrast to the Google cars, cameras are not yet installed, and the status of traffic lights has to be provided manually. Apart from this limitation, the system also relies on GPS and aerial map data to localize itself, and autonomously drove along the 3-km inner ring of Braunschweig in Germany. The demonstration included automatic merging into traffic, turning maneuvers at intersections, lane changes, and autonomous parking. Overall, we conclude that autonomous navigation in real city traffic is already possible to a large extent, but nevertheless still ongoing research and generally requires non-visual perception to cope with the scene complexity.

## 2.2 Driving Behavior Prediction

According to the well-known driver model proposed by Michon [68], which is an automotive version of the more generic cognitive architectures for modeling human behavior in general [88], the cognitive process of driving a car is thought of as involving tasks at three hierarchical levels of abstraction: Overall destination and route planning are dealt with at a strategic level, the appropriate selection of driving maneuvers is handled at a tactical level, and their actual execution by motor commands is performed at an operational level. While autonomous vehicles as discussed in the previous section necessarily involve tasks at all three levels, driver assistance systems typically focus on the tactical level, as their purpose is to assist the human driver in the current or upcoming traffic situation.

In this context, behavior prediction was originally designed to recognize the current maneuver of the human driver at an early stage, to provide appropriate support that is consistent with his or her intentions (see Section 2.2.1). While early approaches only considered the car's physical data for maneuver recognition, other work also included the driver's gaze behavior and other traffic participants in the surrounding environment. Consequently, the focus of behavior prediction shifted from the ego-vehicle towards other cars, thus modeling the traffic situation as a whole (see Section 2.2.2). Such models try to anticipate the likely trajectories, as these are potentially relevant for the ego-vehicle. Recently, machine learning has been used for direct correlation of the actions of a human driver with the traffic situations in which they are typically being performed (see Section 2.2.3). Our own approach, given in Chapter 3, is a representative of the latter group.

### 2.2.1 Maneuver Recognition

The goal is to recognize the ongoing or upcoming maneuver as early as possible in order to be able to offer adequate assistance to the driver, as opposed to classifying entire maneuvers after their completion [37]. The key idea is that driving maneuvers are manifestations of the driver's intentions, which are not directly observable as they depend on his or her internal cognitive state. Thus, the internal state must be inferred from other (observable) quantities, such as changes in the velocity and the steering angle of the vehicle, for example.

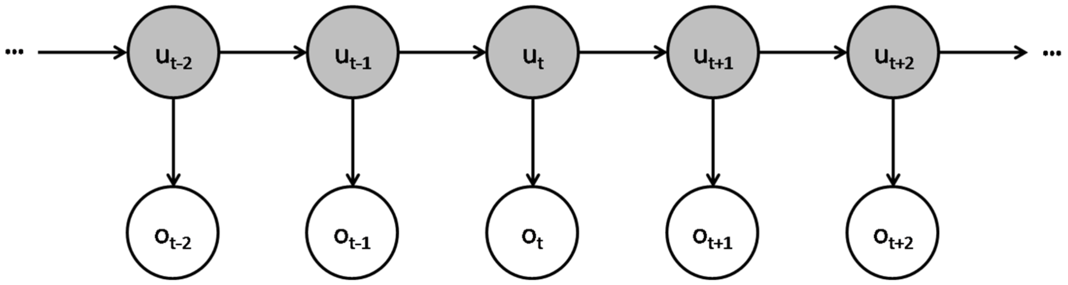


Figure 2.4: Example of a Hidden Markov Model as used for maneuver recognition.

Hidden Markov Models (HMMs) [45] and various extensions are widely used for maneuver recognition, as they offer a principled framework for inferring the unobservable states from observable evidence, and also lend themselves naturally to the modeling of temporal sequences (see Figure 2.4): Each maneuver is represented by a pre-defined series of atomic steps (or states) of which it consists, with transition probabilities linking the individual states in a chain, and output probabilities that associate each state with its typical observations. Thanks to their probabilistic nature, HMMs are capable of dealing with uncertainty, both in terms of noisy observations by the sensors and in terms of the inherent uncertainty by the possible driving maneuvers. Also, efficient algorithms are available for learning their parameters from training data (e.g., the Baum-Welch algorithm) and for inferring the most likely sequence of internal states given the observable data (e.g., the Viterbi algorithm) [74]. The typical procedure is to represent each possible driving maneuver by a separate HMM, and the actual recognition is then achieved by determining the model that is most consistent with the observations. To this end, the Viterbi algorithm can be used as well. Being generative models, HMMs can also be used for temporal prediction into the future, by evaluating the most likely model at some future timestep without observations [51]. While classical HMMs are applied to the raw observations, thus abstracting from their continuous values and operating at the tactical level, extensions have been proposed that also incorporate operational aspects into the HMM framework [58, 80]. These approaches model the raw sensor data with Linear Dynamical Systems (LDS) such as Kalman filters, for example, and then have the HMMs operate on their stabilized output such that the continuous observations are preserved.

As for the observable data underlying the HMMs and their extensions, early approaches only considered physical quantities of the vehicle itself, which can be observed directly from the CAN bus, for example [50, 51, 80]. Since these quantities are directly correlated with the ongoing driving maneuvers, they can be used as evidence for their recognition. The first methods simply evaluated their models at some fixed timestep for early recognition, while later methods often evaluate and update the models at each timestep for continuous on-the-fly recognition [50, 93]. Some approaches model a wide variety of driving maneuvers whereas others restrict themselves to either longitudinal or lateral control (e.g., lane changing versus lane keeping maneuvers).

An important extension for earlier recognition is the direct observation of the human driver [58, 77]: Human gaze is generally influenced by the internal cognitive state, and the gaze patterns of human drivers are strongly correlated with different maneuver types [53, 73, 119]. As gaze patterns can be modeled by a first-order Markov process, their incorporation into HMMs is a natural extension. Because the driver mostly observes the traffic scene in front, such gaze patterns generally indicate the upcoming maneuvers prior to the CAN data.

Finally, the external surroundings of the vehicle have also been taken into account later on [37, 77, 93], since driving maneuvers are essentially reactions to the traffic situation at hand. Thus, external information also contributes to an earlier maneuver recognition, and information about the vehicle in front, the vehicles in the rear and adjacent lanes, as well as lane markings have been used. Maneuver recognition systems of today typically consider the full range of the driver-vehicle-environment spectrum, and achieve good recognition results several seconds before the onset of a maneuver takes place.

However, most of these approaches are designed for highway environments, and early attempts have systematically circumvented real perception by manually annotating the sensor data, although this is no longer the case. Moreover, in analogy to our discussion of autonomous vehicles in the preceding section, laser and radar sensors are heavily used instead of visual perception, which is mainly restricted to the detection of lane markings. In particular, maneuver recognition as discussed above concentrates on recognizing the *actual* (current or future) behavior of the human driver, instead of evaluating its situational necessity.

### 2.2.2 Traffic Situation Models

McCall and Trivedi address the problem of driver assistance for braking [64], using a Bayesian model that incorporates both the prediction of the driver's *intention* to brake as well as the situational *necessity* of braking. These two are handled independently, and are compared later on to determine whether or not a warning should be communicated to the driver. It is arguably the first work that explicitly distinguishes between what the driver *actually* does and what he or she *should* be doing, like in our own work. The driver's intention is estimated from camera data of the head and foot movements, and the Bayesian model performs a probabilistic weighting between the driver's intention and the situational need to assess the overall "criticality" of the current situation, where each part is modeled by a random variable. The input data forms a large feature vector that is processed over a moving time window, and Sparse Bayesian Learning (SBL) is used to extract the most relevant feature vector dimensions, thus reducing the input feature space as well as preventing overfitting. This reduction is similar to our own use of GentleBoost classifiers, which also selects the most relevant feature vector dimensions, but the SBL framework remains fully probabilistic.

Meyer-Delius et al. [67] propose a model for spatio-temporal situations, consisting of an HMM operating on relational descriptions, which in turn are the estimated states of dynamical systems: Observations are integrated by dynamical systems, their estimated states are abstracted to relational descriptions, and hidden states in a Markov Chain are connected to these relational descriptions. The temporal evolution of the agent and its surroundings is modeled by a standard DBN [21], consisting of random variables for the internal state of the agent and the observable data at each timestep, and chained together by conditional probability tables (CPTs). The relational layer on top of the DBN abstracts from the often continuous observation data (e.g., vehicle distances), and on top of the relational layer we have multiple parallel models that are being tracked, representing the situation types (driving maneuvers) that are possible for the agent in its current state. Maneuver recognition involves arbitration between the concurrent HMMs, which is done by computing their Bayes factors [48], a quantitative measure for determining which of two models explains the observed data best.

A similar model of situations is presented in [66], consisting of an observation layer with a corresponding temporal chain of system states on top, and a number of parallely tracked HMMs on top of that. DBNs are used to represent and arbitrate between the possible driving maneuvers. The most notable difference is the missing of an intermediate relational abstraction layer, but the observations on which the HMMs operate are still relations to the other vehicles. The efficiency of the HMM framework gives rise to online recognition. More specifically, the model consists of a lower-level, more fine-grained state space model, realized by a DBN that captures the interactions of agents in the scene where the state is estimated by standard recursive estimation [92], as well as a range of more coarse-grained situation models that are tracked in parallel and implemented by HMMs with Baum-Welch parameter estimation [24]. Model evaluation for recognition, given the observed sequence of states, is performed by computing the model likelihoods via standard forward-procedure and then comparing the resulting posterior odds [87]. The real-world experiments are mostly conducted on highways, where the observations are provided by SICK laser scanners and the vehicle positions are automatically tracked, after manual initialization. However, only following and passing maneuvers are considered in the evaluation.

Gindele et al. [38] present a complex DBN for modeling various traffic-related variables, ranging from metric sensor data over more symbolic behaviors of others to their most likely (metric) trajectories. The focus is on modeling the interactions between vehicles, and the experiments are conducted on a simulated highway using four different maneuvers. The DBN is justified in that most of the relevant data cannot be observed (e.g., vehicle distances, but not the future trajectories), and hence need to be inferred. Also, low-level metric aspects and high-level symbolic aspects can thus be combined in a single framework, and uncertainty as well as sensor noise are properly dealt with. Reasonable conditional independence assumptions between the random variables lead to a "sparse" DBN [74], and it is further assumed that the Markov property holds, i.e., the current state contains all relevant information for estimating the next state. The joint distribution of the DBN is thus decomposed into factors that are regularly updated by applying Bayes' theorem, but no analytical solution exists and the posterior distribution is therefore approximated by means of a particle filter [103].

Lefevre et al. [55] address the problem of maneuver recognition at intersections from vehicle data, as in earlier approaches (e.g., Kuge et al. [50]), but use map data of the roads and lanes. The intersection and the maneuvers are modeled in a probabilistic framework, evaluated on real intersection data, and the Bayesian Network comprises the following random variables: entrance road, entrance lane, exit lane, vehicle path, and turn signals. A rule-based algorithm for computing the CPTs makes the model applicable to any intersection. Inference mainly concerns the lanes taken by the vehicle, which cannot be measured accurately enough (otherwise, lanes and path of the vehicle were evident). Discrete probability distributions are used by the model, contributing to its computational efficiency.

Finally, Agamennoni et al. [2] present a Bayesian approach to driving behavior prediction formulated as stochastic filtering, where the probabilistic approach combines low-level metric observations with high-level symbolic representations and handles the uncertainty. The study focuses on the interaction of two mining vehicles at an intersection with real-world data. Like its most related work [38], the method employs stochastic filtering to estimate the vehicle trajectories. The model is an instance of recursive Bayesian filtering over discretized timesteps, modeling the dynamics of each vehicle and linking the individual models by a context layer on top. The GPS positions of the vehicles are observed, and the vehicle dynamics on top comprise the vehicles' pose, velocity, and steering angle. The context model consists of a term per agent that ensures temporal coherence over time, and a common term over all agents that establishes the inter-vehicle relations. As the model is highly non-linear, variational approximation of the joint posterior distribution is performed rather than exact inference [5, 46].

While the above traffic situation models enable the inference of appropriate driving behavior from the scene content itself, going beyond the actual behavior of the human driver and thus being more directly related to our own approach (see Chapter 3), they are also far more complex than necessary for our purposes. As generative probabilistic models, they contain a wide range of random variables to cover many different aspects about traffic scenes and enable complex reasoning about future timesteps and trajectories, for example. However, we only require behavioral decisions about the appropriate driving behavior at each point in time, which can be obtained more efficiently by learning direct correlations.

### 2.2.3 Direct Correlation Learning

Vidugiriene et al. [120] investigate whether or not similar settings can be used for different drivers. Their goal is to predict the appropriate steering angle from visual data, recorded by a camera while driving on rural roads (like Pugeault [85]), with four lane marking based features as input [22, 23] and a neural network or look-up table as predictive model. Apart from the visual data, CAN data of two different drivers is recorded, with a driving time of about 6 minutes each. The neural network has two layers with only two neurons in the hidden layer, the input is smoothed by a moving window low-pass filtering, and the output is stabilized by averaging the predicted signal over 10 initializations. The look-up table just stores the observed features with their corresponding steering angles in training, and then retrieves the MSE-closest feature values for testing. Both models perform continuous regression, and their predictions are compared to the actual steering angles of the two drivers. Several combinations of the four input features are tested, showing that different feature combinations are optimal for the two drivers. Also, the neural network performs better than the look-up table (1st driver: MSE about 4 – 7 % compared to 7 – 20 %, 2nd driver: MSE about 3 – 5 % compared to 5 – 8 %). Training on one driver and testing on the other leads to the same general pattern, thus confirming the above results.

Garcia Ortiz et al. [79] address driving behavior prediction in the near future, from current and past observations of the vehicle's physical data and selected environmental information. Approaching a traffic light at an intersection or junction is considered, where the state of the traffic light (3 values) as well as distance and velocity of the car (2 values) serve as input features. While the visual features are manually annotated, the behavior data stream is automatically segmented into "behavior primitives" ("stopped", "braking", and "other"). Low-dimensional representations are used, unlike Pugeault [84], Heracles [43], and McCall [64, 65] who operate on high-dimensional input feature vectors that are later reduced to their most relevant dimensions. The problem is cast as multi-class classification like in our previous work [43], but using a Multi-Layer Perceptron (MLP) [41] with backpropagation [89] for correlation learning. The focus is on prediction into the future by simply shifting the target vectors, and a variance-based confidence is also introduced. ROC curves are used for the evaluation, as in Pugeault [84] and McCall [65], annotated with different rejection rates and showing that increasing the confidence threshold improves prediction but then decreases again, with an optimal rejection rate of about 10 %. The 10 % ROC curves for different time-scales show good performance up to 3 seconds (EER: 85 % to 75 %).

Maye et al. [63] address the learning of situation types and typical actions of the driver, from Inertial Measurement Unit (IMU) data and video data. First, the IMU data is cut into motion segments by a probabilistic Bayesian approach to change-point detection [1, 32], with a particle filter implementation for efficiency. Second, the image data is represented in a Bag-of-Words (BoW) scheme [97] by Difference-of-Gaussian keypoints and SIFT descriptors [59], where Dirichlet Compound Multinomials (DCM) represent the final codeword histograms. Third, the motion segments are clustered into "situation types" by visual similarity of their corresponding BoW representations, thus leading to situation labels (e.g., "approaching a red traffic light"). As the situation labels are associated with the original motion data, typical vehicle motions (driver's actions) are also learned (e.g., "braking"). Importantly, no manual interaction or labeling is required at any point, and the learning process happens online while driving. Cast as probabilistic filtering, the joint distribution over motion segment length (online change-point detection), situation label (by image data), and associated action (appropriate driving behavior) is split in three conditionally independent models: The motion segmentation is done automatically by inferring at each timestep whether or not a new motion segment starts, given the observed IMU data, where the posterior distribution is approximated by a particle filter with Rao-Blackwellization [19] for lower variance. The current situation label is determined by Bayesian reasoning over the existing situation types, given the observations. At each timestep, the Bayes factors [48] between the existing situation models are computed, and a confidence threshold indicates whether or not an additional model should be added for the current situation. The possible actions of each situation type are modeled by a Gaussian Mixture Model (GMM), to account for multiple possibilities. The evaluation is done on simulated and real urban data, but in a repeating loop only. Motion segmentation and situation labeling achieve over 90 % accuracy, while the action prediction performs "accurately".

Importantly, Pugeault et al. [85] learn correlations between holistic descriptors of visual traffic scenes and the observed behavior of a human driver. Afterwards, the system can infer the appropriate behavior from a visual description of such traffic scenes alone. In contrast to Heracles [43] and Pugeault [84], their study concentrates on the steering angle only, but learns a continuous regression by means of a random forest (RF) with an extension called "Medoid-RF", which performs better for extreme but correct steering angles. Several image-filter based visual representations are considered: standard GIST, kernel-GIST (see Pugeault [84]), and HOG features. The system is evaluated by autonomous steering of a robot following an indoor track, and by predicting the behavior of a human driver in a rural road-following scenario, showing the kernel-GIST to perform best. The GIST descriptor is the same 2048-dimensional feature vector as in [84], computed at 4 scales and 8 orientations from a 128 x 128 image. For kernel-GIST, the uniform averaging within each grid cell is replaced by a Gaussian kernel weighted version, without an additional layer here. The HOG features are computed directly from the gradient images, again within grid cells but without convolutions, at four scales and eight orientation bins. The random forest [11] is an ensemble classifier similar to the GentleBoost framework, but large trees are used instead of stumps. As the original formulation uses the arithmetic mean for leaf node computations and for the entire forest, which cannot handle outliers and extreme but correct steering angle values in the training data, the median is used instead. The first scenario (indoor track) has sharp 90° curves with clear lane markings, and a remote controlled car is operated by a human to collect training data, 10% of which are randomly sampled for training while the rest is used for testing. The best results (Medoid-RF) have an average angle deviation of about 13° (both for GIST and C-GIST), which is sufficient for autonomous road following. The second dataset (rural road) was recorded while driving without clear lane markings and changing road appearance, again using 10 percent of the data for training and the rest being used for the testing. Here, the angle deviation was approximately 7 degrees (both GIST and C-GIST), which is surprisingly lower. Using a scene categorization approach as well, the work of Pugeault [84] is the most related work to our own. We thus discuss their techniques in greater detail throughout this thesis, and compare our behavior prediction results to theirs.

## 2.3 Visual Scene Understanding

In the following, we provide an overview of techniques that have been developed in the fields of image segmentation (see Section 2.3.1) and scene categorization (see Section 2.3.2). These are relevant for our own scene categorization approach to driving behavior prediction (see Chapter 3), and for our semantic object-level representation of urban traffic scenes (see Chapter 4), respectively. Readers familiar with these domains, however, may directly continue with Chapter 3.

### 2.3.1 Image Segmentation

Segmentation deals with segregating semantic entities such as objects of interest from their surrounding background that is considered irrelevant for the task at hand. It is important to note that in segmentation, one is interested in an accurate description of the exact object boundaries, going beyond what is required for object detection in general, where it is sufficient to approximate the object location by a coarse bounding box (e.g., [29]). The challenge arises from the fact that semantic aspects defining objects of interest are difficult to incorporate into mathematical algorithms, as they involve higher cognitive interpretation skills in humans, and from the requirement of robustness to the often considerable variations in visual appearance, particularly in the case of real-world images.

**Single objects.** Early segmentation approaches mostly dealt with individual objects of interest. Some of them required additional knowledge in form of human interaction. Such interaction could, for example, consists in drawing a coarse bounding box around the object of interest, as in *OBJ-Cut* or *GrabCut* [91], which helps the algorithm to determine which image features can be considered as belonging mostly to the object of interest (within the bounding box) and which are belonging to the background (those outside of the bounding box). Also, initializing an active contour model by manually placing it inside an image, preferrably close to the actual boundaries of the object to be segmented, is an example of user-input to guide the segmentation process [47]. In active contour models, the initial contour then adapts to the actual object contours by following the image gradient, for example, while at the same time maintaining physical constraints that are incorporated in the flexibility of the active contour itself.

**Graph cuts.** Graph cuts, the underlying technique of *OBJ-Cut* and *GrabCut*, became a popular segmentation framework because they enabled the propagation of local evidence based on image features across the image domain, and they essentially operate on energy functions that are minimized such that the resulting segmentation violates as little of the evidence as possible. The original graph cuts were applied to foreground/background segmentation, which is consistent with the user-aided scenario of individual objects as outlined above. In fact, however, graph cuts can be applied to segmenting an image into multiple regions, not just an object of interest. This is an important transition in segmentation that was made possible by the use of *alpha-expansions* [8], which extend the binary graph cuts to multiple labels. While the original graph cuts are exact methods, using alpha-expansions for the multi-case is approximative only. Nevertheless, the approach works well in practice and is efficient to compute.

**Random Field models.** Also note that the use of graph cuts, in which each pixel in the image is modeled by a node in a rectangular, regular graph and where neighboring pixels are connected by an edge, is the beginning of understanding segmentation as a labeling problem. Each node is assigned one out of several possible labels, two in the binary case and more in the extended case, and the goal of optimization is that in the end the labeling is consistent with the objects to be segmented, essentially forming regions in the image as defined by connected pixels of the same label. From this graph-theoretic modeling framework, several important techniques arise, including *Markov Random Fields* and *Conditional Random Fields* [52]. While the former is a generative model that jointly represents the evidence as given by the image as well as the respective pixel labels, and from which a labeling can be inferred by using the Bayes theorem, the latter are discriminative models that directly represent the posterior distribution over the labels. Since in segmentation, one is usually only interested in the labeling itself, Conditional Random Fields have been the preferred method, and they have been continuously developed further in recent years.

**Energy potentials.** Improvements on Conditional Random Fields have addressed their underlying graph structure, in an attempt to enable more long-distance propagation of local evidence across the image, which is limited in a standard grid, and also their energy potentials that govern the pixel label assignments. Typical potentials include a unary potential, modeling the relationship between the local evidence at a pixel (typically by considering the surrounding patch) and the most likely pixel label, as well as a pairwise potential that models the relationship between adjacent pixel labels. The unary potentials are where discriminative models can be used, as they are given by the scores of classifiers that operate at the patch level, whereas the pairwise potentials typically employ a contrast-sensitive Potts model to ensure that neighboring labels tend to be equal unless there is sufficient contrast between the corresponding pixels [8]. The intuition is that the unary potentials determine the most likely labels based on the visual appearance of the patches (e.g., patches recognized as belonging to a car would give rise to "car" pixel labels), while the pairwise potentials tend to spread this evidence (i.e., increasing the likelihood of nearby pixels being labeled as "car", too), unless there is an edge that supports an object boundary.

**Higher-order potentials.** Extensions of the potentials include the introduction of a third type, called *higher-order potentials*, that take into account region information from an unsupervised segmentation of the image [99]. Unsupervised segmentations may be obtained by the *watershed*, *mean-shift*, or some other suitable method. The region information is used to impose constraints on the labels within each region, such that the cost of assigning uniform labels within a region is "cheaper" than assigning different labels. The optimal labeling therefore tends to respect the boundaries of the (unsupervisedly obtained) regions, which in turn are an oversegmentation of the image that respects the actual object boundaries. In particular, higher-order potentials improve the segmentation of thin objects, as these often are preserved in an unsupervised oversegmentation and would otherwise get "propagated over" by the labels of adjacent locations with strong evidence, in a standard Conditional Random Field. Another extension is the introduction of temporal potentials [123], which make use of the coherence between subsequent frames if the images are taken from a video sequence.

**Texture features.** As for the features used to represent and classify the patches sampled from an image, as required by the patch classifiers implementing the unary potentials, the possibilities include histograms of oriented gradients, blob information, color, and texture. Particularly the latter has been shown to be highly useful, being the basis of the well-known *TextonBoost* segmentation approach [96]. In this paradigm, textures are being learned from training images by sampling patches, which are then represented in terms of the output of a filter bank that typically includes oriented edge and blob filters at multiple scales, and finally clustered to form *textons*. These textons are similar to codewords as used in image representations for scene categorization, as we will see later, and new images can subsequently be represented in terms of these textons. One of the most important techniques in the context of TextonBoost is the use of shape filters, which go beyond mere patch classification and instead take into account typical neighborhood relations between textons. As an example, pixels that belong to a car are therefore not only labeled as such because of a car-like visual appearance of the patch located at these pixels, but also because of the visual appearance of patches below that are classified as belonging to the road.

**3D features.** A fundamentally different type of features is 3D information about the scene, which has also been shown to be useful for segmentation. While still images require stereo cameras (or extensions thereof) in order to reconstruct 3D information, and monocular approaches may infer 3D information based on reasonable assumptions about a scene [44], sparse 3D information can also be reconstructed by structure-from-motion techniques [40]. 3D features for segmentation include the height of points above the camera or ground, their distance from the camera trajectory, the reprojection error that highlights moving objects, the density of points that correlates with the amount of texture an object has, and coarse estimations of local surface normals [15]. Not only can these features be used to yield a basic segmentation of a scene, but they are also complementary to appearance-based features such as textons, so their combination leads to a higher segmentation accuracy than using either of the two alone.

State-of-the-art Conditional Random Field models for segmentation integrate unary, binary, and higher-order potentials, and combine 3D features with textons, histograms of oriented gradients, and color information. Importantly, such models have recently been applied to the segmentation of urban traffic scenes [99]. We thus observe not only a transition from the early object-oriented segmentation to more holistic scene decomposition, but also from arbitrary landscapes to rural and urban traffic scenes, among others. This reflects the importance of traffic scene understanding and its growing relevance for the computer vision community.

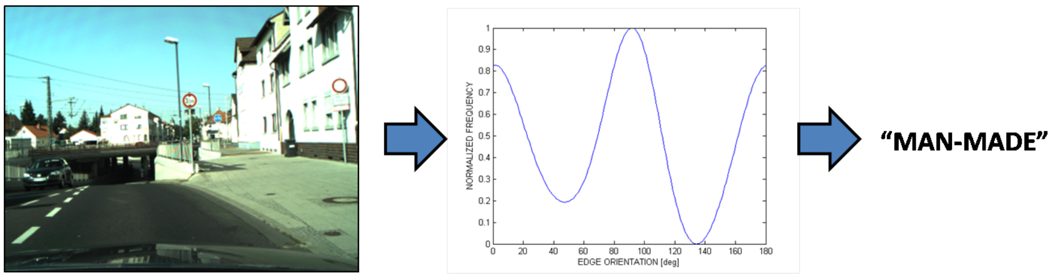


Figure 2.5: Example of scene categorization with a global image histogram.

### 2.3.2 Scene Categorization

The goal of scene categorization is to decide for a given image which of several possible categories it belongs to, based on its visual content. For example, an image depicting cars, pedestrians and roads might be categorized as "city", whereas an image depicting sand, water and boats might be categorized as "beach". The available categories are typically defined in advance, and the methods involve a training stage in which they are given several example images of each category. As we will see, the crucial aspect is how the images are represented such that different categories can be easily distinguished from each other while at the same time preserving the similarity of images falling into the same category. As with object recognition, the considerable intra-class variations are particularly challenging.

**Global image histograms.** The simplest way to represent an image is to consider it as a whole, without explicitly accounting for the fact that the image might actually be composed of semantically different regions or objects. In this vein, the earliest approaches to scene categorization begin by computing global image statistics [116, 117], such as the orientations of edges. By discretizing the possible edge orientations, the image can then be represented by a histogram that counts for each edge orientation how frequently it occurs in the image. Such histograms can be used to distinguish between images falling into categories that are characterized by fundamentally different types of edge orientations, such as "man-made" environments with buildings (characterized by frequent vertical edges that correspond to walls) and "natural" environments with vegetation (characterized by more randomly oriented edges corresponding to branches, see Figure 2.5).

**Image sub-block histograms.** An improvement over the global image histograms is to break an image into multiple sub-blocks [95, 100], computing a histogram over the statistics within each sub-block only. This way, information from different regions within the field of view is kept separate from each other, giving the approach more robustness to misclassifications in individual subblocks of the image that might look like they belonged to another category. By individually categorizing each sub-block based on its respective histogram, and performing a majority vote on the category of the image as a whole based on the individual sub-block categorizations, local ambiguities do not affect the overall categorization. In contrast, a global image histogram would inevitably be altered as a whole, even if the reason is confined to a single sub-block only. Another potential advantage of sub-block representations is that some spatial information about the image features is preserved. However, this property was not used at this early stage of scene categorization and introduced separately later on.

**Bag-of-words representation.** Taking the idea of dividing an image into sub-blocks to the limit, we end up at the level of patches. In this paradigm [81, 97], small image regions are sampled either randomly across the image or densely along a regular grid, and each patch is represented by some feature descriptor, for example, edge orientations and magnitude within the patch, yielding a vector representation. In training, these feature vectors are condensed into *codewords* by clustering the feature vectors in feature space, typically by using k-means. New images can then be represented in terms of how similar the patches sampled from these image are in comparison to the learned codewords. As the codewords are learned beforehand their number and types are fixed, so each new image can therefore be expressed by a histogram over these codewords, where a high value of a bin means that the image contains many patches that are similar to the respective codeword. Decision boundaries can then be learned between the different image categories as before, by operating on the resulting histograms over codewords (or "bags of words"). The advantage of this approach over the previously discussed histograms is that the codewords are not the result of some pre-defined discretization of the feature space but are actually learned from the training data. Hence, the codewords provide a better coverage of the feature space and thus lead to improved classification in the given application domain.

**Bag-of-topics representation.** An important improvement over the raw Bag-of-Words representation that operates at the level of the codewords consists in creating an additional layer of abstraction between the words and the images. As the clustering process that gives rise to the codewords is unsupervised, it frequently happens that different codewords might actually contain similar information that would better be assigned to a single codeword. This is known as *synonymity* of codewords, meaning that several codewords actually refer to the same visual content. Also, a single codeword might represent more than one type of image patches, simply because these lie close to each other in feature space, although humans would tend to represent these patches by different codewords. This is referred to as *polysemy* of codewords, where a codeword does not form only one semantic entity. Topics now "group" multiple of the learned codewords together, to form a new entity at a somewhat higher level of abstraction. For example, patches corresponding to eyes, noses, and mouths could be grouped to a "face" topic, if this is supported by the actual scene categorization task at hand. While topics do not overcome the polysemy of codewords, which would involve splitting them, they improve on the synonymity. Thus, bag-of-topics representations [7, 33, 86] generally perform better than bag-of-words representations. Also, dense patch sampling is superior to random sampling, as it yields more data.

**Spatial Pyramid matching.** Spatial Pyramids [54] counter the most severe drawback of the previous bag representations, namely, their lack of capturing spatial information about the underlying patches. In fact, standard bag representations are unable to distinguish between images consisting of similar patches at completely different locations, as this information is lost in the histograms. To compensate, Spatial Pyramids apply the Bag-of-Words (or Bag-of-Topics) procedure at multiple levels, dividing the image into sub-blocks: Level 0 considers the image as a whole, hence is identical to the standard bag representation, level 1 divides the image into four blocks each of which is represented as a bag of its own, level 2 divides the image into 16 blocks, etc. The scheme is similar to image sub-block histograms as discussed before, but multiple layers are used, and bag representations are computed rather than histograms. Spatial pyramids are an extension of the Pyramid Match Kernel proposed earlier for efficient similarity computation between sets of features.

**Kernel codebooks.** Another improvement of bag representations addresses the way the histograms are formed. The original approach makes hard assignments between image patches and codewords, using a nearest neighbor scheme that gives equal weight to each patch in the resulting histogram. However, *how* similar the patches are is disregarded ("codeword uncertainty"), as well as how *uniquely* a patch is explained by its codeword, which concerns patches lying "between" codewords ("codeword ambiguity"). The improvement consists in replacing the nearest-neighbor paradigm by considering Gaussian kernels at the codeword locations [118], thus weighting the influence of each patch on the resulting histogram by its proximity to the codewords (*all* codewords, that is, but distant codewords have negligible weight). This deals with both codeword uncertainty and ambiguity, and the latter leads to increased performance. The former (i.e., effectively *not* assigning a codeword at all if a patch is too far away) does not, as it is better to assign some approximate codeword rather than none.

**Spatial region constraints.** The previously discussed approaches did not go beyond the domain of patches. However, regions in an image that share some appearance-based property such as color, texture, or the like can be used to impose constraints on the codewords that can be assigned to patches sampled from within the same region. The underlying assumption is that patches sampled from the same region are likely to belong to the same codeword or topic. Taking this constraint into account, and operating at the level of regions as well as patches, enables the method to not only categorize an image but also segment its "topics" in accordance to the region boundaries [18]. This represents a joint categorization and segmentation framework, and recent extensions add the ability to automatically annotate the segments with learned object labels [57, 121].

After this overview of the developments in visual scene categorization over the recent years, we now proceed to develop our own account of scene categorization in a behavior-driven system architecture (see Chapter 3). The segmentation techniques presented here, in turn, will form an important building block for our semantic object-level representation of urban traffic scenes, particularly regarding the Conditional Random Field framework (see Chapter 4). Finally, our behavior prediction experiments (see Chapter 5) involve a detailed performance analysis and comparison to the most related work [84] as identified in this chapter.

# Chapter 3

# Behavior Prediction Framework

In this chapter, we detail our system architecture for learning correlations between visual traffic scenes on the one hand and the typical behavior of a human driver on the other hand. At the heart of our approach lies the assumption that the behavior of the driver has an observable reason in the given traffic situation, such as braking due to a car in front or steering because of a curve. It does not incorporate behavior of non-visible origin, like driving slowly because the driver is unfamiliar with the current environment or steering due to a parking maneuver.

We begin by showing that the problem can be cast as scene categorization, where the continuous behavior data is discretized according to the requirements of the intended application. This discretization gives rise to behavior classes that, in turn, induce visual categories on the corresponding images of traffic situations, serving as training examples for our system. By learning discriminative models of these visual categories, in a training phase prior to the actual application, our system is then able to predict the appropriate behavior class for new images depicting traffic situations while driving, employing a winner-take-all scheme.

Our behavior-driven approach represents an adequate solution to the challenge of identifying visually dissimilar but behaviorally equivalent traffic scenes as such. The proposed subdivision of the behavior space into discrete classes also preserves the ability to approximate the continuous-valued behavior data to an arbitrarily fine-grained degree. We further discuss practical aspects such as the acquisition of ground truth data, temporal stabilization for robustness, and confidence measures to automatically detect and react to ambiguous traffic situations while driving.

## 3.1 Problem Formulation

To start with, we formally define the available data streams that serve as input to our system architecture. These include a visual data stream $I$ that consists of images acquired by a car-mounted camera, monitoring the traffic scene in front, and a behavioral data stream $B$ that consists of physical quantities about the moving vehicle, measured by the CAN bus while driving. In a training phase, the system observes both data streams to learn typical correlations between the visual and the behavioral domain (see Figure 3.1), and subsequently should be able to infer the appropriate driving behavior from the visual scene content alone. In the following, we show how to solve this problem by scene categorization.

### 3.1.1 Input Data

The visual data stream $I = (i^{(1)}, \dots, i^{(T)})$ consists of $T \in \mathbb{N}$ stereo image pairs $i^{(t)} = (i^{(t)}_{\text{left}}, i^{(t)}_{\text{right}})$, acquired by the left and right camera of a calibrated stereo rig, respectively. More specifically, the stereo camera provides color images $i^{(t)}_{\text{left,right}} : W \times H \to \{0, \dots, 255\}^3$ in RGB color space, where $t \in \{1, \dots, T\}$ is a timestamp and $W, H \in \mathbb{N}$ denote the image width and height, respectively. As an alternative, a monocular camera could be used instead, leading to $i^{(t)}_{\text{left}} = i^{(t)}_{\text{right}} = i^{(t)}$, and if the camera provides grayscale images only, we have $i^{(t)} : W \times H \to \{0, \dots, 255\}$. However, color information and stereo disparities arguably represent valuable cues for visual traffic scene understanding, and we therefore continue to develop our behavior prediction framework in its full generality as originally stated above.

The behavior data stream $B = (b^{(1)}, \dots, b^{(T)})$ consists of $T \in \mathbb{N}$ state vectors $b^{(t)} = (q^{(t)}_1, \dots, q^{(t)}_D)$, where each of the $D \in \mathbb{N}$ behavioral quantities $q_1, \dots, q_D \in \mathbb{R}$ is a potentially continuous-valued measurement describing some physical property of the vehicle at timestep $t \in \{1, \dots, T\}$. Such measurements can be obtained directly from the CAN bus of the moving vehicle, which is arguably the easiest and most complete way, or from separate sensors like inertial measurement units, and even from the visual data stream $I$ itself by using structure-from-motion techniques for pose estimation. In practice, however, we found the CAN bus to be the most reliable solution, which is why we continue to focus on the measurements that it provides, assuming that $B$ and $I$ are synchronized by the timestamps $t$.

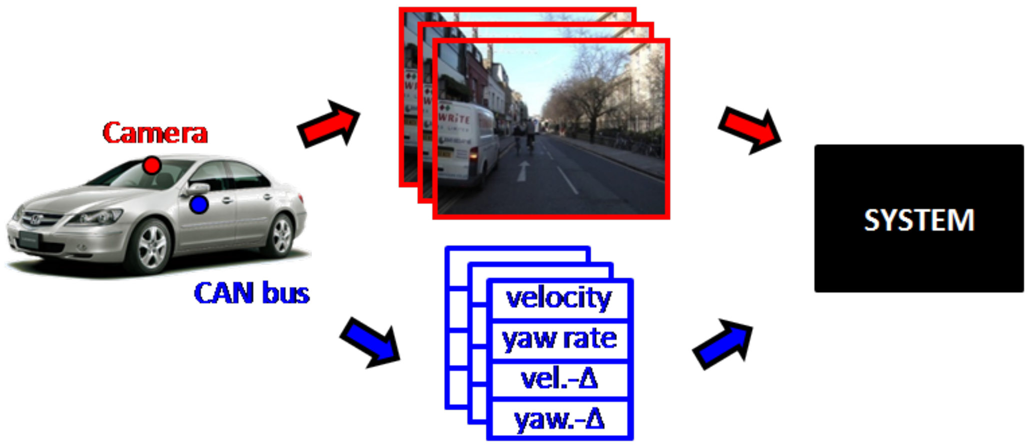


Figure 3.1: The input data consists of a visual stream (top), acquired by a car-mounted camera that observes the traffic scene in front, and a behavioral stream (bottom) that consists of various physical quantities about the moving vehicle.

It is worth spending a moment to examine the range of behavioral quantities provided by the CAN bus, to determine which of them are relevant for us and which are not. A closer look reveals that they can be organized in a hierarchy from the driver to the ego-vehicle, and to other cars in the scene (see Figure 3.2). From the perspective of autonomous driving, it would be sufficient to focus on the third group alone, consisting of the velocity $q_{\text{vel}}^{(t)} = v^{(t)} \in [-V, +V]$ and the yaw rate $q_{\text{yaw}}^{(t)} = y^{(t)} \in [-Y, +Y]$ at any given point in time $t \in \{1, \ldots, T\}$, where $V, Y \in \mathbb{R}_{>0}$ denote the maximum possible velocity and yaw rate, respectively: If we were able to correctly predict the velocity and yaw rate a human driver would maintain in all traffic situations $i^{(t)}$, this would be sufficient for the car to reactively follow the road by steering, avoiding obstacles, etc. The quantities in the second group could be computed from $v^{(t)}$ and $y^{(t)}$ as well, because the lateral acceleration is given by the derivative of the yaw rate, and the longitudinal acceleration is given by the derivative of the velocity of the car. Also, quantities of the first group such as the steering angle and the pedal status could be inferred, as the steering angle is directly related to the yaw rate and the pedal status affects the velocity. Overall, we conclude that the hierarchy is largely redundant, and it is basically our choice which quantities to consider. As we are ultimately interested in the effects on the ego-vehicle, we choose to predict $b^{(t)} = (v^{(t)}, y^{(t)})$.

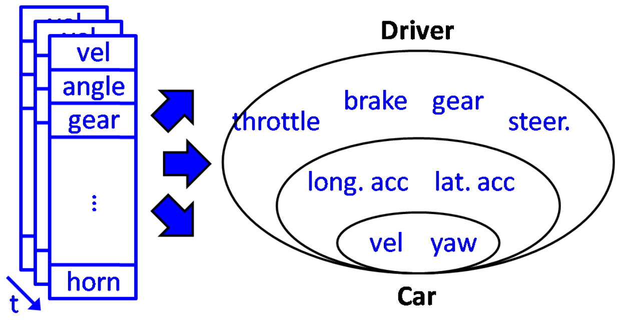


Figure 3.2: The physical quantities of the CAN bus actually form a hierarchy, leading from the driver to the ego-vehicle and to other cars in the scene. Note that most actions of the driver ultimately affect the course of the ego-vehicle.

### 3.1.2 Behavior Classes

Given the visual data stream $I = (i^{(1)}, \dots, i^{(T)})$ and the behavioral data stream $B = (b^{(1)}, \dots, b^{(T)})$, our goal is to have a technical system learn to predict the appropriate velocity and yaw rate $b^{(t^*)} = (v^{(t^*)}, y^{(t^*)})$ for traffic situations at some future timestep $t^* \in \mathbb{N}$ with $t^* > T$, solely based on the visual scene content of the corresponding camera images $i^{(t^*)} = (i^{(t^*)}_{\text{left}}, i^{(t^*)}_{\text{right}})$. As both velocity and yaw rate are continuous-valued quantities $v^{(t)} \in [-V, +V] \subseteq \mathbb{R}$ and $y^{(t)} \in [-Y, +Y] \subseteq \mathbb{R}$, respectively, when taken together they span a two-dimensional behavior space $\mathcal{B} = [-Y, +Y] \times [-V, +V] \subseteq \mathbb{R}^2$ (see Figure 3.3, left). While the learning problem is therefore an instance of continuous regression, it is worth spending a moment to judge whether this granularity is appropriate for the intended applications.

While it appears beneficial for autonomous navigation to be able to predict the exact values $b^{(t^*)} = (v^{(t^*)}, y^{(t^*)})$ at any given timestep $t^* > T$, as these result in a smooth trajectory of the vehicle when used to generate motor commands, this is not true for driver assistance: Here, qualitative predictions are more useful, such as telling the driver that he or she should drive slowly in a certain situation, or that gentle steering in a particular direction is required now, whereas providing the exact values (like steering $y^{(t^*)} = 23.5°$ to the right or keeping a velocity of $v^{(t^*)} = 27.1$ km/h) would be unnecessary and even distracting. For this reason, we discretize the behavior space into disjoint behavior classes $\mathcal{B}_1, \dots, \mathcal{B}_M \subseteq \mathcal{B}$ such that $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for all $i, j \in \{1, \dots, M\}$ with $i \neq j$, and $M \in \mathbb{N}$.

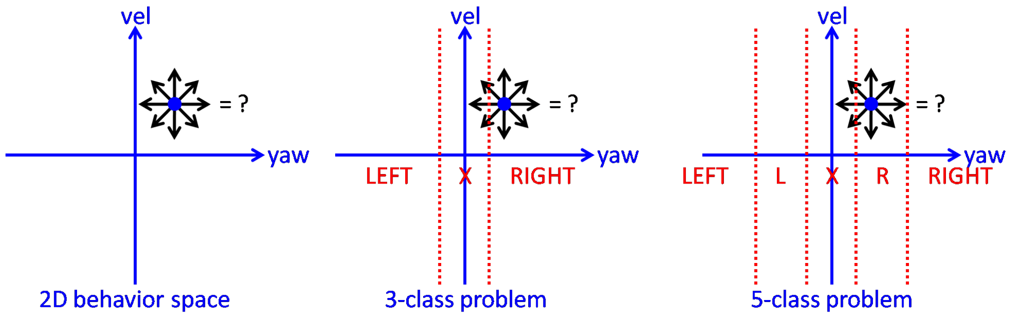


Figure 3.3: The continuous behavior space spanned by velocity and yaw rate (left) is discretized into semantically meaningful behavior classes (center). Depending on the application, this discretization can be arbitrarily fine-grained (right).

One way to define such behavior classes $\mathcal{B}_1, \dots, \mathcal{B}_M$ is by imposing thresholds $\tau_1, \dots, \tau_{M-1} \in \mathbb{R}$ on the continuous domains of velocity and yaw rate, respectively. Assuming $\tau_m < \tau_{m+1}$ for all $m \in \{1, \dots, M-2\}$, each $\tau_m$ effectively represents the boundary between adjacent behavior classes $\mathcal{B}_m$ and $\mathcal{B}_{m+1}$. More specifically, each behavior class is then given by $\mathcal{B}_m^{(d)} = \mathcal{B}_{q_d \le \tau_m} \setminus \mathcal{B}_{m-1}^{(d)}$ for $m \in \{1, \dots, M-1\}$, with $\mathcal{B}_0^{(d)} = \emptyset$, $\mathcal{B}_M^{(d)} = \mathcal{B} \setminus \mathcal{B}_{M-1}^{(d)}$, and $d \in \{1, \dots, D\}$ denoting the dimension. For example, the yaw rate domain could be subdivided by thresholds $\tau_1 \in \mathbb{R}_{<0}$ and $\tau_2 \in \mathbb{R}_{>0}$, leading to behavior classes $\mathcal{B}_1^{\text{yaw}} = \mathbb{B}_{y \le \tau_1}$, $\mathcal{B}_2^{\text{yaw}} = \mathbb{B}_{y \le \tau_2} \setminus \mathbb{B}_1^{\text{yaw}}$, and $\mathbb{B}_3^{\text{yaw}} = \mathbb{B} \setminus \mathbb{B}_2^{yaw}$ that correspond to steering left, straight, and right, respectively (see Figure 3.3, center). Despite the coarse granularity of these behavior classes, they are already useful for potential applications in driver assistance.

Note that the proposed subdivision of the behavior space is also suitable for autonomous navigation: While the above 3-class subdivision would result in a considerably non-smooth trajectory if used to control a real vehicle, more fine-grained subdivisions could be used instead. For example, a 5-class subdivision by thresholds $\tau_1 < \tau_2 \in \mathbb{R}_{<0}$ and $\tau_3 < \tau_4 \in \mathbb{R}_{>0}$ gives rise to behavior classes $\mathcal{B}_1^{\text{yaw}} = \mathcal{B}_{y \le \tau_1}$, $\mathcal{B}_2^{\text{yaw}} = \mathcal{B}_{y \le \tau_2} \setminus \mathbb{B}_1^{\text{yaw}}$, $\mathcal{B}_3^{\text{yaw}} = \mathcal{B}_{y \le \tau_3} \setminus \mathbb{B}_2^{\text{yaw}}$, $\mathcal{B}_4^{\text{yaw}} = \mathcal{B}_{y \le \tau_4} \setminus \mathbb{B}_3^{\text{yaw}}$, and $\mathcal{B}_5^{\text{yaw}} = \mathcal{B} \setminus \mathbb{B}_4^{\text{yaw}}$ that correspond to steering strongly left, slightly left, straight, slightly right, and strongly right, respectively (see Figure 3.3, right). Thus, the discretization can be chosen to match the application, with coarser granularity for driver assistance and more fine-grained granularity for autonomous navigation.

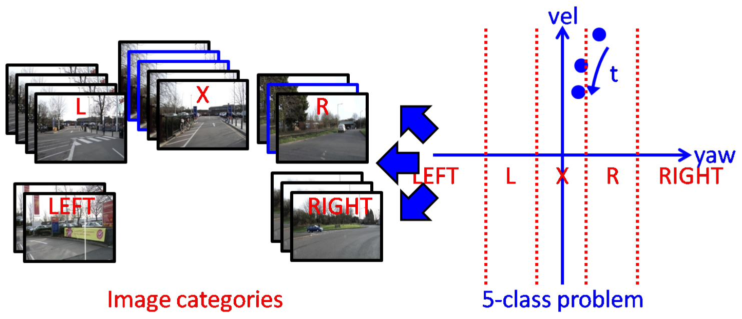


Figure 3.4: As the observed measurements of velocity and yaw rate fall into different behavior classes (right), the behavior stream is effectively segmented into batches whose corresponding images give rise to visual categories (left).

### 3.1.3 Image Categories

We assume that we have a subdivision of the continuous-valued behavior space $\mathcal{B} = [-Y, +Y] \times [-V, +V] \subseteq \mathbb{R}^2$ into discrete behavior classes $\mathcal{B}_1^{\text{vel}}, \ldots, \mathcal{B}_M^{\text{vel}}$ and $\mathcal{B}_1^{\text{yaw}}, \ldots, \mathcal{B}_N^{\text{yaw}}$. These behavior classes effectively segment the behavior stream $B = (b^{(1)}, \ldots, b^{(T)})$ into batches, as each $b^{(t)} \in B$ is also an element $(v^{(t)}, y^{(t)}) \in \mathcal{B}$ that falls into exactly one of the behavior classes $\mathcal{B}_1^{\text{vel}}, \ldots, \mathcal{B}_M^{\text{vel}}$ and $\mathcal{B}_1^{\text{yaw}}, \ldots, \mathcal{B}_N^{\text{yaw}}$, respectively. Abstracting from the temporal order within $B$, we therefore have corresponding sets $B_1^{\text{vel}}, \ldots, B_M^{\text{vel}}$ and $B_1^{\text{yaw}}, \ldots, B_N^{\text{yaw}}$ that contain all the $b^{(t)} \in B$ belonging to the same behavior class, i.e., $B_i^{(d)} = \{b^{(t)} \in B | (v^{(t)}, y^{(t)}) \in \mathcal{B}_i^{(d)}\}$ for $i \in \{1, \ldots, M\}$ and $i \in \{1, \ldots, N\}$, respectively, where $d \in \{1, \ldots, D\}$ denotes the dimension. As the image sequence $I = (i^{(1)}, \ldots, i^{(T)})$ is synchronized with $B = (b^{(1)}, \ldots, b^{(T)})$ via the timestamps $t \in \{1, \ldots, T\}$, the sets $B_i^{(d)}$ in turn induce corresponding sets $I_i^{(d)} = \{i^{(t)} \in I | b^{(t)} \in B_i^{(d)}\}$ in the visual domain, containing all stereo image pairs $i^{(t)} \in I$ that depict traffic situations in which the driver maintained a velocity and yaw rate falling into behavior class $\mathcal{B}_i^{(d)}$ (see Figure 3.4).

In contrast to image categories that are typically encountered in classical scene categorization tasks (see Chapter 2), the image categories $I_1^{\text{vel}}, \ldots, I_M^{\text{vel}}$ and $I_1^{\text{yaw}}, \ldots, I_N^{\text{yaw}}$ are not defined in terms of similar appearance in the visual domain. Rather, their formation is automatically induced by the observed behavior of the human driver who provides the training data $B = (b^{(1)}, \ldots, b^{(T)})$. In this sense, our approach is behavior-driven and ensures that the resulting image categories

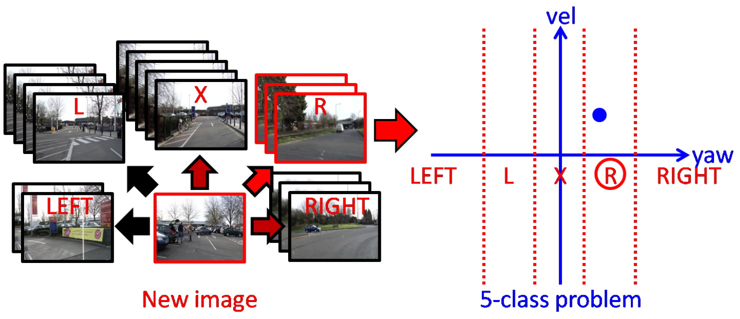


Figure 3.5: By comparing the visual scene content of previously unseen images to the image categories as learned by the system in training (left), we obtain corresponding predictions of the appropriate driving behavior (right).

are grounded in the task itself (i.e., safely driving a car). As a consequence, no explicit labeling of the image data $i^{(t)} \in I$ is required at any point, since the labeling is implicitly performed on-the-fly while driving the car. This is a fundamental difference to supervised learning in general, which is often limited in the availability of training data as it has to be manually annotated, either at the pixel-level (as in segmentation) or at the image-level (as in scene categorization). We will further discuss the process of acquiring ground truth data in order to provide the system with training data in a subsequent section of this chapter.

By discretizing the velocity and yaw rate values into behavior classes, we have effectively cast the behavior prediction problem as visual scene categorization: Given a previously unseen (stereo) image $i^{(t^*)} \notin I$ that shows the traffic situation at some future timestep $t^* \in \mathbb{N} \setminus \{1, \ldots, T\}$, our goal is to decide on the correct image categories $I_{m^*}^{(vel)} \in \{I_1^{(vel)}, \ldots, I_M^{(vel)}\}$ and $I_{n^*}^{(yaw)} \in \{I_1^{(yaw)}, \ldots, I_N^{(yaw)}\}$ that give us the appropriate driving behavior by their corresponding behavior classes $\mathcal{B}_{m^*}^{(vel)} \in \{\mathcal{B}_1^{(vel)}, \ldots, \mathcal{B}_M^{(vel)}\}$ and $\mathcal{B}_{n^*}^{(yaw)} \in \{\mathcal{B}_1^{(yaw)}, \ldots, \mathcal{B}_N^{(yaw)}\}$ (see Figure 3.5). To this end, a similar procedure as in classical scene categorization can now be applied, where the images $i^{(t)} \in I$ are represented in a suitable feature space that will be discussed in detail in Chapter 4, image classifiers are then learned from the resulting feature vectors as explained in the following section, and the individual classifier responses for $i^{(t^*)}$ are finally combined to obtain predictions of the appropriate behavior classes $\mathcal{B}_{m^*}^{(vel)}$ and $\mathcal{B}_{n^*}^{(yaw)}$, respectively.

## 3.2 Training Procedure

As explained in the previous section, our subdivision of the continuous-valued behavior space $\mathcal{B} \subseteq \mathbb{R}^2$ into disjoint behavior classes $\mathcal{B}_1^{(vel)}, \ldots, \mathcal{B}_M^{(vel)} \subseteq \mathcal{B}$ and $\mathcal{B}_1^{(yaw)}, \ldots, \mathcal{B}_N^{(yaw)} \subseteq \mathcal{B}$ yields corresponding image categories $I_1^{(vel)}, \ldots, I_M^{(vel)} \subseteq I$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)} \subseteq I$, respectively. By construction, all the images $i \in I_j^{(d)}$ are examples of traffic situations in which the human driver has performed some behavior $b \in \mathcal{B}_j^{(d)}$, and hence they can be used as positive training examples to learn an image classifier $C_j^{(d)}$ that is specific for behavior class $\mathcal{B}_j^{(d)}$. Now we show how to learn such image classifiers $C_1^{(vel)}, \ldots, C_M^{(vel)}$ and $C_1^{(yaw)}, \ldots, C_N^{(yaw)}$ from the training images in $I_1^{(vel)}, \ldots, I_M^{(vel)}$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)}$ (see Figure 3.6).

### 3.2.1 Ground Truth Acquisition

We have already seen how the two input data streams $B = (b^{(1)}, \ldots, b^{(T)})$ and $I = (i^{(1)}, \ldots, i^{(T)})$ are automatically acquired from observations of a human driver while conducting a camera-equipped vehicle, and that no manual annotations are necessary for organizing the training images $i^{(t)} \in I$ into image categories $I_1^{(vel)}, \ldots, I_M^{(vel)} \subseteq I$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)} \subseteq I$. However, an implicit assumption underlying the subsequent training procedure is that the human driver provides valid examples of how to drive correctly throughout the entire recording session, such that $b^{(t)} = (v^{(t)}, y^{(t)}) \in B$ is indeed an appropriate driving behavior when confronted with a traffic situation as depicted by $i^{(t)} \in I$, for all $t \in \{1, \ldots, T\}$. Only then will recognizing a previously unseen image $i^{(t^*)} \notin I$ with $t^* > T$ as belonging to the image categories $I_{m^*}^{(vel)}$ and $I_{n^*}^{(yaw)}$ enable the system to predict the appropriate driving behavior by the behavior classes $\mathcal{B}_{m^*}^{(vel)}$ and $\mathcal{B}_{n^*}^{(yaw)}$.

The assumption about the human driver to always perform the appropriate driving behavior is only made when acquiring the ground truth data $B$ and $I$, however. Once the system has learned the image classifiers $C_1^{(vel)}, \ldots, C_M^{(vel)}$ and $C_1^{(yaw)}, \ldots, C_N^{(yaw)}$ from the training images $i^{(t)} \in I$ that are distributed over the image categories $I_1^{(vel)}, \ldots, I_M^{(vel)}$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)}$, respectively, we do not make any such assumptions anymore. On the contrary, it is precisely the goal of the driver assistance application to detect inappropriate actions of the driver. This is achieved by having the trained system compare the observed behavior

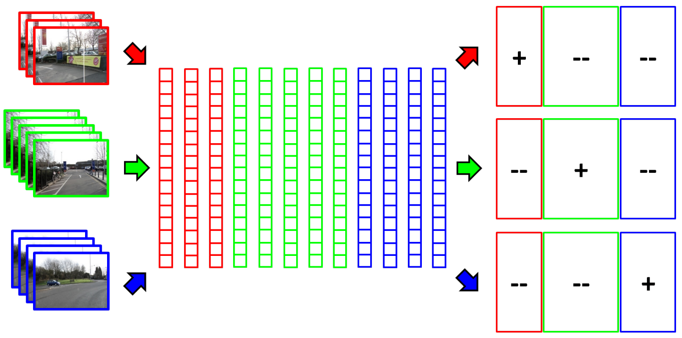


Figure 3.6: Internally, all images are first represented by feature vectors (left) that are subsequently split into positive and negative training examples (right). From these data splits, dedicated image classifiers are then learned.

$b^{(t^*)} \notin B$ of the human driver, which satisfies $b^{(t^*)} \in \mathcal{B}^{(vel)}_{m^\circ}$ and $b^{(t^*)} \in \mathcal{B}^{(yaw)}_{n^\circ}$ for some $m^\circ \in \{1, \dots, M\}$ and $n^\circ \in \{1, \dots, N\}$, to the predicted behavior classes $\mathcal{B}^{(vel)}_{m^*}$ and $\mathcal{B}^{(yaw)}_{n^*}$ as obtained by recognizing that the corresponding image $i^{(t^*)} \notin I$ belongs to the image categories $I^{(vel)}_{m^*}$ and $I^{(yaw)}_{n^*}$, respectively. The independence of the trained system from the actual behavior of a human driver is even more evident in the case of autonomous navigation, as there is no observed behavior $b^{(t^*)} = (v^{(t^*)}, y^{(t^*)})$ for any future $t^* \in \mathbb{N} \setminus \{1, \dots, T\}$ at all, and instead has to be generated from the predicted behavior classes $\mathcal{B}^{(vel)}_{m^*}$ and $\mathcal{B}^{(yaw)}_{n^*}$ themselves.

Besides learning the image classifiers $C^{(vel)}_1, \dots, C^{(vel)}_M$ and $C^{(yaw)}_1, \dots, C^{(yaw)}_N$ from the ground truth data in the two input streams $B$ and $I$, we also require ground truth data for evaluating these classifiers once the training has finished. This enables us to quantitatively measure the accuracy of the resulting velocity and yaw rate predictions. The requirement of using different ground truth data for training and testing can be met by recording additional streams $B'$ and $I'$, or by using only a subset $B'' \subset B$ and $I'' \subset I$ of the ground truth for training and testing on the remaining data $B' = B \backslash B''$ and $I' = I \backslash I''$. Either way, the accuracy is then measured by applying the $C^{(vel)}_1, \dots, C^{(vel)}_M$ and $C^{(yaw)}_1, \dots, C^{(yaw)}_N$ to all images $i^{(t')} \in I'$ of the test set, which results in predictions $\mathcal{B}^{(vel)}_{m_{t'}}$ and $\mathcal{B}^{(yaw)}_{n_{t'}}$ with $m_{t'} \in \{1, \dots, M\}$ and $n_{t'} \in \{1, \dots, N\}$, counting the correct predictions $c^{(vel)} = |\{i^{(t')} \in I' | b^{(t')} \in \mathcal{B}^{(vel)}_{m_{t'}}\}|$ and $c^{(yaw)} = |\{i^{(t')} \in I' | b^{(t')} \in \mathcal{B}^{(yaw)}_{n_{t'}}\}|$, and computing the ratios $a^{(vel)} = c^{(vel)} / |I'| \in [0, 1]$ and $a^{(yaw)} = c^{(yaw)} / |I'| \in [0, 1]$.

### 3.2.2 Image Classifier Training

There are several possibilities when it comes to learning discriminative models for multiple classes to be distinguished, as in the case of our image categories $I_1^{(vel)}, \ldots, I_M^{(vel)}$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)}$. Two widely used paradigms in this context are the one-versus-all scheme and the one-versus-one scheme (see Figure 3.7). Both of these paradigms have been shown to work comparably well in practice, as far as the accuracy of their resulting predictions is concerned, but they differ in terms of other relevant aspects such as the memory consumption, for example. Therefore, we briefly introduce and discuss both alternatives in the following, and then pick the one that better matches the intended applications of our system.

In the one-versus-all paradigm (see Figure 3.7, left), a single image classifier $C_j^{(d)}$ is learned for each of the image categories $I_j^{(d)}$. The training data for $C_j^{(d)}$ consists in a set of positive training images $P_j^{(d)} = I_j^{(d)} \subset I$, containing all available ground truth images $i^{(t)} \in I$ that are known to belong to the image category $I_j^{(d)}$, and a set of negative training images $N_j^{(d)} = I \setminus P_j^{(d)}$ that contains all other images from the available ground truth data, regardless of their actual class memberships. In the one-versus-one paradigm (see Figure 3.7, right), the image categories $I_j^{(d)}$ are considered pairwise, hence a classifier $C_{j,k}^{(d)}$ is learned for each possible pair of image categories $I_j^{(d)}$ and $I_k^{(d)}$, where $k > j$ to prevent learning classifiers twice due to symmetry. In this scheme, the positive training examples for classifier $C_{j,k}^{(d)}$ consist of all available ground truth images $P_{j,k}^{(d)} = I_j^{(d)} \subset I$ that are known to belong to image category $I_j^{(d)}$, and the negative training examples consist of all ground truth images $N_{j,k}^{(d)} = I_k^{(d)} \subset I$ that belong to image category $I_k^{(d)}$.

Since the training examples in $P_{j,k}^{(d)}$ and $N_{j,k}^{(d)}$ are only a subset of the available ground truth data in $I$, which generally satisfies $|P_{j,k}^{(d)} \cup N_{j,k}^{(d)}| << |I|$, an advantage of the one-versus-one paradigm is its comparably low memory consumption when learning the classifiers $C_{j,k}^{(d)}$. However, there are $M(M-1)/2$ and $N(N-1)/2$ possible pairs that can be formed without symmetry from the image categories $I_1^{(vel)}, \ldots, I_M^{(vel)}$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)}$, respectively, leading to a quadratic number of classifiers $C_{j,k}^{(d)}$. In contrast, the one-versus-all paradigm only requires learning $M$ and $N$ classifiers $C_j^{(d)}$, respectively, which is preferrable. Its disadvantage is the higher memory consumption in training, as each $C_j^{(d)}$ is learned from all available

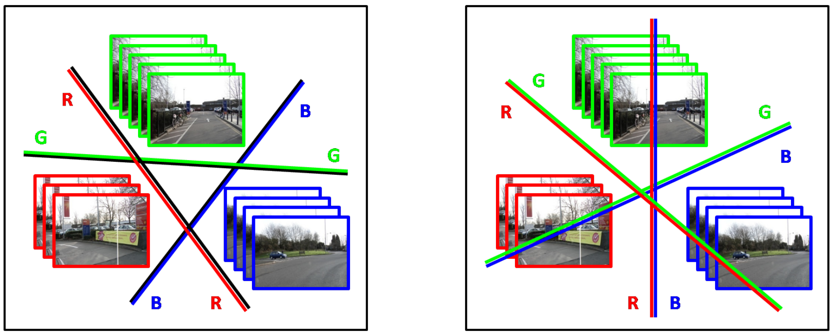


Figure 3.7: In the one-versus-all scheme (left), classifiers are trained separately for each class, using all other classes as negative examples. In the one-versus-one scheme (right), classifiers are trained pairwise with data of only two classes each.

ground truth data $P_j^{(d)} \cup N_j^{(d)} = I$. However, the training is conducted off-line prior to the actual application on a mobile platform, as far as we are concerned, hence it is safe to assume that no particular constraints on the hardware resources apply in the training phase, and we therefore employ the one-versus-all scheme.

While the overall scene categorization problem involves multiple categories $I_1^{(vel)}, \ldots, I_M^{(vel)}$ and $I_1^{(yaw)}, \ldots, I_N^{(yaw)}$, respectively, learning the image classifiers $C_1^{(vel)}, \ldots, C_M^{(vel)}$ and $C_1^{(yaw)}, \ldots, C_N^{(yaw)}$ is thus a series of binary problems only, which can be handled separately from each other. Any type of binary classifier can be used for the individual $C_j^{(d)}$: In our implementation, we use an ensemble of decision stumps $S_{j,1}^{(d)}, \ldots, S_{j,R}^{(d)}$ that are combined in a GentleBoost framework, where the optimal number of decision stumps $R \in \mathbb{N}$ is empirically determined by blockwise cross-validation, for each of the $C_j^{(d)}$ (see Chapter 5). Specifically, in each round $r \in \{1, \ldots, R\}$ of the boosting procedure, the most discriminative feature vector dimension $\hat{f}_{j,r}^{(d)} \in \{1, \ldots, F\}$ is determined by iteratively checking which of all feature vector dimensions $f \in \{1, \ldots, F\}$ separates $P_j^{(d)}$ and $N_j^{(d)}$ best, resulting in a corresponding threshold $\hat{\theta}_{j,r}^{(d)} \in \mathbb{R}$ and an accuracy-based weight $\hat{\omega}_{j,r}^{(d)} \in \mathbb{R}$. The boosting procedure also performs a dimensionality reduction on the usually high-dimensional input feature space $\mathcal{F} \subseteq \mathbb{R}^F$ (see Chapter 4), as we generally have $R << F$ in practice, consistent with our philosophy to tolerate high computational complexity and memory requirements in the training phase to enable a comparably low resource consumption in the application later on.

### 3.2.3 Training Data Balance

Before we explain how to combine the decision stumps $S_{j,1}^{(d)}, \ldots, S_{j,R}^{(d)}$ into their corresponding image classifiers $C_j^{(d)}$, and how to combine these binary classifiers $C_1^{(vel)}, \ldots, C_M^{(vel)}$ and $C_1^{(yaw)}, \ldots, C_N^{(yaw)}$ into an M-ary classifier $C^{(vel)}$ and an N-ary classifier $C^{(yaw)}$, respectively, one important aspect has yet to be discussed: Discriminative models for binary classification implicitly assume the cardinalities $|P_j^{(d)}|$ and $|N_j^{(d)}|$ of the positive and the negative training data sets to be equal. Violating this assumption effectively gives more weight to the overrepresented set during training, which leads to a bias in the responses of the resulting classifier later on. For example, when training a classifier to distinguish traffic situations in which braking is appropriate from all other situations, without compensating for the fact that there are much more examples of the latter type (i.e., braking is performed only occasionally in practice), the resulting classifier would have a tendency to predict not to brake, just because there were more such examples.

The reason is that the optimization problem underlying the learning process involves a scoring function that imposes a penalty for each training example being misclassified. If $|P_j^{(d)}| = |N_j^{(d)}|$ is satisfied, the sum of these penalties is at minimum when the best possible separation between $P_j^{(d)}$ and $N_j^{(d)}$ has been reached. If there are significantly more examples in either of the two sets, however, the average number of misclassifications will be higher for that particular set, which is penalized by the scoring function. As a consequence, the optimization yields parameters that avoid misclassifying the examples in the overrepresented training set, at the cost of misclassifying examples in the underrepresented set. While the resulting separation between $P_j^{(d)}$ and $N_j^{(d)}$ is not the same as before, its sum of penalties is lower, which should not be the case (see Figure 3.8).

Several strategies exist in the machine learning community for balancing the training data sets, as this problem is of significance in terms of the resulting classification performance, and many classification tasks are multi-class problems for which imbalanced data is particularly prevalent. One approach is to use the smaller training data set as it is, and to downsample the larger training data set such that it has the same size. The downsampling can be performed randomly, or by identifying a subset of training examples that preserves as much of the

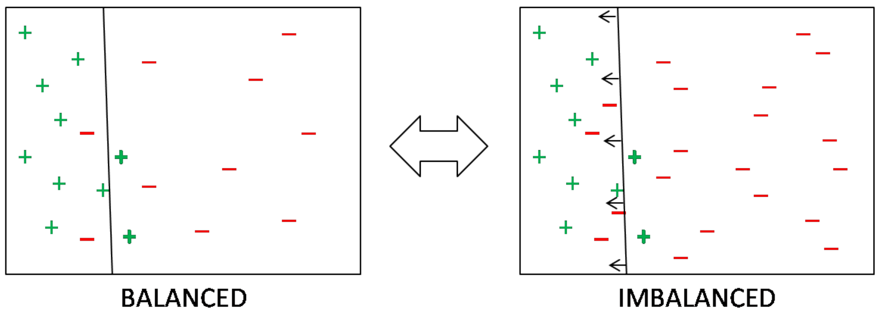


Figure 3.8: Imbalanced training sets (right) can influence the decision boundary, as misclassifications in the overrepresented training set have a stronger effect. Balancing the training sets (left) before the learning process avoids this problem.

information content as possible (i.e., by removing largely redundant examples). For example, when operating on data streams like in our case, successive frames carry similar information and can thus be dropped to some extent. An advantage of downsampling is that the training data sets are relatively small, while at the same time preserving most of the information. On the downside, information is deliberately thrown away, and learning from the smaller sets might also lead to inaccuracies in the parameter optimization.

Another viable strategy for balancing the training data sets is to keep the larger training data set as it is and upsample the smaller training data set instead, until it has the same size as the larger set. In practice, this can be achieved by simply including some of the existing training examples multiple times. This approach involves a similar process of choosing suitable examples from the data set in question, which can again be done either randomly or by identify the most informative examples. However, the upsampling strategy does not actually increase the information content in the training set, as the information is entirely redundant. Upsampling has the advantage that all available training data is used for learning, potentially leading to higher accuracy. However, a strong replication of training examples in the smaller set might lead to artifacts caused by the resulting distortion of the actual distribution. Since we are dealing with streams of data, where relatively large numbers of frames can be dropped without losing much information if done carefully, we adopt the downsampling approach.

## 3.3 Application Phase

We now assume that we have a number of decision stumps $S_{j,1}^{(d)}, \dots, S_{j,R}^{(d)}$ for each image classifier $C_j^{(d)}$ to be learned, which are the result of imposing a one-versus-all training scheme on the image categories $I_1^{(vel)}, \dots, I_M^{(vel)} \subseteq I$ and $I_1^{(yaw)}, \dots, I_N^{(yaw)} \subseteq I$ containing the available ground truth images, respectively. Each decision stump $S_{j,r}^{(d)}$ is defined by exactly one feature space dimension $\hat{f}_{j,r}^{(d)} \in \{1, \dots, F\}$ that separates the training examples in $P_j^{(d)}$ and $N_j^{(d)}$ best at round $r \in \{1, \dots, R\}$ of the boosting, a corresponding threshold $\hat{\theta}_{j,r}^{(d)} \in \mathbb{R}$ that applies to this particular feature space dimension, and a weighting factor $\hat{\omega}_{j,r}^{(d)} \in \mathbb{R}$ that defines the contribution of $S_{j,r}^{(d)}$ to the ensemble formed by all $S_{j,1}^{(d)}, \dots, S_{j,R}^{(d)}$. Remember that each of the resulting image classifiers $C_j^{(d)}$ is specifically trained to recognize images of category $I_j^{(d)}$, and hence to predict behavior class $\mathcal{B}_j^{(d)}$.

In the following, we explain how the decision stumps $S_{j,r}^{(d)}$ are combined to obtain the image classifiers $C_j^{(d)}$, and how to combine the binary classifiers $C_j^{(d)}$ to form multi-class image classifiers $C^{(vel)}$ and $C^{(yaw)}$, respectively. Furthermore, we introduce additional mechanisms such as temporal stabilization and various confidence measures to our behavior prediction framework, which can be derived from the continuous-valued responses of the image classifiers $C^{(vel)}$ and $C^{(yaw)}$, to address conceptual aspects that we have not yet considered. Our experiments show that incorporating these mechanisms into our system architecture leads to considerable improvements of the resulting prediction accuracy (see Chapter 5).

### 3.3.1 Combined Prediction

Given the decision stumps $S_{j,r}^{(d)}$ and a new image $i \notin I$, the ensemble classifier $C_j^{(d)}$ for deciding whether or not $i \in I_j^{(d)}$ is defined by the weighted linear combination $C_j^{(d)}(f) = \sum_{r=1}^{R} \hat{\omega}_{j,r}^{(d)} S_{j,r}^{(d)}(f) \in \mathbb{R}$, where $f \in \mathcal{F} \subseteq \mathbb{R}^F$ is the feature vector that represents the input image $i$ in our $F$-dimensional feature space (see Chapter 4), $S_{j,r}^{(d)}(f) \in \mathbb{R}$ is the response of decision stump $S_{j,r}^{(d)}$ on the input image $i$, and $C_j^{(d)}(f) \in \mathbb{R}$ is the response of the resulting binary classifier $C_j^{(d)}$. This process is performed in parallel for each of the $C_j^{(d)}$, hence we end up with a series of binary classifier scores $C_1^{(vel)}(f), \dots, C_M^{(vel)}(f)$ and $C_1^{(yaw)}(f), \dots, C_N^{(yaw)}(f) \in \mathbb{R}$, respectively, all of which are continuous-valued in nature (see Figure 3.9, left).

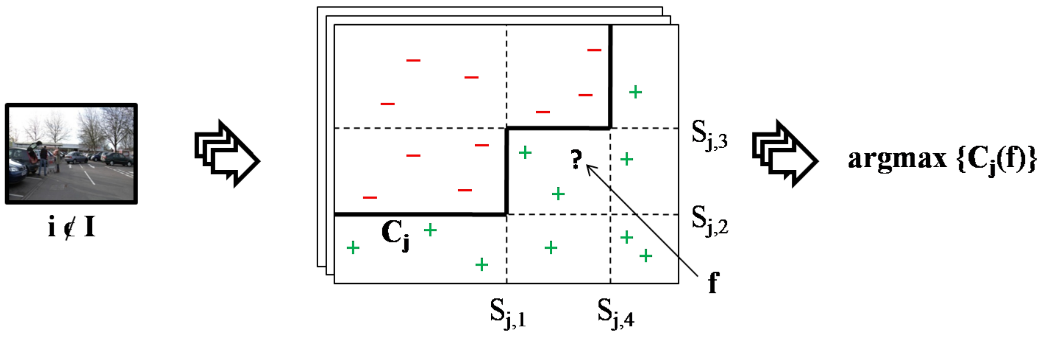


Figure 3.9: A new image $i$ is converted into its feature vector representation $f$ and processed by all binary classifiers $C_j$ in parallel (left). The resulting scores $C_j(f)$ are then compared to determine the strongest-response classifier (right).

Intuitively, each of the continuous-valued binary classifier scores $C_j^{(d)}(f) \in \mathbb{R}$ is a similarity measure that quantitatively tells us how well the input image $i \notin I$ matches the corresponding image category $I_j^{(d)} \subset I$. To obtain decisions $C^{(vel)}(f) \in \{1, \ldots, M\}$ and $C^{(yaw)}(f) \in \{1, \ldots, N\}$, respectively, it is therefore reasonable to compare all of the velocity-related binary classifier responses $C_1^{(vel)}(f), \ldots, C_M^{(vel)}(f) \in \mathbb{R}$ and all of the yaw-rate related classifier responses $C_1^{(yaw)}(f), \ldots, C_N^{(yaw)}(f) \in \mathbb{R}$ to each other, and to decide on the image categories $I_{m^*}^{(vel)} \subset I$ and $I_{n^*}^{(yaw)} \subset I$ whose corresponding image classifiers $C_{m^*}^{(vel)}$ and $C_{n^*}^{(yaw)}$ have the strongest responses $C_{m^*}^{(vel)}(f) \in \mathbb{R}$ and $C_{n^*}^{(yaw)}(f) \in \mathbb{R}$, respectively. Thus, we have $m^* = \mathrm{argmax}_{m \in \{1,\ldots,M\}} C_m^{(vel)}$ and $n^* = \mathrm{argmax}_{n \in \{1,\ldots,N\}} C_n^{(yaw)}$, and the predictions are the behavior classes $\mathcal{B}_{m^*}^{(vel)}$ and $\mathcal{B}_{n^*}^{(yaw)}$ (see Figure 3.9, right).

The above procedure is a winner-take-all scheme, as the decisions $C^{(vel)}(f) \in \{1, \ldots, M\}$ and $C^{(yaw)}(f) \in \{1, \ldots, N\}$ are given by the image classifiers $C_{m^*}^{(vel)}$ and $C_{n^*}^{(yaw)}$ with the strongest responses $C_{m^*}^{(vel)}(f) \in \mathbb{R}$ and $C_{n^*}^{(yaw)}(f) \in \mathbb{R}$ alone. In particular, the continuous-valued scores $C_m^{(vel)}(f) \in \mathbb{R}$ and $C_n^{(yaw)}(f) \in \mathbb{R}$ of all other image classifiers $C_m^{(vel)}$ and $C_n^{(yaw)}$ with $m \in \{1, \ldots, M\} \setminus \{m^*\}$ and $n \in \{1, \ldots, N\} \setminus \{n^*\}$, respectively, are entirely disregarded afterwards. However, they can also be used to compute a confidence value for the predictions, which we discuss in one of the next sections. But first we address another important aspect that should be considered when computing the $C_j^{(d)}(f)$ and making the decisions $C^{(vel)}$ and $C^{(yaw)}$, namely, to exploit temporal coherence between frames.

### 3.3.2 Temporal Stabilization

Until now, the predictions $\mathcal{B}_{m^*}^{(vel)}$ and $\mathcal{B}_{n^*}^{(yaw)}$ of the system for an image $i \notin I$ with feature vector representation $f \in \mathcal{F} \subseteq \mathbb{R}^F$ are given by the image categories $I_{m^*}^{(vel)}$ and $I_{n^*}^{(yaw)}$ whose corresponding binary classifiers $C_{m^*}^{(vel)}$ and $C_{n^*}^{(yaw)}$ have the strongest responses, but without considering any other images $i' \notin I$. That is, our system architecture as described so far implicitly assumes the images $i, i' \notin I$ to be independent samples and hence to be processed separately from each other. However, this assumption is clearly not valid in practical applications, as all $i' \notin I$ are acquired by a car-mounted camera while driving, just like the ground truth data streams $I = (i^{(1)}, \ldots, i^{(T)})$ and $B = (b^{(1)}, \ldots, b^{(T)})$ were obtained, and hence they actually form similar data streams $I' = (i'^{(T+1)}, \ldots, i'^{(T+T')})$ and $B = (b'^{(T+1)}, \ldots, b'^{(T+T')})$ each having a total duration of $T' \in \mathbb{N}$ frames.

Considering the nature of the drivers actions, it can be observed from the CAN data that these are not singular events but rather have a certain duration. This is partly because the vehicle is a massive physical body and therefore subject to considerable inertia that has to be overcome for an action to be effective (e.g., when accelerating and braking), and partly because of the spatial extent of the road that is being followed (e.g., when steering in a curve). In other words, the actions of the human driver always exhibit temporal coherence, and high-frequency changes are neither plausible nor appropriate driving behavior, which is however not yet addressed by the single-frame predictions $\mathcal{B}_{m^*}^{(vel)}$ and $\mathcal{B}_{n^*}^{(yaw)}$.

This lack of accounting for the temporal coherence between successive frames $i'^{(T+t')} \in I'$ and $i'^{(T+t'+1)} \in I'$ as well as $b'^{(T+t')} \in B'$ and $b'^{(T+t'+1)} \in B'$, with $t' \in \{1, \ldots, T'-1\}$ as an index, generally results in noisy single-frame predictions $\mathcal{B}_{m^*}^{(vel)}$ and $\mathcal{B}_{n^*}^{(yaw)}$, which is particularly problematic at the boundaries between adjacent behavior classes $\mathcal{B}_{m}^{(vel)}, \mathcal{B}_{m+1}^{(vel)}$ and $\mathcal{B}_{n}^{(yaw)}, \mathcal{B}_{n+1}^{(yaw)}$. But knowing that high-frequency changes in the behavior predictions are implausible, as explained above, and therefore can only be erroneous predictions of the system rather than the actual behavior of the driver, we are able to eliminate such changes by applying a window-based filter to the single-frame predictions of the system. Median filtering is preferable to Gaussian filtering, as it leaves the correct predictions unaffected by the noisy outliers, no matter how wrong the latter might be.

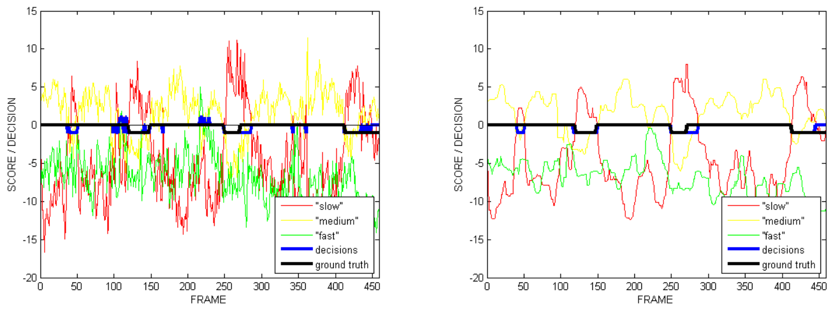


Figure 3.10: Example of temporal stabilization. Shown are the continuous-valued binary classifier responses for a previously unseen image sequence and the resulting decisions with ground truth, before (left) and after stabilization (right).

More specifically, we apply a one-dimensional median filter $\mu : \mathbb{N}^w \to \mathbb{N}$ with window size $w \in \mathbb{N}_{>0}$ to the temporal sequences $C^{(vel)}(f^{(T+1)}), \dots, C^{(vel)}(f^{(T+T')})$ and $C^{(yaw)}(f^{(T+1)}), \dots, C^{(yaw)}(f^{(T+T')})$ formed by the single-frame predictions, where $\mu(n_1, \dots, n_w) \in \mathbb{N}$ is the median value of all $n_1, \dots, n_w \in \mathbb{N}$ inside the current window. The median filter $\mu$ is in fact successively applied to each of the positions $t' \in \{T+1, \dots, T+T'\}$ such that the window is effectively shifted along the entire temporal sequences, replacing each of the single-frame predictions $C^{(vel)}(f^{(t')}) \in \{1, \dots, M\}$ and $C^{(yaw)}(f^{(t')}) \in \{1, \dots, N\}$ by stabilized predictions $\mu(C^{(vel)}(f^{(t')})) \in \{1, \dots, M\}$ and $\mu(C^{(yaw)}(f^{(t')})) \in \{1, \dots, N\}$, respectively. Intuitively, this filter operation performs a soft version of majority voting within the sliding window, eliminating outliers that arise due to misclassifications.

The above filtering only operates at the level of the single-frame predictions $C^{(d)}(f^{(t')}) \in \mathbb{N}$, after the decisions have already been made. In addition, we therefore stabilize the underlying continuous-valued responses $C_j^{(d)}(f^{(t')}) \in \mathbb{R}$ of the binary classifiers as well, before they are combined to make the actual decisions. This is achieved by applying a second median filter $\mu' : \mathbb{R}^w \to \mathbb{R}$ to the temporal sequences $C_j^{(vel)}(f^{(T+1)}), \dots, C_j^{(vel)}(f^{(T+T')})$ and $C_j^{(yaw)}(f^{(T+1)}), \dots, C_j^{(yaw)}(f^{(T+T')})$ for each $j \in \{1, \dots, M\}$ and $j \in \{1, \dots, N\}$, respectively. In practice, we found this combination to be more effective than filtering at only one of these two levels, and also superior to using Gaussian filters instead. We report on the accuracy with and without temporal stabilization in our experiments (see Chapter 5).

### 3.3.3 Ambiguous Situations

One last aspect that has not yet been addressed by our system architecture, up to this point, is the occasional encountering of ambiguous traffic situations in which more than one particular driving behavior is possible. For example, such situations occur when the vehicle arrives at a crossing, where the driver can choose to continue in any of the available directions by steering appropriately. The decision mainly depends on the unobservable intentions of the human driver, and it is therefore impossible for the system to make any valid predictions from the visual scene content itself. So far, our system architecture has been designed in a way that mainly addresses the challenge of identifying multiple, visually dissimilar traffic situations as requiring the same choice of appropriate driving behavior, whereas in this context it now should be able to identify individual traffic situations as allowing for multiple possible driving behaviors. Doing so requires an additional mechanism that we build on the computation of confidence values for the individual behavior predictions of our system, and we proceed to describe this confidence mechanism and how it prevents erroneous predictions in ambiguous traffic situations in the following.

The key idea is to not only consider the binary classifiers $C^{(vel)}_{m^*}$ and $C^{(yaw)}_{n^*}$ that have the strongest responses $C^{(vel)}_{m^*}(f) \in \mathbb{R}$ and $C^{(yaw)}_{n^*}(f) \in \mathbb{R}$, when given a traffic situation depicted by image $i \notin I$ with feature vector $f \in \mathcal{F} \subseteq \mathbb{R}^F$, but also how they relate to the other binary classifier responses $C^{(vel)}_{m}(f) \in \mathbb{R}$ and $C^{(yaw)}_{n}(f) \in \mathbb{R}$, where $j \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, N\}$, respectively. Currently, none of the $C^{(vel)}_{j}(f)$ and $C^{(yaw)}_{j}(f)$ is taken into account any further once the decisions $C^{(vel)}(f) \in \{1, \ldots, M\}$ and $C^{(yaw)}(f) \in \{1, \ldots, N\}$ have been made, no matter how strong they might be. Intuitively, however, these decisions should be regarded as having a high confidence if $C^{(vel)}_{m^*}(f)$ and $C^{(yaw)}_{n^*}(f)$ are high compared to the other $C^{(vel)}_{j}(f)$ and $C^{(yaw)}_{j}(f)$, and as having a low confidence if $C^{(vel)}_{m^*}(f)$ and $C^{(yaw)}_{n^*}(f)$ are not significantly higher than any of the other $C^{(vel)}_{j}(f)$ and $C^{(yaw)}_{j}(f)$, respectively. Then, the adequate response of the system would consist in saying that it is unable to make a decision in the current situation, or in providing the plausible behaviors without a decision. We therefore introduce a reject option to automatically suppress predictions of insufficient confidence.

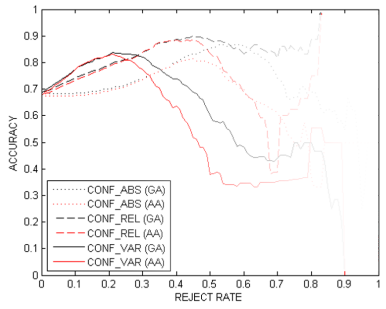


Figure 3.11: Example of the three different confidence measures. Typically, the third measure has the strongest effect at low rejection rates (although becoming detrimental for higher rejection rates), while the first measure is least effective.

There are several possiblities how to derive confidence values $\zeta^{(vel)}(f) \in \mathbb{R}$ and $\zeta^{(yaw)}(f) \in \mathbb{R}$ from the binary classifier responses $C_1^{(vel)}(f), \ldots, C_M^{(vel)}(f) \in \mathbb{R}$ and $C_1^{(yaw)}(f), \ldots, C_N^{(yaw)}(f) \in \mathbb{R}$, respectively. A simple measure is to just consider the magnitudes $\zeta^{(vel)}(f) = C_{m^*}^{(vel)}(f) \in \mathbb{R}$ and $\zeta^{(yaw)}(f) = C_{n^*}^{(yaw)}(f) \in \mathbb{R}$ of the strongest classifier responses $C_{m^*}^{(vel)}(f)$ and $C_{n^*}^{(yaw)}(f)$. This measure already goes beyond the decisions $C_{m^*}^{(vel)}(f) \in \{1, \ldots, M\}$ and $C_{n^*}^{(yaw)}(f) \in \{1, \ldots, N\}$ themselves, and could be thresholded by appropriate $\Theta^{(vel)}, \Theta^{(yaw)} \in \mathbb{R}$ to suppress the decisions if $\zeta^{(vel)}(f) < \Theta^{(vel)}$ and $\zeta^{(yaw)}(f) < \Theta^{(yaw)}$, respectively. However, this measure does not consider any of the other classifier responses $C_j^{(d)}(f) \in \mathbb{R}$. Another confidence measure that takes into account one additional classifier response is to compute the ratios $\zeta^{(vel)}(f) = C_{m^*}^{(vel)}(f)/C_{m^+}^{(vel)}(f) \in \mathbb{R}$ and $\zeta^{(yaw)}(f) = C_{n^*}^{(yaw)}(f)/C_{n^+}^{(yaw)}(f) \in \mathbb{R}$, where $C_{m^+}^{(vel)}$ and $C_{n^+}^{(yaw)}$ are the binary classifiers with the second-highest responses, respectively. This measure is also used in other domains, such as the stereo correspondence problem in which image patches have to be matched based on their pairwise similarity. Taking this idea one step further, we can take into account all of the binary classifier responses $C_1^{(vel)}(f), \ldots, C_M^{(vel)}(f) \in \mathbb{R}$ and $C_1^{(yaw)}(f), \ldots, C_N^{(yaw)}(f) \in \mathbb{R}$ by computing their variances $\zeta^{(vel)}(f) = \sum_{m=1}^{M}(C_m^{(vel)}(f) - \text{avg}(C_1^{(vel)}, \ldots, C_M^{(vel)}))^2/M \in \mathbb{R}$ and $\zeta^{(yaw)}(f) = \sum_{n=1}^{N}(C_n^{(yaw)}(f) - \text{avg}(C_1^{(yaw)}, \ldots, C_N^{(yaw)}))^2/N \in \mathbb{R}$, where $\text{avg}(C_1^{(vel)}, \ldots, C_M^{(vel)}) = \sum_{m=1}^{(M)} C_m^{(vel)}(f)/M \in \mathbb{R}$ and $\text{avg}(C_1^{(yaw)}, \ldots, C_N^{(yaw)}) = \sum_{n=1}^{(N)} C_n^{(yaw)}(f)/N$ is the arithmetic mean, respectively (see Figure 3.11).

To the best of our knowledge, we have been the first to propose a vision-based scene categorization approach for driving behavior prediction. It is worth noting that a closely related approach was published shortly after ours, however [84], which also uses scene categorization for driving behavior prediction. Nevertheless, our approach being published first [43] proves the originality of our solution.

Second, our approach is based on a discretization into multiple behavior classes, which enables an arbitrarily fine-grained subdivision of the behavior space, depending on the requirements of the intended application. The other approach, in contrast, only considers binary problems such as whether or not to press a pedal or to turn the steering wheel, but not to what extent. Importantly, these binary problems can be seen as a special case of our more generic behavior prediction framework, and the idea of approximating the continuous-valued behavior data is explicitly developed in our framework but not in the other.

Third, the features we use to represent the traffic scenes capture information about the scene at the semantic level, as they are based on a scene decomposition that is learned in a supervised manner. In contrast, the other approach employs raw image-filter based responses, only capturing the magnitude and orientation of edge information in the scene, without any explicit notion of objects. Arguably, our semantic object-level representation is therefore more sophisticated, and we conduct a quantitative comparison of the two representations in our experiments (see Chapter 5), showing our object-level representation to be more robust.

Fourth, we have addressed the inherent limitations of operating at the single-frame level as well as the processing of ambiguous traffic situations, and we have proposed temporal stabilization and a confidence mechanism to this end. The other approach, in contrast, does not consider such limitations at all, although our experiments give quantitative evidence that the proposed techniques help.

To summarize, our scene categorization approach to driving behavior prediction is indeed novel, and can be seen as a more generic version of the other approach, on a conceptual level. We continue to compare these two approaches in the following chapters up to the quantitative level, as the other approach is the most related work to our own. In the next chapter, we will turn towards the question of how the traffic scenes are represented, before they are fed into the scene categorization architecture that we have presented in this chapter.

# Chapter 4

# Traffic Scene Representations

In this chapter, we examine two fundamentally different but related feature spaces to which all camera images depicting traffic scenes are converted. The resulting feature vectors can then be processed by our scene categorization architecture as described in the previous chapter. More specifically, one representation operates at the level of raw image filter responses, while the other one operates at the level of object classifier responses that correspond to semantic entities in the scene.

The filter-based representation employs a series of oriented edge filters at different scales that are applied in parallel to each image. Their continuous-valued response maps are then post-processed by computing the average response values within regularly-spaced image grid cells, and the resulting feature vectors are further stabilized by Gaussian kernel weighting inside each grid cell. In contrast, the object-level representation employs an array of dedicated object classifiers, each of which is specific for a different semantic entity and trained beforehand in a supervised manner. The continuous-valued response maps that result from applying these classifiers in parallel to each image are then post-processed as well, by averaging over image grid cells and Gaussian kernel weighting as before.

Both traffic scene representations address the challenge of dealing with the enormous visual complexity of urban traffic environments. In this context, our behavior prediction framework serves as a testbed, enabling us to directly compare these representations to each other, both qualitatively as in this chapter and quantitatively as in the next chapter. Although the underlying techniques are well-known state-of-the-art methods, no such comparison has been done before.

## 4.1 Image Filter Responses

We begin by formally describing the filter-based representation of traffic scenes as employed by the state-of-the-art method for driving behavior prediction that is most directly related to our own work [84]. This representation is an extension of the classical GIST [76], in which a predefined number of edge filter kernels $e_\omega^\lambda$ with different orientations $\omega$ and scales $\lambda$ is applied to each of the input images $i$ as explained below. The result is a continuous-valued response map $r_\omega^\lambda$ for each filter kernel $e_\omega^\lambda$, having the same size as the original image $i$ and containing values that are directly determined by the values in $e_\omega^\lambda$. Their subsequent averaging over regular image grid cells is part of the GIST procedure, while their stabilization by Gaussian kernel weights is an extension as in visual codeword improvement. [78]

Despite the simplicity of the resulting feature vectors, which only represent edge information without any semantic notion attached to them, the filter-based representation performs well in practice (see Chapter 5), and serves as a baseline to our semantic object-level representation as presented later on in this chapter. It is hypothesized [84] that conducting a vehicle involves a substantial amount of sub-conscious, pre-attentive visual processing by the human driver and therefore can largely be explained by such filter-based representations, because early vision in biological systems also relies on filter operations to a large extent.

### 4.1.1 Oriented Edge Filters

Generally speaking, an image filter operation is uniquely defined by its filter kernel $k : \{0, \ldots, W_k-1\} \times \{0, \ldots, H_k-1\} \to \mathbb{R}$, where $W_k, H_k \in \mathbb{N}$ denote the width and height of the filter kernel, respectively. In practice, we typically have $W_k << W$ and $H_k << H$ for input images $i : \{0, \ldots, W-1\} \times \{0, \ldots, H-1\} \to \{0, \ldots, 255\}^3$ in RGB color space. To apply $k$, the color image $i$ is first converted into a grayscale image $i'$, obtained by setting $i'(w, h) = (i_R(w, h)+i_G(w, h)+i_B(w, h))/3$ at each pixel $(w, h) \in \{0, \ldots, W - 1\} \times \{0, \ldots, H - 1\}$, where $i_R, i_G, i_B$ denote the red, green, and blue color channel of $i$, respectively. The grayscale image $i'$ can then be convolved with the filter kernel $k$, which results in a response map $r : \{0, \ldots, W - 1\} \times \{0, \ldots, H - 1\} \to \mathbb{R}$ whose values are given by $r(w, h) = \sum_{w'=-X}^{X} \sum_{h'=-Y}^{Y} i(w+w', h+h')k(X-w', Y-h')$, with $X = \lfloor W_k/2 \rfloor, Y = \lfloor H_k/2 \rfloor$.

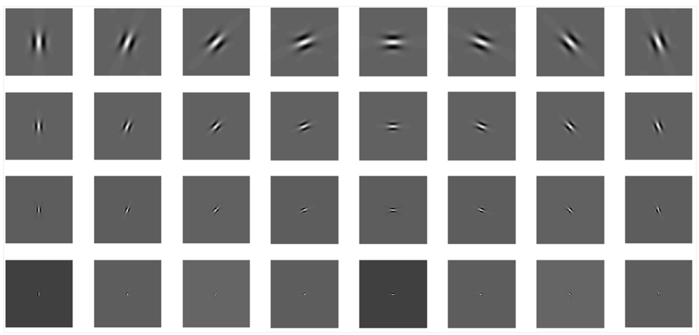


Figure 4.1: Example of a filter bank that consists of Gabor kernels, which are sensitive to oriented contrast edges at various scales. These are used by the filter-based representation of traffic scenes, as in the state-of-the-art method [84].

Oriented edge filters have a special type of kernel $k = e_\omega : \{0, \ldots, W_k - 1\} \times \{0, \ldots, H_k - 1\} \to \mathbb{R}$ that is sensitive to contrast edges of orientation $\omega \in [0°, 180°)$ in the grayscale image $i'$ with which is it convolved. Its characteristic structure is given by an elongated center region of positive values, surrounded by one or two corresponding regions of negative values to enhance the local contrast. A well-known implementation is the Gabor filter, which combines a two-dimensional Gaussian envelope function $g(x, y) = \exp(-(\tilde{x}^2 + \tilde{y}^2\epsilon^2)/2\Sigma^2) \in \mathbb{R}$ with a harmonic carrier function $c(x, y) = \cos(2\pi\tilde{x}/\lambda) \in \mathbb{R}$, where $\tilde{x} = x\cos\omega + y\sin\omega$ and $\tilde{y} = -x\sin\omega + y\cos\omega$ give rise to the edge orientation, $\lambda \in \mathbb{R}$ is the wavelength of $c$ and hence sets the scale of the filter, and $\Sigma, \epsilon \in \mathbb{R}$ denote the standard deviation and ellipticity of $g$, respectively. The Gabor filter kernel is then given by $k(w, h) = e_\omega(w, h) = c(w, h)g(w, h) \in \mathbb{R}$, for all $(w, h) \in \{0, \ldots, W_k\} \times \{0, \ldots, H_k\}$.

In practice, such filter kernels are typically pre-computed for a fixed number of orientations $\omega$ and wavelengths $\lambda$, thus leading to the formation of a filter bank (see Figure 4.1). Specifically, we use 8 different orientations and 4 different scales, such that the filter bank consists of 32 edge filter kernels $e_\omega^\lambda$ in total. When applied to an input image $i$, each of these filter kernels is independently convolved with $i$ in parallel, and hence the result is a two-dimensional response map $r_\omega^\lambda$ for each $e_\omega^\lambda$. Although the amount of data increases linearly with the size of the filter bank, parallel convolution operations effectively circumvent a corresponding increase in computation time, which is important for practical applications.

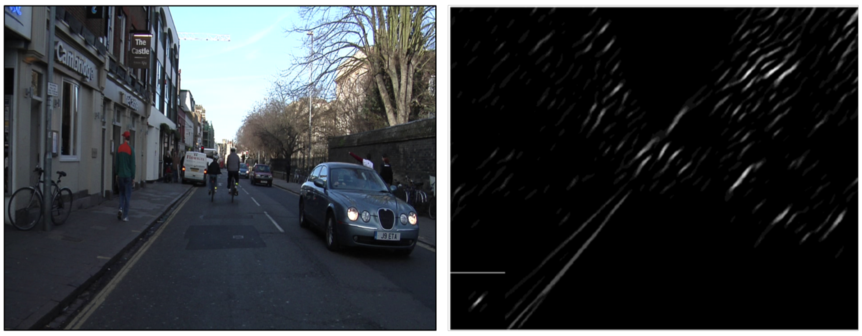


Figure 4.2: Input image (left), and one of the continuous-valued response maps that result from the application of the Gabor filter bank (right). The filter kernel used in the underlying convolution is shown as well (see bottom left corner).

### 4.1.2 Image Grid Histograms

By applying the pre-computed filter bank of Gabor kernels $e_\omega^\lambda$ to an input image $i$, we obtain a series of continuous-valued response maps $r_\omega^\lambda$, in which $r_\omega^\lambda(w,h) \in \mathbb{R}$ indicates the presence of a contrast edge with orientation $\omega \in [0°, \dots, 180°)$ and scale $\lambda \in \mathbb{R}$ at location $(w,h) \in \{0,\dots,W-1\} \times \{0,\dots,H-1\}$ (see Figure 4.2). Each image pixel $(w,h)$ can now be associated with a fixed-size array $R(w,h) = (r_{\omega_1}^{\lambda_1}(w,h),\dots,r_{\omega_\Omega}^{\lambda_1}(w,h);\dots;r_{\omega_1}^{\lambda_\Lambda}(w,h),\dots,r_{\omega_\Omega}^{\lambda_\Lambda}(w,h)) \in \mathbb{R}^{\Lambda\Omega}$, having a length of $L = \Lambda\Omega \in \mathbb{N}$ and containing all edge filter response values $r_\omega^\lambda(w,h)$. In theory, the same procedure could then be applied to the entire image $i$, by concatenating all of these arrays $R(w,h) \in \mathbb{R}^L$ to form a single feature vector $f = R(i) = (R(0,0),\dots,R(0,W-1);\dots;R(H-1,0),\dots,R(H-1,W-1)) \in \mathbb{R}^{WHL} = \mathcal{F}$, representing the image $i$ in the feature space $\mathcal{F}$ as defined by the Gabor filters.

While this approach preserves all of the available information, the resulting feature vectors would be ill-suited for learning the behavior classifiers when fed into our scene categorization architecture, however: First, the above feature space is generally too large to be tractable, as the optimization process underlying our learning procedure essentially performs an exhaustive greedy search for the most discriminative feature vector dimensions (see Chapter 3). Second, the pixel-level response values are inherently subject to noise, which impedes the ability of the classifiers to find consistent patterns throughout the entire training set. Regular image grids are a common technique for addressing both issues simultaneously.

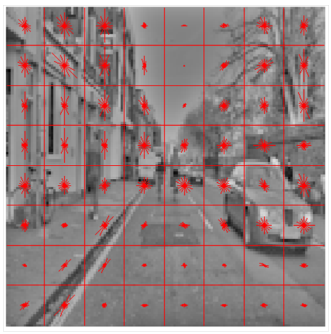


Figure 4.3: Example image shown with rectangular grid cells on top (red boxes), using an 8x8 image grid. The edge orientation histograms (red bars) summarize the pixel values of all response maps within each grid cell (only one scale shown).

A regular image grid $G_{M\times N}$ consists of $C = MN \in \mathbb{N}$ rectangular grid cells that cover the entire image domain $D = \{0,\ldots,W-1\}\times\{0,\ldots,H-1\}$ without overlapping. Specifically, each grid cell $G_{M\times N}(m,n) \subseteq D$ contains $\lfloor\frac{W}{M}\rfloor\lfloor\frac{H}{N}\rfloor \in \mathbb{N}$ image pixels as given by $G_{M\times N}(m,n) = \{(w,h) \in D | (m-1)\frac{W}{M} \leq w < m\frac{W}{M} \wedge (n-1)\frac{H}{N} \leq h < n\frac{H}{N}\}$, where $m \in \{0,\ldots,M-1\}$ and $n \in \{0,\ldots,H-1\}$. By construction, we have $G_{M\times N}(m,n) \cap G_{M\times N}(m',n') = \emptyset$ for grid cells with $m \neq m' \in \{0,\ldots,M-1\}$ and $n \neq n' \in \{0,\ldots,H-1\}$, and we also have $(w,h) \in G_{M\times N}(m^*,n^*)$ for some $m^* \in \{0,\ldots,M-1\}$ and $n^* \in \{0,\ldots,N-1\}$. To abstract from the response values $R(w,h) = (r_1(w,h),\ldots,r_L(w,h)) \in \mathbb{R}^L$ at the level of individual pixels $(w,h) \in D$, a single edge orientation histogram $H_{M\times N}(m,n) \in \mathbb{R}^L$ is then computed for each grid cell $G_{M\times N}(m,n)$, by setting $H_{M\times N}(m,n) = (\sum_{(w',h')\in G_{M\times N}(m,n)} r_1(w',h'),\ldots,\sum_{(w',h')\in G_{M\times N}(m,n)} r_L(w',h'))$.

Intuitively, $H_{M\times N}(m,n) \in \mathbb{R}^L$ correlates with the presence of contrast edges at orientation $\omega$ and scale $\lambda$ anywhere within the corresponding image grid cell $G_{M\times N}(m,n)$, without further specifying their exact locations (see Figure 4.3). The entire image can thus be summarized by $C = MN \in \mathbb{N}$ continuous-valued edge orientation histograms $H_{M\times N}(m,n) \in \mathbb{R}^L$, which are further serialized into a sparse feature vector $f_{M\times N} = R_{M\times N}(i) = (H_{M\times N}(0,0),\ldots,H_{M\times N}(0,N-1);\ldots;H_{M\times N}(M-1,0),\ldots,H_{M\times N}(M-1,N-1)) \in \mathbb{R}^{MNL} = \mathfrak{F}_{M\times N}$. Since $MNL << WHL \in \mathbb{N}$ is generally satisfied in practice, $f_{M\times N}$ is a much more compact representation of the input image $i$ than $f$, and hence more feasible.

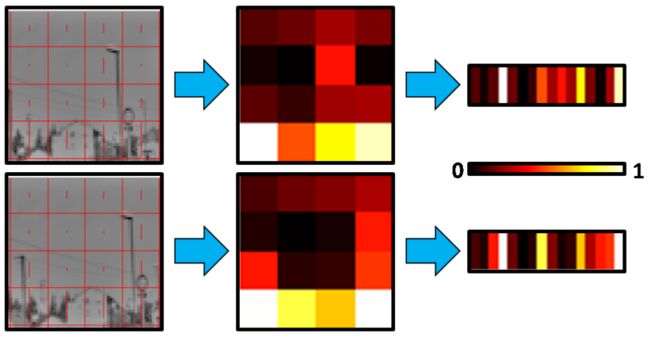


Figure 4.4: Toy example showing an undesired side-effect that is induced by the discretization of the image domain into grid cells (using a 4x4 image grid here): Slightly shifting the image (left) can greatly alter its feature vector (right).

### 4.1.3 Gaussian Kernel Weighting

As explained in the previous section, pooling the response values $R(w,h) \in \mathbb{R}^L$ at individual pixels $(w,h) \in D$ within regular image grid cells $G_{M\times N}(m,n) \subseteq D$ results in a rather low-dimensional feature vector representation $f_{M\times N} \in \mathbb{R}^{MNL}$ of the input image $i$. At the same time, the values in $f_{M\times N}$ are stabilized by the average computations of the pooling process, and also exhibit a certain degree of invariance to small shifts in the exact locations of the response values $R(w,h)$. However, the application of regular image grids also introduces artifacts into the feature vector representation, as a direct consequence of its spatial discretization. To see this, consider the case in which a strong response $r^* \in R(w^*,h^*)$ is located in a grid cell $G_{M\times N}(m^*,n^*)$. Such a response could be caused by a sign post, for example, especially when it is highly contrasting with the background. While driving along, the pole will apparently change its location within the field of view, until it enters some adjacent grid cell $G_{M\times N}(m^+,n^+)$ with $(m^+,n^+) \neq (m^*,n^*)$. Inevitably, the feature vector $f_{M\times N}$ undergoes a sudden change at this point, since $r^*$ now fully contributes to the respective histogram value in $H_{M\times N}(m^+,n^+)$ instead of $H_{M\times N}(m^*,n^*)$. As a consequence, the feature vector representation suggests the situations right before and after the transition to be more dissimilar than they actually are (see Figure 4.4). This phenomenon is a general drawback of histogram-based representations, and typically leads to a slight but significant distortion in the values of the resulting feature vectors.

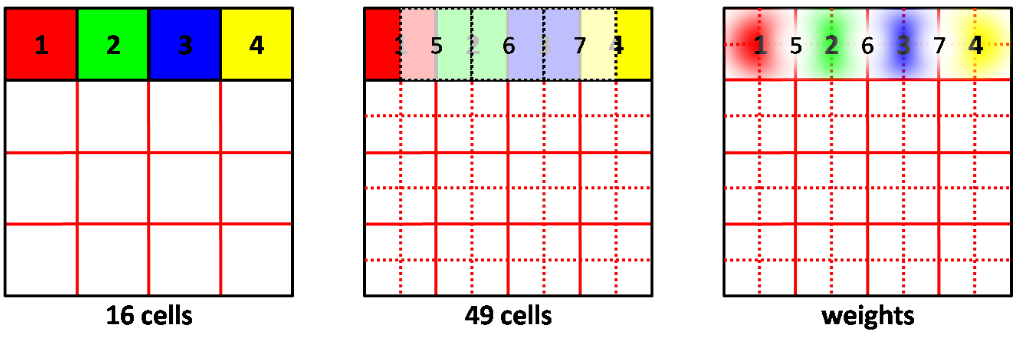


Figure 4.5: Standard image grid (left), and extended image grid with overlapping grid cells (center). Each of these grid cells is weighted by a Gaussian kernel to suppress borderline response values (right, showing one-dimensional example).

To further stabilize the feature vectors $f_{M\times N} \in \mathbb{R}^{MNL}$, we therefore employ the same technique as proposed in [84], which extends an image grid $G_{M\times N}$ by Gaussian kernels $\Gamma : \{0, \dots, \lfloor \frac{W}{M} \rfloor - 1\} \times \{0, \dots, \lfloor \frac{H}{N} \rfloor - 1\} \to \mathbb{R}$. More specifically, we define that $\Gamma(w', h') = \mathcal{N}_{\mu,\sigma}(w', h')$ for all $w' \in \{0, \dots, \lfloor \frac{W}{M} \rfloor - 1\}$ and $h' \in \{0, \dots, \lfloor \frac{H}{N} \rfloor - 1\}$, where $\mathcal{N}_{\mu,\sigma}$ is a two-dimensional Gaussian function with mean $\mu = (\frac{1}{2}\lfloor \frac{W}{M} \rfloor, \frac{1}{2}\lfloor \frac{H}{N} \rfloor)$ and standard deviation $\sigma = (\frac{1}{4}\lfloor \frac{W}{M} \rfloor, \frac{1}{4}\lfloor \frac{H}{N} \rfloor)$. This way, $\Gamma$ exactly matches the size of a grid cell $G_{M\times N}(m, n)$, with a Gaussian at its center that assumes close-to-zero values at the boundaries. When multiplied to the pixel-level responses $R(w', h')$ in $G_{M\times N}(m, n)$, the histogram computation gives stronger weight to responses that are located closer to the center of $G_{M\times N}(m, n)$, while giving less weight to responses at its boundaries. Then, $H_{M\times N}(m, n) = (\sum_{(w',h') \in G_{M\times N}(m,n)} \Gamma(w', h') r_1(w', h'), \dots, \sum_{(w',h') \in G_{M\times N}(m,n)} \Gamma(w', h') r_L(w', h'))$.

As the histogram representations using such Gaussian kernel weights, however, also become effectively "blind" to any observations close to grid cell boundaries, the authors propose to incorporate additional grid cells at regular intervals, located "in between" the original grid cells. The additional grid cells are again weighted by Gaussian kernels (see Figure 4.5), and explicitly capture responses close to the boundaries of the original grid cells that would otherwise be lost. Note that the histogram representation increases in size, regarding the number of feature values obtained, which is due to the increased number of grid cells (roughly doubling in the process).

## 4.2 Object Classifier Responses

While we generally agree with the assumption of the filter-based representation that pre-attentive visual processing at the level of raw image filter responses contributes to visual scene understanding in human drivers, we also hypothesize that higher cognitive functions play an important role as well. For example, humans are capable of analyzing and interpreting traffic scenes in terms of their constituent objects and semantic entities, which introduces an abstraction layer in between the raw image filter responses and the resulting feature vectors. Our semantic object-level representation of traffic scenes explicitly takes into account such information, considering a wide variety of different object types.

More specifically, this abstraction is achieved by the supervised learning of object classifiers $\mathcal{O}_1, \ldots, \mathcal{O}_O$ that effectively decompose each traffic scene into its constituent semantic entities. The semantics are grounded in human perception, since the training procedure relies on manually annotated ground truth images showing the different object types that are typically encountered in urban traffic. To determine which of the resulting features are actually relevant for predicting the appropriate driving behavior, in the various types of traffic scenes as defined by the behavior classes of our scene categorization architecture, is left for the system to learn without further supervision (see Chapter 3).

### 4.2.1 Object Recognition

Fundamental to our object-level representation of traffic scenes is the ability to detect and localize potentially relevant objects in an image $i : D \to \{0, \ldots, 255\}^3$ that depicts the current traffic situation. To this end, we train a broad range of object classifiers $\mathcal{O}_1, \ldots, \mathcal{O}_O$ each of which is specific for a particular object class, where $O \in \mathbb{N}$ denotes the total number of object classes considered. In practice, these object classes include static scene elements with large spatial extent, such as the road and sidewalks, dynamic scene elements that exhibit independent motion, such as other cars and pedestrians, and symbolic scene elements that have some pre-defined meaning, such as lane markings, for example (see Chapter 5). How can these objects be described in terms of their characteristic features, such that we can detect and localize them in previously unseen images $i$?

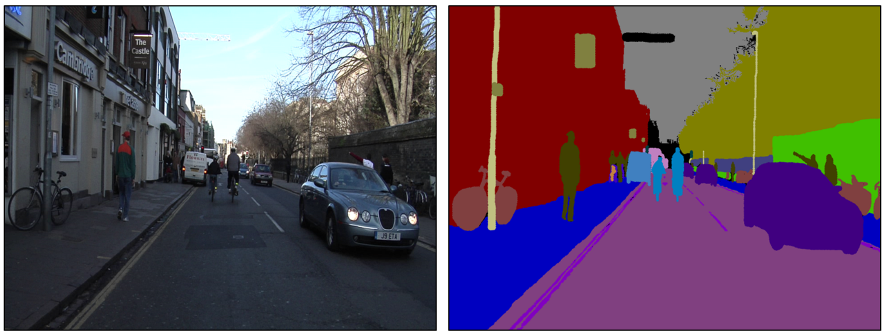


Figure 4.6: A typical urban traffic scene (left) taken from the CamVid dataset [14] that was recorded in the UK, and its manually annotated object labels (right). The different colors are globally defined to indicate the different object types.

To this end, we assume that we have an independent set $\mathcal{I} = \{i_1, \ldots, i_P\}$ of training images $i_p : D \to \{0, \ldots, 255\}^3$ in RGB color space, each of which is depicting a real-world traffic scene, and a corresponding set $\mathcal{L} = \{l_1, \ldots, l_P\}$ of label images $l_p : D \to \{0, \ldots, O\}$ that are obtained by manually annotating the training images $i_p \in \mathcal{I}$ to serve as ground truth for the object classifiers, where $p \in \{1, \ldots, P\}$. Intuitively, $l_p \in \mathcal{L}$ indicates for each pixel $(w, h) \in D$ in the respective image $i_p \in \mathcal{I}$ which object class it belongs to, with $l_p(w, h) = o$ if $(w, h) \in D$ belongs to an object of class $o \in \{1, \ldots, O\}$. The label images $l_p \in \mathcal{L}$ therefore specify both the class-membership and the location of objects, and actually represent a full segmentation of the $i_p \in \mathcal{I}$ (see Figure 4.6). Though expensive a process, labeling tools can be used to create $\mathcal{L}$ from $\mathcal{I}$ [14].

By sampling local image patches $\pi_{w,h} = \{(w', h') \in D | (w - X \leq w' \leq w + X) \wedge (h - Y \leq h' \leq h + Y)\} \subset D$ with $X, Y \in \mathbb{N}$ from the training images $i_p \in \mathcal{I}$, and assigning a single object label $l_p(\pi_{w,h}) = \text{argmax}_{o \in \{1,\ldots,O\}}\{\#(\pi_{w,h}, o)\}$ to each $\pi_{w,h}$ by majority voting, we end up with a set of patches $\Pi_o = \{\pi_{w,h} | l_p(\pi_{w,h}) = o\}$ per object class. Specifically, $\#(\pi_{w,h}, o) = |\{(w', h') \in \pi_{w,h} | l_p(w', h') = o\}|$ denotes the number of pixels $(w', h') \in \pi_{w,h}$ labeled as $o \in \{1, \ldots, O\}$. Given $\Pi_1, \ldots, \Pi_O$, we can train each object classifier $\mathcal{O}_o$ in a one-versus-all manner, where $\pi_+ \in \Pi_o$ are the positive examples and $\pi_- \in \bigcup_{o' \in \{1,\ldots,O\} \setminus \{o\}} \Pi_{o'}$ are the negative examples, using GentleBoost classifiers again (see Chapter 3). Thus, our object classifiers are trained like the behavior classifiers, but on patches rather than images.

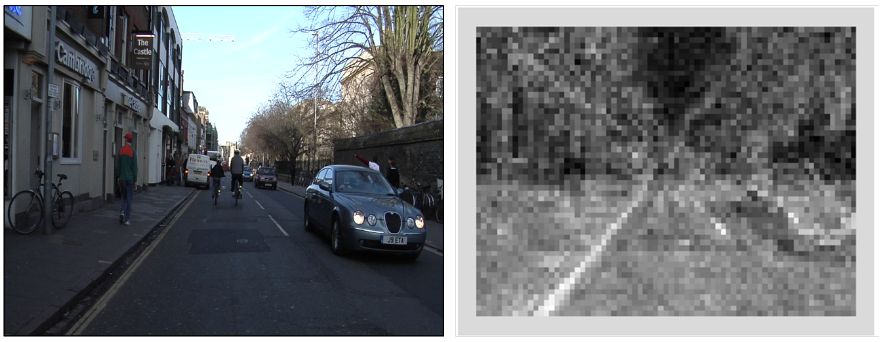


Figure 4.7: Input image (left), and one of the continuous-valued response maps that result from the application of the object classifiers (right). The map shown corresponds to lane markings, detected without using edge orientations at all.

### 4.2.2 Scene Decomposition

Given the trained object classifiers $\mathcal{O}_1, \dots, \mathcal{O}_O$, we can then apply each $\mathcal{O}_o$ to a previously unseen image $i : D \to \{0, \dots, 255\}^3$ of the current traffic situation, which results in a continuous-valued object classifier response map $c_o : D \to \mathbb{R}$ (see Figure 4.7). More specifically, each classifier response value $c_o(w, h) \in \mathbb{R}$ indicates the similarity of the patch $\pi_{w,h} \subset D$ at location $(w, h) \in D$ to the positive training examples $\pi_+ \in \Pi_o$ of object class $o \in \{1, \dots, O\}$, and a high classifier response value $c_o(w, h) \in \mathbb{R}$ thus correlates with the presence of an object of class $o \in \{1, \dots, O\}$ at location $(w, h) \in D$ while a low value suggests the absence of such an object. Since we have $O \in \mathbb{N}$ classifier response maps $c_o : D \to \mathbb{R}$ for each input image $i : D \to \{0, \dots, 255\}^3$, we also have $O \in \mathbb{N}$ classifier response values $c_o(w, h) \in \mathbb{R}$ at each image pixel $(w, h) \in D$. Therefore, the most likely object class at location $(w, h) \in D$ is given by the object classifier $\mathcal{O}_o$ with the strongest response $c_o(w, h) \in \mathbb{R}$ at that pixel, using a winner-take-all scheme. Like the label images $l_p \in \mathcal{L}$, the resulting map $s : D \to \{1, \dots, O\}$ as obtained by setting $s(w, h) = \text{argmax}_{o \in \{1,\dots,O\}}\{c_o(w, h)\}$ for each $(w, h) \in D$ is a segmentation of $i$. Intuitively, this segmentation can therefore be used to visualize the object classifier output on $i$. In principle, $s$ could also be processed to form a feature vector and then be fed into our scene categorization architecture for driving behavior prediction (see Chapter 3), as in our original approach [43].

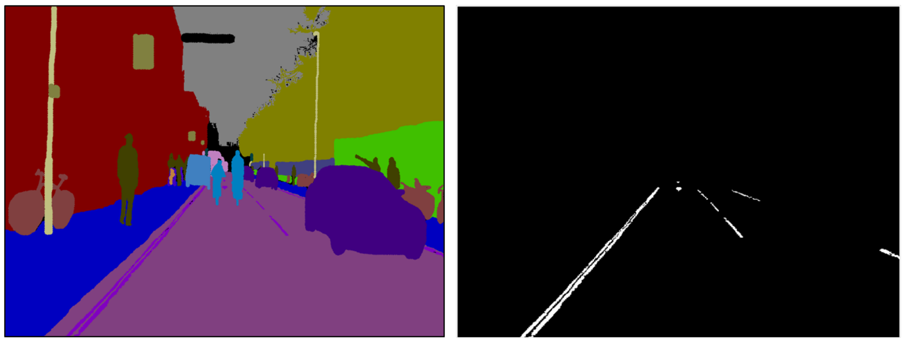


Figure 4.8: Given a segmentation of the scene (left), binary maps can be extracted to separate the information by object type, as shown for lane markings (right). Note, however, that our current approach does not rely on binary maps anymore.

To this end, we have derived a series of binary maps $s_1, \ldots, s_O : D \to \{0, 1\}$ from $s$ by setting $s_o(w, h) = 1$ if $s(w, h) = o$ and $s_o(w, h) = 0$ if $s(w, h) \neq o$, for all $(w, h) \in D$. The segmentation data of $s$ is thus split by object type, with each $s_o$ now indicating only the locations of objects in class $o \in \{1, \ldots, O\}$ (see Figure 4.8). The binary maps $s_o$ were then post-processed by computing their average response values in regular image grid cells, possibly with Gaussian kernel weighting, for the same purpose of dimensionality and noise reduction as with the post-processing of the raw image filter response maps $r_1, \ldots, r_L$ in the previous section. Comparing the $S(w, h) = (s_1(w, h), \ldots, s_O(w, h)) \in \mathbb{R}^O$ to the $R(w, h) = (r_1(w, h), \ldots, r_L(w, h)) \in \mathbb{R}^L$, however, reveals that the latter preserves the entire distribution of oriented edge filter responses, while the former only represents the final decision $s(w, h) = o \in \{1, \ldots, O\}$ without the underlying distribution of classifier responses: In particular, each $S(w, h) \in \mathbb{R}^O$ has a single non-zero value, which always equals 1. As with the confidence measures that we have discussed in the context of our behavior classifiers (see Chapter 3), however, an object label should be regarded as more confident if the maximum classifier response value is significantly higher than all others, and less confident if their difference is not particularly pronounced. To incorporate this kind of information into our feature vectors, the semantic object-level representation in this work is always computed directly from the object classifier response maps $c_1, \ldots, c_O$, rather than from the segmentation $s$ and its binary maps $s_1, \ldots, s_O$.

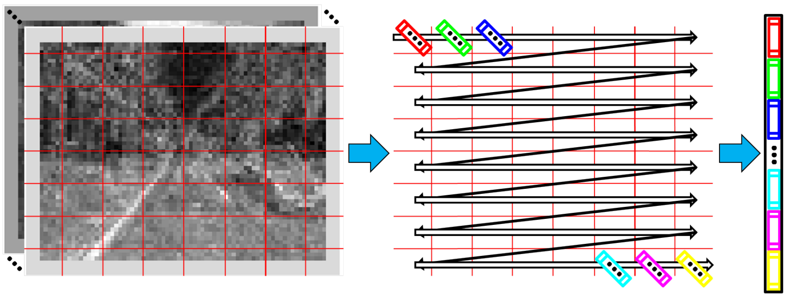


Figure 4.9: Our traffic scene representation is derived from the object classifier response maps (left), by computing a response histogram for each grid cell (center) and serializing all of the resulting histograms to form a large feature vector (right).

### 4.2.3 Feature Vector Formation

Given the continuous-valued response maps $c_1, \ldots, c_O : D \to \mathbb{R}$ that result from applying the object classifiers $\mathcal{O}_1, \ldots, \mathcal{O}_O$ to input image $i : D \to \{0, \ldots, 255\}^3$, we now compute a single feature vector $f' = C(i) \in \mathbb{R}^Z = \mathcal{F}'$ to represent $i$, where $Z \in \mathbb{N}$ is the dimensionality of feature space $\mathcal{F}'$. The response maps $c_1, \ldots, c_O$ are technically equivalent to the continuous-valued response maps $r_1, \ldots, r_L$ as obtained by applying the oriented edge filters $e_\omega^\lambda$ to $i$, which have been introduced in the context of the filter-based representation. We thus perform the same average computations over regular image grid cells $G_{M\times N}(m,n) = \{(w,h) \in D | (m-1)\frac{W}{M} \leq w \leq m\frac{W}{M} \wedge (n-1)\frac{H}{N} \leq h \leq \frac{H}{N}\}$ on each $c_o$, with a Gaussian kernel $\Gamma : \{0, \ldots, \lfloor \frac{W}{M} \rfloor - 1\} \times \{0, \ldots, \lfloor \frac{H}{N} \rfloor - 1\} \to \mathbb{R}$ centered at each grid cell $G_{M\times N}(m,n)$ to define weights (see Figure 4.9). Specifically, we have $\Gamma(w,h) = \mathcal{N}_{\mu,\sigma}(w,h) \in \mathbb{R}$ with $\mu = (\frac{1}{2}\lfloor \frac{W}{M} \rfloor, \frac{1}{2}\lfloor \frac{H}{N} \rfloor)$ and $\sigma = (\frac{1}{4}\lfloor \frac{W}{M} \rfloor, \frac{1}{4}\lfloor \frac{H}{N} \rfloor)$ as before, and $W, H \in \mathbb{N}$ are the width and height of domain $D$ while $M, N \in \mathbb{N}$ are the horizontal and vertical number of grid cells, respectively, with $(m,n) \in \{1, \ldots, M\} \times \{1, \ldots, N\}$. The histograms over all corresponding $G_{M\times N}(m,n)$ in $c_1, \ldots, c_O$, i.e., $H'_{M\times N}(m,n) = (\sum_{(w',h') \in G_{M\times N}(m,n)} \Gamma(w',h') c_1(w',h'), \ldots, \sum_{(w',h') \in G_{M\times N}(m,n)} \Gamma(w',h') c_O(w',h'))$, are then serialized to form $f'_1 = (H'_{M\times N}(0,0), \ldots, H'_{M\times N}(M-1, N-1)) \in \mathbb{R}^{MNO}$, which is a partial feature vector whose dimensionality is comparable to that of $f_{M\times N} \in \mathbb{R}^{MNL}$, which represents $i$ in the filter-based feature space $\mathcal{F}_{M\times N}$. Importantly, however, $f'_1 \in \mathbb{R}^{MNO}$ represents semantic information about $i$.

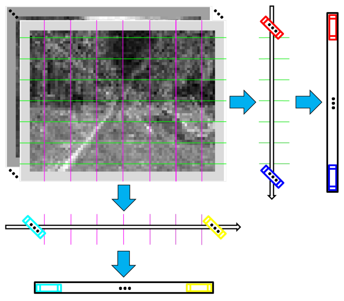


Figure 4.10: In addition to the feature vector that is obtained by applying the standard image grid cells, we also compute a row feature vector (right) and a column feature vector (bottom). These are appended to the main feature vector.

Operating at the level of objects, we can also incorporate information about the spatial distribution of traffic scene elements with respect to additional types of rectangular image grid cells. Following the idea of Ess et al. [30], we define elongated row cells $G_{\text{row}}(1), \ldots, G_{\text{row}}(P)$ and column cells $G_{\text{col}}(1), \ldots, G_{\text{col}}(Q)$ that span the entire width $W \in \mathbb{N}$ and height $H \in \mathbb{N}$ of $i$, respectively. Formally, this is achieved by setting $G_{\text{row}}(p) = \{(w', h') \in D | (p-1)\lfloor \frac{H}{P} \rfloor \leq h' \leq p \lfloor \frac{H}{P} \rfloor\}$ and $G_{\text{col}}(q) = \{(w', h') \in D | (q-1)\lfloor \frac{W}{Q} \rfloor \leq w' \leq q \lfloor \frac{W}{Q} \rfloor\}$, for all $p \in \{1, \ldots, P\}$ and $q \in \{1, \ldots, Q\}$. We then compute the average response values $F_{\text{row}}(p) = (\sum_{(w',h') \in G_{\text{row}}(p)} c_1(w', h'), \ldots, \sum_{(w',h') \in G_{\text{row}}(p)} c_O(w', h'))$ as row features, and the average responses $F_{\text{col}}(q) = (\sum_{(w',h') \in G_{\text{col}}(q)} c_1(w', h'), \ldots, \sum_{(w',h') \in G_{\text{col}}(q)} c_O(w', h'))$ as column features, which are serialized to form another partial feature vector $f'_2 = (F_{\text{row}}(1), \ldots, F_{\text{row}}(P); F_{\text{col}}(1), \ldots, F_{\text{col}}(Q))$ (see Figure 4.10). Row features correlate with the distance of objects and hence are relevant for the velocity, while column features correlate with the lateral position of obstacles, which is related to steering. A third feature type is derived from the classifier reponse map $c_{o^*}$ for lane markings, by applying oriented edge filters $e_\omega^\lambda$, computing histograms $H^{\text{ori}}_{M \times N}(m, n) = (\sum_{(w',h') \in G_{M \times N}(m,n)} l_1(w', h'), \ldots, \sum_{(w',h') \in G_{M \times N}(m,n)} l_{\Omega\Lambda}(w', h'))$, and serializing them to form $f'_3 = (H^{\text{ori}}_{M \times N}(0, 0), \ldots, H^{\text{ori}}_{M \times N}(M-1, N-1))$. Note that, unlike the filter-based representation, these features are actually computed at the object-level, even if using filters. Finally, the partial feature vectors are all combined to form $f' = (f'_1, f'_2, f'_3)$ to represent $i$.

## 4.3 Implementation Details

We have not yet specified how the patches $\pi_{w,h}$ of our object-level representation are themselves represented for the object classifiers $\mathcal{O}_1, \ldots, \mathcal{O}_O$. Just like the traffic scene representations in this chapter serve to represent images for our behavior classifiers (see Chapter 3), these patches need to be represented as well for our object classifiers. In this context, we have implemented two alternatives that mainly differ in terms of their choice of patch descriptors and post-processing. Specifically, the first approach employs an edge filter bank with a state-of-the-art Conditional Random Field (CRF), whereas the second approach is built around a computationally efficient Walsh-Hadamard filter bank.

### 4.3.1 Conditional Random Field

At the heart of our first approach to representing the patches $\pi_{w,h} \subset D$ for our object classifiers $\mathcal{O}_1, \ldots, \mathcal{O}_O$ lies the application of a filter bank that is similar to the bank of oriented edge filters $e_\omega^\lambda$ as discussed at the beginning of this chapter (see Figure 4.11). It also contains a variety of edge filters at multiple orientations and scales, but incorporates center-surround filters for blob detection as well. The application of this filter bank to an input image $i$ is performed separately for each of its color channels $i_R, i_G, i_B : D \to \{0, \ldots, 255\}$, which results in a series of continuous-valued response maps $m_1^R, \ldots, m_A^R; m_1^G, \ldots, m_A^G; m_1^B, \ldots, m_A^B : D \to \mathbb{R}$ akin to the response maps $r_1, \ldots, r_L : D \to \mathbb{R}$ of the filter-based representation.

To further increase the discriminative power of the patch descriptors $d(\pi_{w,h}) \in \mathbb{R}^B$ that are computed on this basis, it is common to cluster the response values $m_i^*(w,h) \in \mathbb{R}$ into a fixed set of textons $t_1, \ldots, t_C \in \mathbb{R}$. Intuitively speaking, these textons are used as representatives for the actual response values in $m_i^*$, and can be learned without supervision from example images such as the $i \in \mathcal{I}$ by running a k-means algorithm on the set of all $m_i^*(w,h) \in \mathbb{R}$ resulting from the training images $i \in \mathcal{I}$. After this learning process, previously unseen input images $i \notin \mathcal{I}$ can then be converted into a texton map $T : D \to \{t_1, \ldots, t_C\}$, by applying the filter bank in order to obtain the $m_i^R, m_i^G, m_i^B : D \to \mathbb{R}$ and replacing each response value $m_i^*(w,h)$ by the texton $t_i^*(w,h) = \operatorname{argmin}_{t \in \{t_1, \ldots, t_C\}} |t - m_i^*(w,h)|$ that represents it best, in a nearest-neighbor sense.

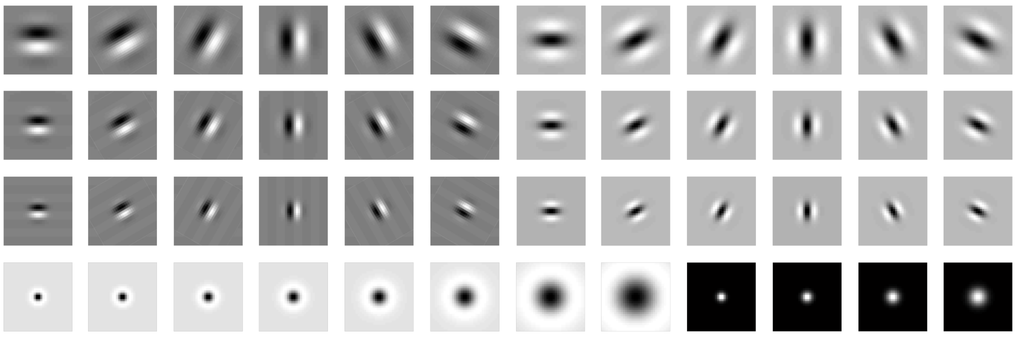


Figure 4.11: The Leung-Malik filter bank [56] on which the CRF implementation of our semantic object-level representation is based. In contrast to the filter-based traffic scene representation, object classifiers are learned from its response maps.

One technique to further increase the discriminatory power of the features, beyond the formation of textons, is to consider them in pairwise conjunctions. For example, cars typically drive on the road, and pedestrians are usually located along the sidewalks. As a consequence, observing features $d(\pi_{w,h}) \in \mathbb{R}^B$ that strongly suggest an image patch $\pi_{w,h} \subset D$ to belong to the road or the sidewalk at the same time gives evidence that image patches $\pi_{w',h'} \subset D$ close to $\pi_{w,h}$ might belong to a car or a pedestrian, respectively, if not to the road or sidewalk as well. Because such pairs of textons are potentially more discriminative than the individual textons alone, in the above sense, we systematically incorporate them into our object classifiers by extending the GentleBoost framework to implement shape filters [96]. Instead of operating on the textons $t \in \{t_1, \dots, t_C\}$ themselves, shape filters work by considering spatial constellations $s \in \mathbb{R}^3$ each of which is characterized by a triplet $s = (\phi, \rho, \xi)$. The parameters $\phi \in [-\pi, +\pi] \subset \mathbb{R}$ and $\rho \in [\theta_{\min}, \theta_{\max}] \subset \mathbb{N}$ denote the direction and the distance to the second location of the pair, while the first location is always set to the current pixel $\xi = (w, h) \in D$. Further, $\theta_{\min}, \theta_{\max} \in \mathbb{N}$ set the minimum and maximum distance between these two locations, respectively. In practice, the exact values for $\phi, \rho, \xi$ are randomly sampled from a pre-defined range of discrete sets during training, thus generating candidate pairs (see Figure 4.12). From these candidate pairs, the subsequent boosting procedure determines their most discriminative subset as explained earlier (see Chapter 3 for details on the boosting procedure).

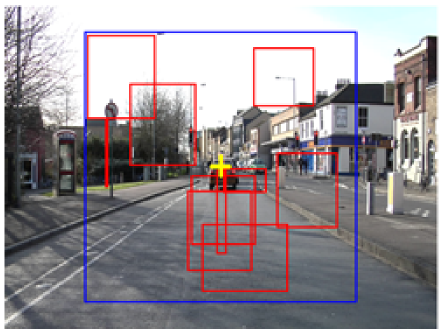


Figure 4.12: The principle of shape filters is to consider pairwise constellations as potential features. Like in the example [61], random rectangles (red) are sampled at each location (yellow), serving as feature candidates for the boosting scheme.

The most essential operation that takes us from mere object classification to the benefits of CRF integration is the incorporation of the pairwise potentials, in addition to the unary potentials as given by the response maps of our object classifiers. In practice, we follow the standard procedure by adopting a contrast-sensitive Potts model, which essentially penalizes adjacent labels that do not match each other, thus giving rise to transitions from one object class to another and hence to object boundaries, unless they are supported by visual evidence in form of an edge. This does not necessarily prevent object boundaries from being maintained, even if there is little such evidence at a given location, since the pairwise potentials merely introduce a penalty to avoid implausible boundaries.

In effect, the pairwise potentials therefore act as a force that encourages the object boundaries to follow visible edges, but at the same time allows them to smoothly continue if there are gaps in the contours (i.e., object boundaries with occasional lack of visible edges, which often occur in real-world images due to inhomogeneous lighting conditions and background clutter, for example). The primary purpose of the pairwise potentials, and hence the CRF framework in general, is to stabilize the raw, noisy object classifier response maps such that local evidence is being obeyed where present and, as far as the limited range of pairwise potentials permits, to also propagate such evidence to less confident or ambiguous locations. As a consequence, we obtain contiguous regions for all scene elements, seamed by continuous object boundaries (see Figure 4.13).

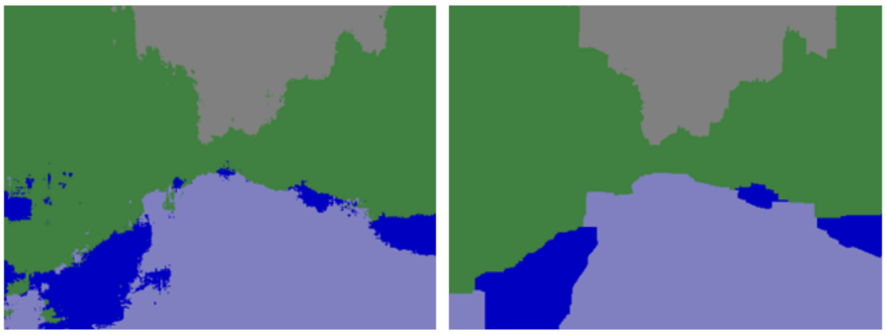


Figure 4.13: Example of a traffic scene decomposition that results from applying our Conditional Random Field [61], using only unary potentials (left) that directly correspond to the object classifier output, and pairwise potentials as well (right).

The CRF framework also enables us to integrate additional information about traffic scenes in particular, most prominently the typical spatial distributions of objects in the scene. While the existence of such constraints is not guaranteed for scenes in general, we have a fundamentally different situation due to the fixed camera focusing forward and sharing the same field of view as perceived by the human driver. For example, the road is typically at the bottom part of the scene, the sky is always above, buildings typically appear to the left and to the right of the road, and traffic participants can only occupy the lower part of the scene due to physical constraints. Mathematically, we can account for such spatial prior knowledge about the object types by probability maps (see Figure 4.14), spanning the entire field of view just like the filter and classifier response maps, with pixel values ranging from 0 to 1 to indicate the prior probability that an object of the respective type is observed at the given location. In practice, these probabilities can be learned offline, in parallel to learning the object classifiers, from the same manually annotated training data. Specifically, the frequency of a pixel labeled as belonging to an object class is being counted across the entire set of training images, and the resulting values are normalized within each such map to yield a probability distribution. In the end, the unary potentials from the object classifiers, the pairwise potentials from the Potts model, and the location potentials are all combined to determine an optimal segmentation, which is obtained by energy minimization using graph cuts (e.g., see [99]).

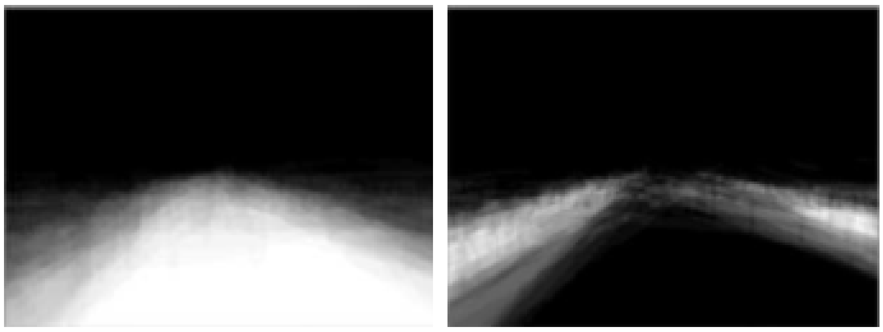


Figure 4.14: Examples of spatial prior knowledge about urban traffic scenes, learned from manually annotated label images. The left probability map shows typical locations of the road, the right one shows typical locations of sidewalks.

### 4.3.2 Walsh-Hadamard Features

The second approach to represent the patches $\pi_{w,h}$ for our object classifiers $\mathcal{O}_1, \ldots, \mathcal{O}_O$ relies on a different type of filter bank [42], which implements a Walsh-Hadamard transform when applied to the individual $\pi_{w,h} \subset D$. Unlike the previously discussed LM filter bank that mainly consists of oriented edge filters, and hence resembles the Gabor filter bank of the filter-based representation as discussed at the beginning of this chapter, the Walsh-Hadamard filter bank is sensitive to the spatial frequency of contrast edges rather than their orientations. Its filters can efficiently be implemented by means of binary operations alone, in contrast to the continuous-valued floating point operations that are involved when convolving an image with the aforementioned edge orientation filters, and this is a main reason why Walsh-Hadamard features have become a predominant texture descriptor for applications in the automotive domain (see Figure 4.15). The Walsh-Hadamard transform of an image can be seen as a binary version of the Discrete Cosine Transform, which is itself closely related to the continuous Fourier transform that is capable of approximating any given function or texture to an arbitrarily fine-grained degree. The accuracy to which this approximation is performed can be influenced by the number of coefficients computed for a patch: While computing all Walsh-Hadamard coefficients enables lossless reconstruction, it is practical to restrict their computation to the first 16 coefficients, for example, since higher coefficients are increasingly subject to the image noise anyway.

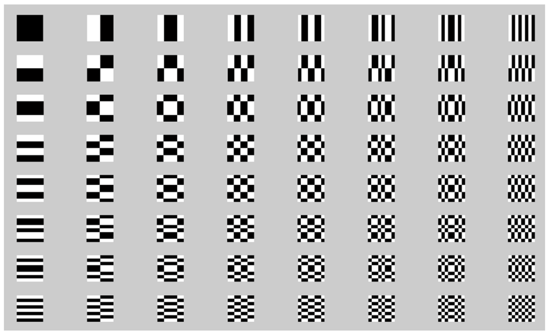


Figure 4.15: The Walsh-Hadamard transform of an image patch can efficiently be implemented by a filter bank with only binary operations. These filters are sensitive to the local frequencies of the patch texture, from coarse- to fine-grained.

Specifically, we first convert the label images $l \in \mathcal{L}$ of the CamVid dataset [14] to reflect the $O = 9$ object classes that we actually consider in our experiments (see Chapter 5), resulting in a corresponding set of simplified label images $l' \in \mathcal{L}'$ that only represent the object classes of interest and in particular match the labeling used by the state-of-the-art method [99]. The second pre-processing step consists in converting the camera images $i \in \mathcal{I}$ from RGB color space to the metric $L^*a^*b^*$ color space, thereby effectively decoupling the individual color channels from each other as well as separating the luminance component $L^* : D \to \mathbb{R}$ from the color component, which is in turn being represented along two orthogonal color opponency axes to yield $a^*, b^* : D \to \mathbb{R}$. Further, we apply the well-known gray-world assumption to the resulting $L^*a^*b^*$ images $i' \in \mathcal{I}'$, essentially stating that all channels $L^*, a^*, b^*$ of an image $i' \in \mathcal{I}'$ should individually be normalized to have zero mean and unit variance [30]. In practice, this can be achieved by computing the global average $\text{avg}(L) = \frac{1}{WH} \sum_{(w,h) \in D} L(w,h)$ as well as the variance $\text{var}(L) = \frac{1}{WH} \sum_{(w,h) \in D} (L(w,h) - \text{avg}(L))^2$ of $L$, and by then setting $L(w,h) = (L(w,h) - \text{avg}(L))/\text{var}(L)$ for all pixels $(w,h) \in D$ (and analogously for $a^*$ and $b^*$). This step is generally considered an important contributor to robustness against fluctuations in the visual appearance of objects in the scene, in particular with respect to illumination changes that are frequently occuring when driving in unconstrained real-world environments, as is the main scope of the present work.

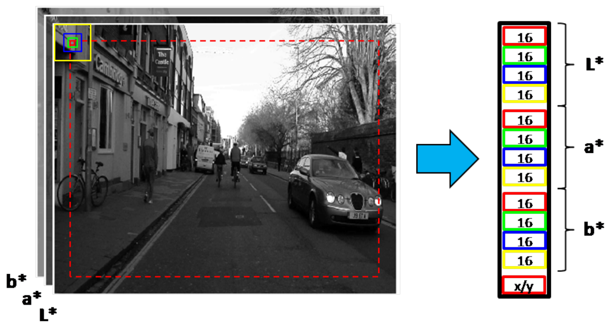


Figure 4.16: Illustration of the way how we sample patches from an input image: Within the image (dashed red box), 4x4 patches are considered (solid red box), with 8x8 (green), 16x16 (blue), and 32x32 patches (yellow) in addition.

Finally, we densely sample patches $\pi_{w,h}^{(U \times V)} = \{(w', h') \in D | (w \leq w' \leq w + U - 1) \wedge (h \leq h' \leq h + V - 1)\}$ of size $U = V = 4 \in \mathbb{N}$ from each of the gray-world images $i'' \in \mathcal{I}''$, starting in the upper left corner of $i''$ with the patch $\pi_{(1)+14,(1)+14}^{(4x4)}$, and ending in the lower right corner of $i''$ with the patch $\pi_{(W-U+1)-14,(H-V+1)-14}^{(4x4)}$, where the offset of 14 pixels allows for the incorporation of larger patches as well, as explained further below. The pre-computed Walsh-Hadamard filter bank is then applied to each of these 4x4 patches $\pi_{w,h}^{(4x4)} \in D$, which results in a fixed number of 16 Walsh-Hadamard coefficients for each of the $L^*a^*b^*$ channel, and the $3 * 16 = 48$ values thus obtained are serialized to form a partial feature vector $d'(\pi_{w,h}) \in \mathbb{R}^{48}$ that fully describes $\pi_{w,h}^{(4x4)}$ without any loss of information. In addition, and following well-established practice (e.g., [30, 123]), we also consider larger patches of size 8x8, 16x16, and 32x32, all centered at their corresponding basic patches $\pi_{w,h}^{(4x4)} \in D$ (see Figure 4.16). The reason is to also consider the larger spatial context of each of the basic 4x4 patches, and by applying the Walsh-Hadamard filter bank to each of these additional patches $\pi_{w,h}^{(8x8)}, \pi_{w,h}^{(16x16)}, \pi_{w,h}^{(32x32)} \subset D$ as well, and serializing their first 16 coefficients again, respectively, we end up with an extended feature vector $d_{w,h} \in \mathbb{R}^{(3*16)*4}$ that describes the local texture at position $(w, h) \in D$ in the field of view. Instead of using location maps that are learned from the ground truth per object class, as in the CRF implementation discussed before, we now directly incorporate the patch locations $(w, h) \in D$ into their corresponding feature vectors $d(\pi_{w,h}) \in \mathbb{R}^{194}$, having 194 feature dimensions.

# Chapter 5

# Performance Analysis

In this chapter, we quantitatively evaluate the behavior prediction methods that were discussed in Chapter 3 and Chapter 4. In particular, we conduct a detailed comparison of our own approach to the best vision-based correlation method currently known for driving behavior prediction [84], and we also investigate into the impact of different weather conditions on the prediction accuracy. To our knowledge, no such comparison has been done before, and we are also the first to systematically analyze the effect of weather conditions in this context.

Our experiments are conducted on a challenging video dataset [36], acquired by a car-mounted camera while driving naturally in real-world urban traffic. Its total duration is over 1 hour, and five different weather conditions are covered by separate streams. Apart from the behavior prediction experiments, we also perform a stand-alone evaluation of the segmentation techniques underlying our traffic scene representation (see Chapter 4), including a quantitative comparison to the results achieved by state-of-the-art segmentations [15, 99]. Importantly, all datasets are publicly available such that others are able to reproduce our results.

We begin with a discussion of public video datasets from the perspective of driving behavior prediction, focusing in particular on the CamVid dataset [14] and the HRI dataset [36]. Subsequently, we report on the results of our segmentation experiments, which include both the CRF and the WH implementation (see Chapter 4). Finally, we turn to the actual behavior prediction and conduct our main experiments. As we shall see, our method achieves competitive results for yaw rate prediction, and outperforms the state-of-the-art on velocity prediction.

## 5.1 Video Datasets

Our comparative performance evaluation of different correlation methods for vision-based behavior prediction is an important part of our thesis work. While naturally limited to behavior prediction methods of today, we envision our work to serve as a starting point for other researchers to evaluate their own approaches under the same conditions. Common datasets have proven to be effective for driving scientific progress in other research fields such as object detection and recognition [31, 39], optic flow and stereo [4, 94], and image segmentation [14, 60]. It is our hope that the present work facilitates the formation of a common basis for evaluating behavior prediction methods as well. For this reason, it is vital that all datasets used in our experiments are also accessible to others.

We therefore consider a number of publicly available video datasets that were recorded by car-mounted cameras while driving naturally in real urban traffic. Specifically, these include the *Daimler Pedestrian Dataset* (Daimler) [35], the *Cambridge-driving Labeled Video Dataset* (CamVid) [14], the *HRI Road Traffic Dataset* (HRI) [36], and the *Caltech Pedestrian Dataset* (Caltech) [29]. All of them were created for applications other than behavior prediction, and hence require careful examination with regard to their suitability for our own experiments. In particular, these datasets exhibit a considerable variability in terms of their overall duration, number of streams, camera setup, and available ground truth, among others (see Table 5.1).

### 5.1.1 Overview

**CamVid.** The CamVid dataset consists of four different streams, recorded by a high-resolution color camera, and has a total duration of about 10 minutes. While this is too short for behavior prediction other than a proof-of-concept evaluation (see [43]), the advantage of this dataset lies in its rich annotations. Specifically, it comes with pre-computed structure-from-motion data, and manually labeled ground truth segmentation data of pixel-level accuracy. We therefore use the CamVid dataset for training the object classifiers underlying our traffic scene representation (see Chapter 4), and for conducting our segmentation experiments. Further details on this particular dataset are given in the following section.

| Dataset | Duration | Framerate | Resolution | Camera | Color | GT |
|---|---|---|---|---|---|---|
| **CamVid** [14] | 10 min | 1 fps | 960 x 720 | Mono | Yes | Yes |
| **Daimler** [35] | 30 min | 10 fps | 640 x 480 | Mono | No | No |
| **HRI** [36] | 1 hour | 10 fps | 400 x 300 | Stereo | Yes | Yes |
| **Caltech** [29] | 10 hour | 30 fps | 640 x 480 | Mono | Yes | No |

Table 5.1: Public video datasets, each recorded from within a moving vehicle while driving in real urban traffic.

**Daimler.** The Daimler dataset consists of a single stream whose duration of about 30 minutes is considerably longer than that of the CamVid dataset. In addition, it has a much higher framerate of approximately 10 fps. A serious drawback of the Daimler dataset, however, is its lack of ground truth information about the driver's behavior or the vehicle motion. Moreover, it was recorded by a monocular grayscale camera only, although color information is arguably relevant for traffic scene understanding. For these reasons, we do not use this dataset in any of our experiments and resort to the HRI dataset instead.

**HRI.** The HRI dataset consists of five different streams, which in turn correspond to five different weather and lighting conditions: sunny, overcast, raining, night, and snow. Its total duration is more than 1 hour of driving, which exceeds the previous datasets by far. Also, the streams were all acquired by a calibrated stereo camera instead of a monocular camera only, and they are available in color. What sets the HRI dataset apart from the other datasets, in the context of driving behavior prediction, is its being accompanied by ground truth behavior data that was directly recorded from the CAN bus of the moving vehicle. We therefore conduct our behavior prediction experiments on this particular dataset.

**Caltech.** The Caltech dataset is clearly the largest of the above datasets, having a total duration of about 10 hours. It consists of eleven video streams that were all recorded at a high framerate of 30 fps. As with the Daimler dataset, however, no ground truth behavior data is made available, hence such information would have to be inferred from the vehicle motion (e.g., by structure-from-motion like in the CamVid dataset). Moreover, only a monocular camera was used, making the computation of 3D features more difficult than with the stereo setup of the HRI dataset. Thus, the Caltech dataset it not an alternative for us.

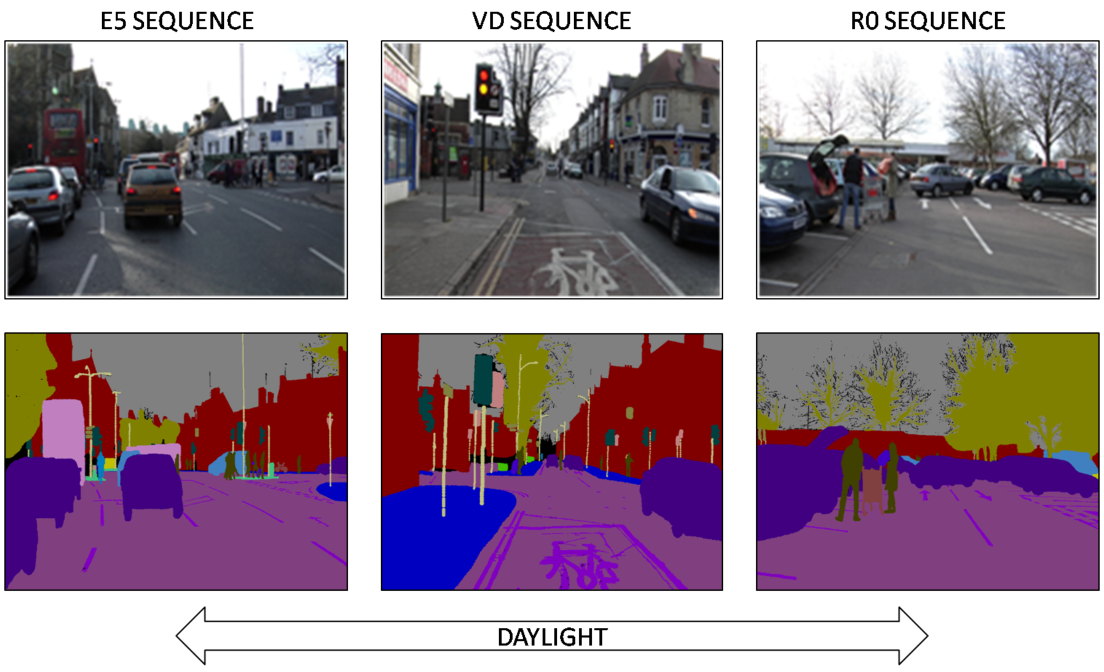


Figure 5.1: The CamVid dataset consists of four streams, recorded in the UK. Note that one of the streams corresponds to dusk conditions (see Figure 5.2).

### 5.1.2 CamVid Dataset

As pointed out in the previous section, we use the CamVid dataset for conducting our segmentation experiments, mainly because of the segmentation ground truth that accompanies the video streams. Specifically, the ground truth enables us to train and evaluate our object classifiers, whose activation maps are the basis of the semantic object-level representation used by our behavior prediction. As current state-of-the-art methods for object recognition and segmentation were evaluated on this dataset as well, we can quantitatively compare their reported performance to our own results. In the following, we therefore examine the CamVid dataset in more detail, giving examples of the video streams and the ground truth data, and identifying the most frequent object types.

Example images of the four video streams are shown in Figure 5.1 (top row) and Figure 5.2 (top left). As we can see, the vehicle is driving in typical city traffic that is characterized by large static objects such as the road, buildings, and trees

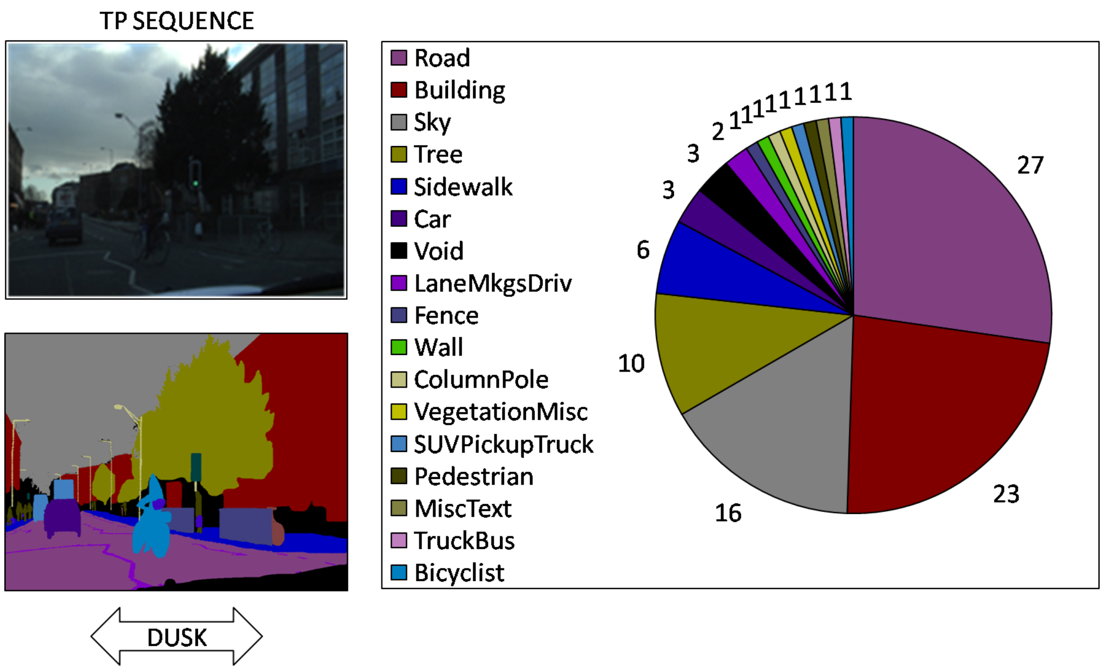


Figure 5.2: Over 30 object classes were manually annotated at the pixel-level. The most frequent ones are shown in the diagram. (numbers in percent)

as well as dynamic objects such as other cars, pedestrians, and bicyclists. The objects are all meticulously annotated in the corresponding segmentation images that are provided as ground truth, which are shown in Figure 5.1 (bottom row) and Figure 5.2 (bottom left). In particular, the accuracy of these annotations goes far beyond the bounding boxes or polygons that are typical for datasets used in object detection, and is beneficial for both training and evaluation.

Due to the abundance of object classes that are labeled in the CamVid dataset, a complete decomposition of the traffic scenes into their constituent semantic entities is actually provided. However, while some of the classes are predominant in most of the scenes (see Figure 5.2), others are either very rare or very small. For this reason, it is common practice to consider a subset of the most frequent object classes only, typically around ten, and to recombine some of the labels into larger classes. For example, the various types of vehicles that are annotated separately from each other are often merged to form a single class, representing cars in general without further distinction.

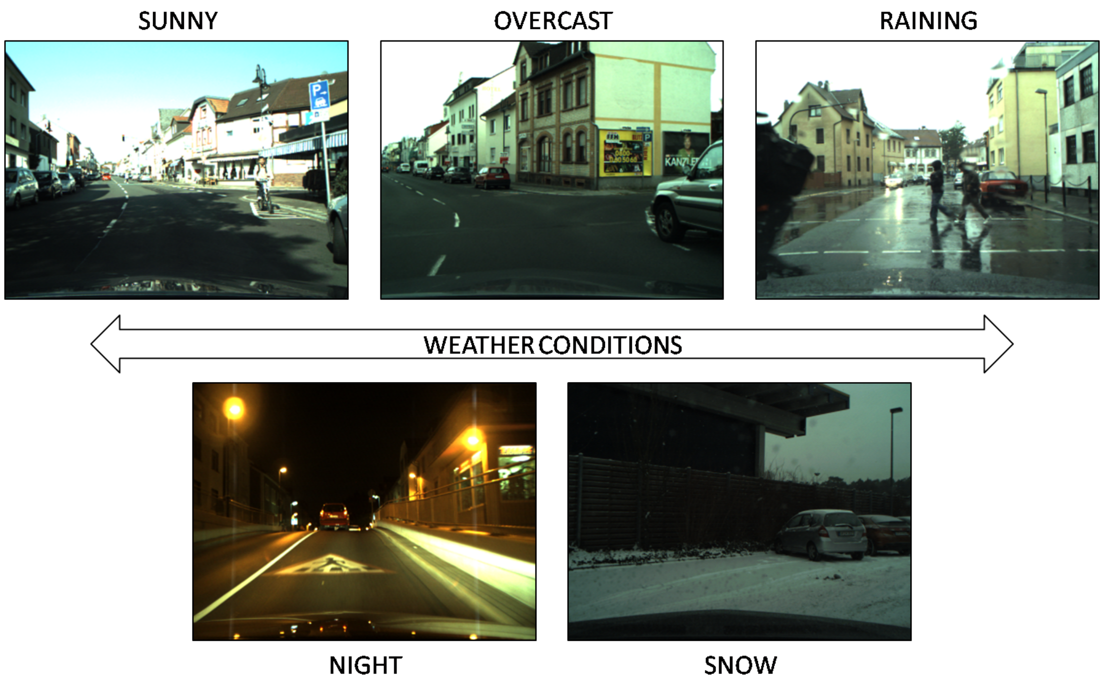


Figure 5.3: The HRI dataset consists of five streams, all recorded in Germany. Each stream corresponds to one of five different weather conditions.

### 5.1.3 HRI Road Traffic Dataset

In contrast to the CamVid dataset that was discussed in the previous section, the HRI dataset itself does not include any segmentation data as ground truth. However, the behavior data that is provided alongside with its video streams enables us to train and evaluate our behavior classifiers (see Chapter 3), and to directly compare their prediction accuracy to the state-of-the-art method [84]. The clear separation of the different weather conditions into different streams, moreover, makes it possible to systematically analyze how the visual appearance of the traffic scenes affects the performance of the two approaches. We therefore take a closer look at the video streams and their corresponding behavior data.

Example images of the video streams, also showing their weather conditions, are given in Figure 5.3. Note the considerable influence of the weather on the visual appearance of the traffic scenes: While the road surface, for example, appears homogeneous in the overcast stream, strong shadows are frequently cast

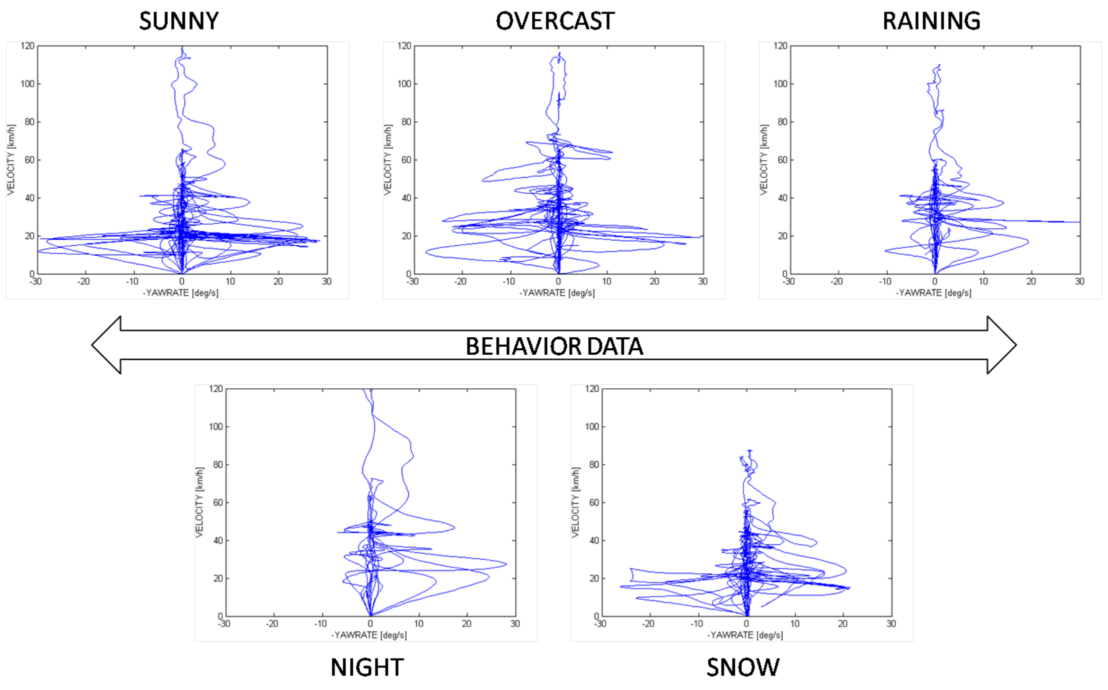


Figure 5.4: The video streams are accompanied by ground truth behavior data from the vehicle's CAN bus. Only velocity and yaw rate are shown here.

under sunny conditions, and rain often leads to bright reflections. Driving at night fundamentally changes the illumination of the entire scene, and snow covers the scene elements to a degree that impedes their clear segregation. Apart from these differences in their visual appearance, however, the streams are all very similar to each other: All were recorded while driving along the same route, although the overcast stream follows this route in the opposite direction (i.e., to the left, see Figure 5.4), and the night and raining streams begin somewhat later and end somewhat earlier than the others (thus missing some curves, see Figure 5.4).

The behavior data consists of various physical quantities about the vehicle, including its current velocity, yaw rate, steering angle, pedal status, and gear. As explained in Chapter 3, however, we only require the velocity and yaw rate data for our behavior prediction experiments. These two quantities are visualized in Figure 5.4, shown as plots over time. Note that each video frame gives rise to a single point, hence the formation of a trajectory for each stream.

| **Stream** | **Urban** | **Rural** | **Highway** | **Private** | **Invalid** |
|---|---|---|---|---|---|
| Overcast | 53 % | 21 % | 11 % | 14 % | 1 % |
| Sunny | 53 % | 6 % | 9 % | 26 % | 6 % |
| Night | 54 % | 9 % | 11 % | 0 % | 26 % |
| Rain | 78 % | 8 % | 14 % | 0 % | 1 % |
| Snow | 59 % | 7 % | 13 % | 17 % | 5 % |
| **Total** | **58 %** | **10 %** | **11 %** | **13 %** | **7 %** |

Table 5.2: Overview of the different scene types. Note that urban traffic scenes are clearly predominating, both in the entire dataset and in each of the streams.

| **Stream** | **First** | **Last** | **Frames** | **Duration** |
|---|---|---|---|---|
| Overcast | 1 | 9335 | 9335 | 16 min |
| Sunny | 1 | 9190 | 9190 | 15 min |
| Night | 150 | 4750 | 4601 | 8 min |
| Rain | 1 | 6315 | 6315 | 11 min |
| Snow | 340 | 8200 | 7861 | 13 min |
| **Total** | | | **37302** | **1h 2 min** |

Table 5.3: Clipping the invalid data at the beginning and ending of each stream leaves us with over 1 hour of natural driving. (Note that all streams have been downsampled to a framerate of 10 fps.)

A closer look at the plots shown in Figure 5.4 reveals that the vehicle has occasionally been driving at velocities that are much higher than appropriate for urban traffic. The reason lies in the heterogeneity of the route itself: While most of the driving actually takes place in urban traffic environments, it also includes a short period of driving on a highway, with rural roads leading to that highway, and some private property as well. Table 5.2 shows their relative frequencies, where differences mainly arise from the amount of traffic encountered.

Another important observation is that the original streams of the HRI dataset, as introduced by [36], contain short phases of invalid data: These phases are typically characterized by the vehicle waiting before the actual session begins, or after it has already finished. To ensure that each stream begins with the vehicle moving and ends with the vehicle stopping (or before driving into the dark garage), we have clipped all streams as specified in Table 5.3.

## 5.2 Segmentation Experiments

We now turn to our segmentation experiments, in which we evaluate the accuracy of our traffic scene representations as described in Chapter 4. By independently evaluating the representations prior to the behavior prediction system as a whole, we can assess the quality of our visual scene decomposition before using it further. This helps to distinguish whether potential limitations in the performance are inherent to our architecture, or caused by the perception already. Moreover, evaluating the traffic scene representations at the level of classical segmentation gives us quantitative results that we can directly compare to the state-of-the-art as reported in the literature. Note that our behavior prediction actually operates on the continuous response maps from which the segmentations are computed, but the latter are their direct visualization, as explained in Section 4.2.2.

In this context, we start with a quantitative evaluation of our representations on the CamVid dataset, using exactly the same object classes and data splits as in the literature. After that, we critically discuss the appropriateness of the object classes from the perspective of our behavior prediction experiments, and also propose a different split of the available data into a training and a test set. Finally, we apply the object classifiers thus trained on the CamVid dataset to each stream of the HRI dataset, and report on the results of this transfer.

### 5.2.1 Comparison to State-of-the-Art

Since the CamVid dataset consists of three daylight streams and one dusk stream, the available data is usually split for training and testing as follows (see Table 5.4). Two of the daylight streams (E5 and R0) are combined to form a day-specific training set, while the third daylight stream (VD) serves as a test set. Note that, in this case, training and testing can be performed not only on disjoint data but also on disjoint streams. As there is only one dusk stream (TP) in the dataset, however, the first half of this stream is commonly used for dusk-specific training, and testing is done on the second half of this stream. Although it implies that, in the dusk scenario, training and testing are conducted on the same stream, the blockwise splitting nevertheless ensures that the system is still evaluated on previously unseen data, given by disjoint parts along the driving route.

| Set | Sequence | First | Last | Frames |
|---|---|---|---|---|
| Day Train | E5, R0 | 390 / 930 | 8640 / 3930 | 305 |
| Day Test | VD | 0 | 5100 | 171 |
| Dusk Train | TP | 6690 | 8520 | 62 |
| Dusk Test | TP | 8550 | 10380 | 62 |

Table 5.4: Typical splits of the CamVid dataset into training and testing sets, separately for daylight and dusk conditions. (The remaining 101 frames, being part of the E5 sequence but labeled at a higher framerate, are usually ignored.)

| ID | Class | Labels |
|---|---|---|
| 1 | Building | Building |
| 2 | Tree | Tree, VegetationMisc |
| 3 | Sky | Sky |
| 4 | Car | Car, SUVPickupTruck, Truck_Bus |
| 5 | Sign-Symbol | SignSymbol, TrafficLight, Misc_Text |
| 6 | Road | Road, RoadShoulder, LaneMkgsDriv, LaneMkgsNonDriv |
| 7 | Pedestrian | Pedestrian |
| 8 | Fence | Fence, Wall |
| 9 | Column-Pole | Column_Pole |
| 10 | Sidewalk | Sidewalk, ParkingBlock |
| 11 | Bicyclist | Bicyclist |

Table 5.5: Standard object classes as in state-of-the-art segmentations [15, 99], and their corresponding CamVid labels (determined by visual inspection).

As for the object classes, it is common to consider a subset of the raw labels used in the segmentation ground truth, and to form 11 standard object classes by merging (see Table 5.5). For each of these object classes, a separate classifier is then learned from the training set (day or dusk), as explained in Section 4.2.1. The resulting object classifiers are then combined in a winner-take-all manner, where the predicted object label for a new image patch is given by the strongest-response classifier. By applying the combined classifiers to each of the test images, we obtain segmentations as shown in Figure 5.5.

To obtain a quantitative evaluation of the classifiers, we compute their global and average accuracies. The former is the share of correctly recognized patches

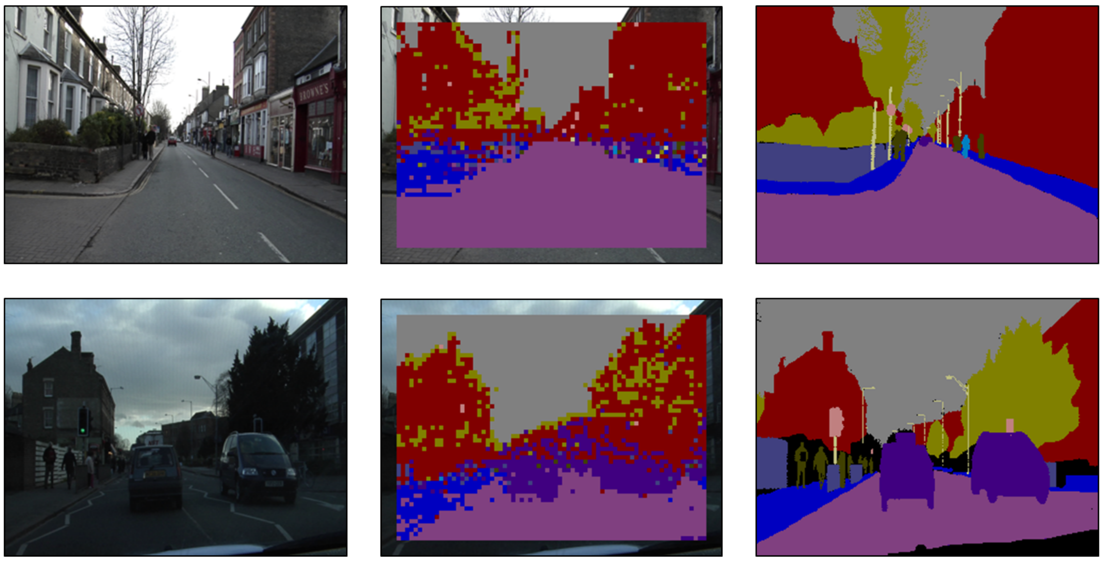


Figure 5.5: Example images (left) and their segmentations (center), using the WH implementation. The ground truth segmentations are also shown (right).

| | WH | | | CRF | | | Brostow | Sturgess |
|---|---|---|---|---|---|---|---|---|
| | **Day** | **Dusk** | **All** | **Day** | **Dusk** | **All** | **All** | **All** |
| **Global** | 69 | 62 | **66** | 68 | 67 | **68** | 69 | 84 |
| **Average** | 35 | 38 | **37** | 35 | 41 | **38** | 53 | 59 |
| **Baseline** | 9 | 9 | **9** | 9 | 9 | **9** | 9 | 9 |

Table 5.6: Accuracy of our WH and CRF segmentations, and state-of-the-art.

in the test set (i.e., $a_{\text{global}} = \frac{\#_{correct}}{\#_{all}}$), and intuitively correlates with the visual quality of the segmentations. The latter is obtained as the mean of the per-class accuracies (i.e., $a_{\text{average}} = \frac{1}{O}\sum_{o\in\{1,\dots,O\}}\frac{\#^{(o)}_{correct}}{\#^{(o)}_{all}}$), a harder measure giving equal weight to all classes regardless of their actual frequencies. It shows whether the classifiers are consistent, or better on some objects than on others. Our results (see Table 5.6) are similar with both implementations. In comparison to the state-of-the-art methods, our global accuracy is comparable to that of Brostow, although not reaching the level of Sturgess. Our average accuracy, however, is considerably lower than achieved by either of the two methods, indicating that misclassifications are mainly introduced by the infrequent classes.

| ID | Class | Labels |
|---|---|---|
| 1 | Sky | Sky |
| 2 | Tree | Tree, VegetationMisc |
| 3 | Building | Building |
| 4 | Sidewalk | Sidewalk, ParkingBlock |
| 5 | Road | Road, RoadShoulder |
| 6 | Car | Car, SUVPickupTruck, Truck_Bus |
| 7 | Pedestrian | Pedestrian |
| 8 | Bicyclist | Bicyclist |
| 9 | Marking | LaneMkgsDriv, LaneMkgsNonDriv |

Table 5.7: Our revised 9 object classes, and their corresponding CamVid labels. Note the grouping into "static" (1 – 5), "dynamic" (6 – 8), and "symbolic" (9).

### 5.2.2 Stand-alone Evaluation

As smaller object classes are a major source of confusion in our representations, it is worth examining which of them to consider at all. While larger object classes are generally justified in that they account for large parts of the traffic scenes, the inclusion of small objects should be governed by their potential relevance for driving behavior: For example, the "Column-Pole" class mostly consists of thin, elongated sign posts that are very difficult to detect and at the same time of rather limited behavioral relevance. The "Fence" class, in turn, is mostly comprised of narrow walls and fences in front of buildings, which are frequently confused with either the buildings or the adjacent sidewalks. Finally, the "Sign-Symbol" class contains many irrelevant signs like advertisements, intermixed with traffic signs and traffic lights. Although the latter are clearly relevant for driving behavior, traffic signs go beyond the requirement that the reason for the driver's behavior is visible in the corresponding traffic scenes, being valid long after leaving the field of view. Also, they cannot be treated as a single class because different signs have different semantics, which is also true for traffic lights and their various states. Since the HRI dataset itself does not support learning the proper reactions to traffic lights anyway, as the camera's field of view is too narrow to depict them while waiting, we leave symbolic information of these types for future extensions of this work, and concentrate on the object classes as given in Table 5.7.

|  | TRAIN |  |  | TEST |  |  |
|---|---|---|---|---|---|---|
|  | First | Last | Total | First | Last | Total |
| **E5** | 390 | 4515 | 74 | 4516 | 8640 | 130 |
| **R0** | 930 | 2430 | 51 | 2431 | 3930 | 50 |
| **VD** | 0 | 2550 | 86 | 2551 | 5100 | 85 |
| **TP** | 6690 | 8535 | 62 | 8536 | 10380 | 62 |

Table 5.8: Revised split of the CamVid dataset into a training and a test set.

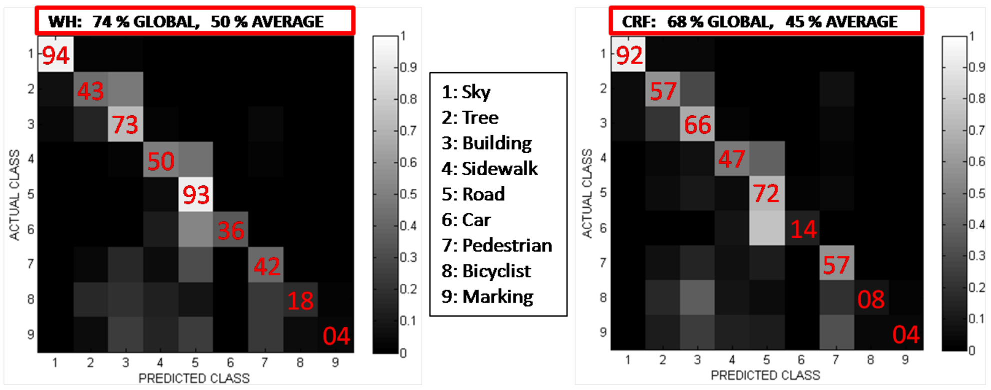


Figure 5.6: Confusion matrices of our trained object classifiers (WH and CRF).

Since we intend to apply the trained object classifiers to the HRI streams, which are characterized by heterogeneous weather and lighting conditions, a broad range of visual appearances should be covered in training already. We therefore combine all four streams of the CamVid dataset, mixing their lighting conditions and traffic scenes for robustness. To still obtain a quantitative evaluation of the resulting object classifiers, training is done on the first half of each stream, and the second half is used for testing (see Table 5.8). The results (see Figure 5.6) show that the average accuracy has greatly improved in both implementations (compare to Table 5.6), indicating that the classifiers perform well on most of the revised object classes. From the confusion matrices, the CRF implementation is slightly better on some of the larger static objects (e.g., trees vs. buildings), but the WH implementation is able to maintain a higher performance on the smaller dynamic objects as well (e.g., cars vs. road). We therefore hypothesize that it also performs better in the subsequent behavior prediction (see Section 5.3).

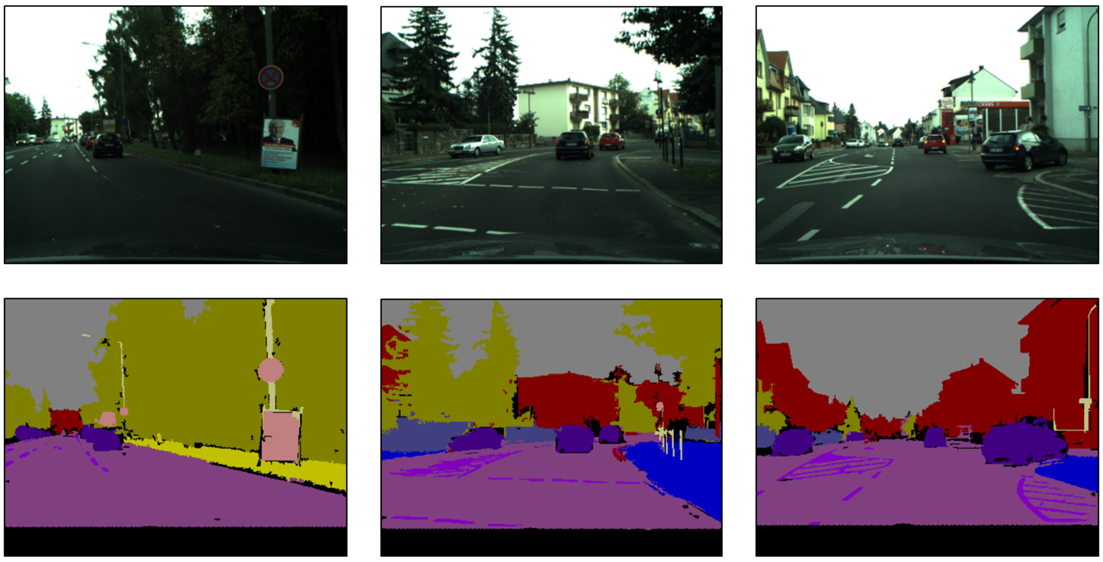


Figure 5.7: Example images taken from the HRI overcast stream (top row), and their manually annotated segmentation ground truth (bottom row).

### 5.2.3 Application to HRI Streams

We now apply the object classifiers, trained on the CamVid dataset as described in the previous section, to the video streams of the HRI dataset. To obtain a quantitative impression of the segmentation accuracy that is maintained after this transition, despite the lack of ground truth segmentation data in the HRI dataset, we have sparsely annotated the overcast stream as a representative example. The annotations were done analogously to those of the CamVid dataset, using the same annotation tool (called *InteractLabeler* [14]) but for every 100-th frame only, corresponding to an interval of 10 seconds (see Figure 5.7). The performance of the object classifiers was then measured on the 94 resulting frames, by computing their global and average accuracies as before. We obtain 62 % global accuracy and 44 % average accuracy for the WH implementation, and 58 % global accuracy and 41 % average accuracy for the CRF implementation. In other words, we only lose about 10 % global accuracy and 5 % average accuracy when making the transition from the CamVid dataset to the HRI dataset. However, we also hypothesize that weather conditions not included in the CamVid dataset, such as night and snow, lead to a greater loss and affect behavior prediction (see Section 5.3).

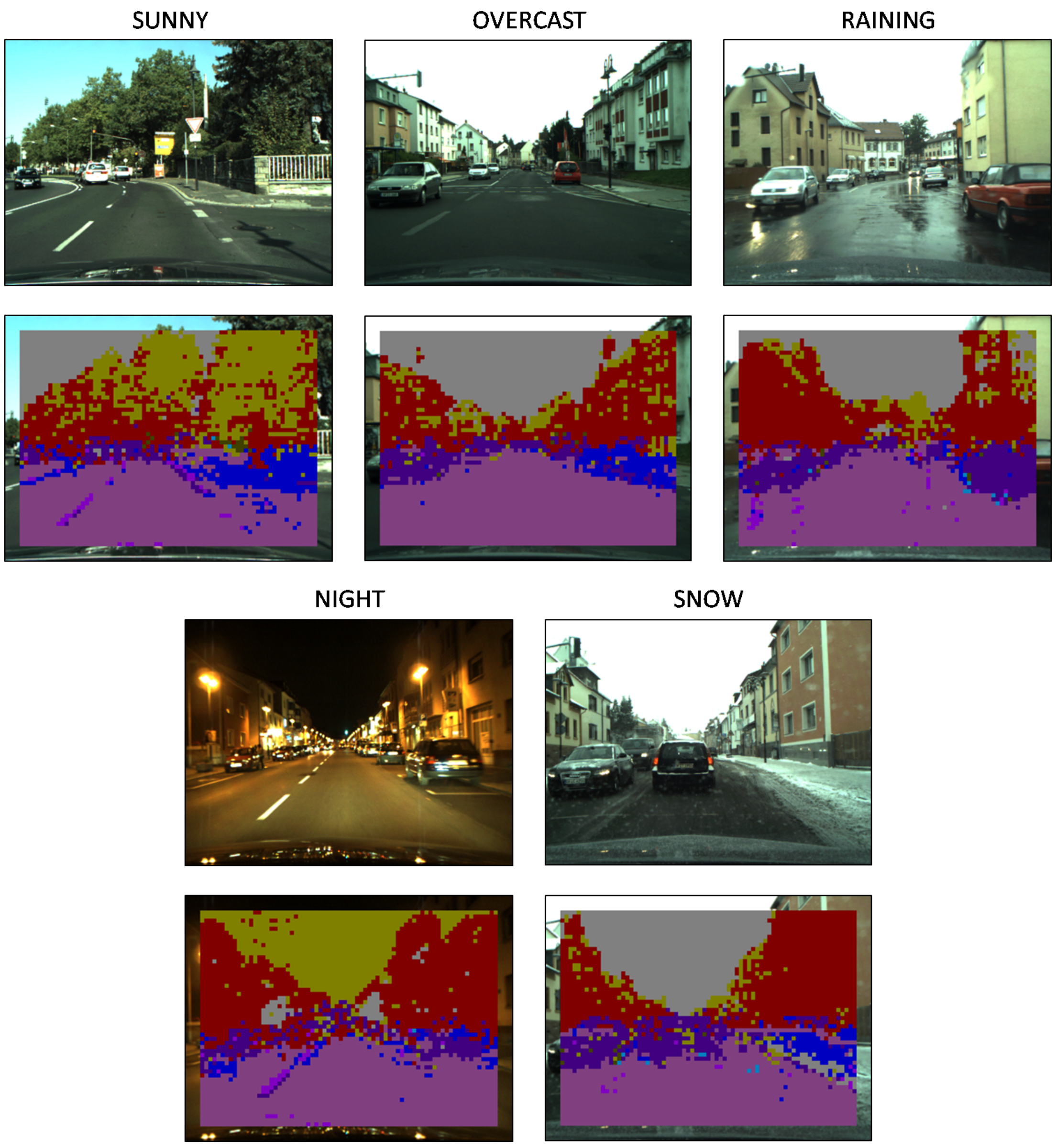


Figure 5.8: Qualitative results of our object classifiers applied to the HRI streams, using the WH implementation.

Example segmentations for all weather conditions are shown in Figure 5.8. The noise is an artifact of the winner-take-all scheme by which they are computed. As our behavior prediction utilizes the underlying response maps of the object classifiers, not the above visualizations, these artifacts have no further effect.

## 5.3 Behavior Prediction Experiments

In the following, we build on the results of the previous section by computing the semantic object-level representation for the HRI traffic scenes, as described in Chapter 4: These computations involve a kernel-weighted averaging of the continuous activation maps that are the immediate output of our CamVid-trained object classifiers, with regular but overlapping image grid cells. After serialization of the resulting feature maps into a single feature vector for each frame in the HRI dataset, we can then use these feature vectors to train our behavior classifiers (see Chapter 3). For comparison, we also compute the continuous response maps of the oriented edge filters that form the basis of the state-of-the-art method for behavior prediction [84], resulting in separate WH-, CRF-, and GIST-based behavior classifiers that we quantitatively evaluate and compare to each other.

Our evaluation of these behavior classifiers begins at the level of their raw single-frame responses, representing independent predictions of the appropriate driving behavior for each point in time. To obtain a physically plausible and robust performance, we then establish temporal coherence in successive frames by means of various signal filters, applied to the single-frame responses. Finally, we address the incorporation of different confidence measures to enable the system to refrain from making a behavior prediction in ambiguous traffic situations.

### 5.3.1 Behavior Class Thresholds

Prior to the actual behavior prediction experiments, however, it is fundamental to ensure that the behavior classes under consideration reflect the typical range of human driving behavior. While, technically speaking, any choice of thresholds in the continuous-valued domains of velocity and yaw rate yields a corresponding set of behavior classes, as explained in Chapter 3, not all can be considered semantically meaningful such that they are useful for practical applications in driver assistance or autonomous navigation. We therefore begin with a discussion of how to adequately set these thresholds, given the actual behavior data as recorded by the CAN bus of the moving vehicle. The resulting behavior classes of velocity and yaw rate are then used throughout the remainder of this chapter, forming the basis for our quantitative evaluation of their prediction accuracy.

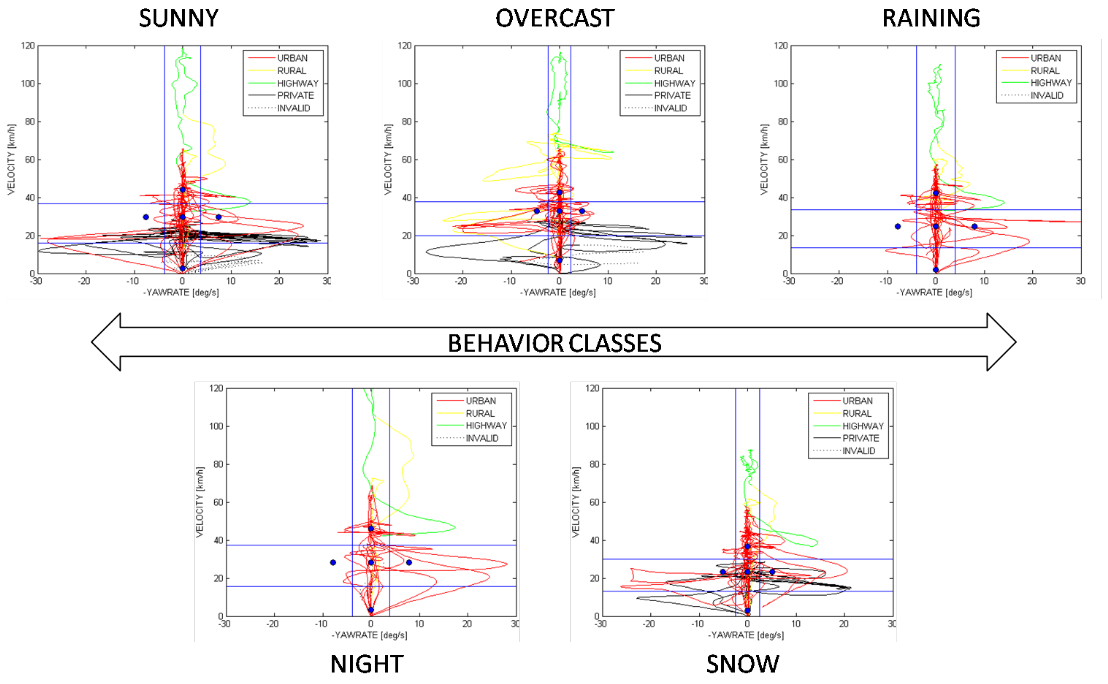


Figure 5.9: The behavior class thresholds (blue lines) as used in our experiments, resulting in three velocity classes (horizontal) and three yaw rate classes (vertical).

Since our goal is to have the behavior classes match the actual behavior of a human driver, it is reasonable to consider real CAN data for the purpose of setting their underlying thresholds. Figure 5.9 shows plots of the velocity and yaw rate for each of the different streams or weather conditions in the HRI dataset, with further distinction of the overall scene types as specified earlier (see Table 5.2). The horizontal blue lines indicate the velocity thresholds that are used for our subsequent experiments, and the vertical blue lines show the yaw rate thresholds. Their values are given in Table 5.9, which could either be determined manually as in our original approach [43], or automatically fitted to the data by running a $k$-means clustering algorithm. More specifically, we only consider the inner-city urban part of the velocity and yaw rate data, respectively, and apply the clustering to each of these two subsets separately, with $k \in \mathbb{N}$ defining the number of desired behavior classes. To obtain symmetric classes despite the obvious imbalance of left and right curves, the yaw rate subset is also included with all signs flipped.

| | VEL | | | | | YAW | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $C_{slow}$ | **t** | $C_{medium}$ | **t** | $C_{fast}$ | $C_{left}$ | **t** | $C_{straight}$ | **t** | $C_{right}$ |
| **Sunny** | 3 | **16** | 30 | **37** | 44 | 8 | **4** | 0 | **-4** | -7 |
| **Overcast** | 7 | **20** | 33 | **38** | 43 | 5 | **2** | 0 | **-2** | -5 |
| **Raining** | 2 | **13** | 25 | **33** | 42 | 8 | **4** | 0 | **-4** | -8 |
| **Snow** | 3 | **13** | 23 | **30** | 37 | 5 | **3** | 0 | **-3** | -5 |
| **Night** | 3 | **16** | 28 | **37** | 46 | 8 | **4** | 0 | **-4** | -8 |

Table 5.9: Rounded values of the behavior class thresholds t (see blue lines in previous figure) and the behavior class centers $C_*$ (blue circles in previous figure).

| | VEL | | | YAW | | |
|---|---|---|---|---|---|---|
| | $C_{slow}$ | $C_{medium}$ | $C_{fast}$ | $C_{left}$ | $C_{straight}$ | $C_{right}$ |
| **Sunny** | 24 % | 41 % | 35 % | 7 % | 80 % | 13 % |
| **Overcast** | 26 % | 28 % | 46 % | 19 % | 67 % | 14 % |
| **Raining** | 24 % | 21 % | 55 % | 5 % | 85 % | 10 % |
| **Snow** | 29 % | 47 % | 24 % | 15 % | 64 % | 21 % |
| **Night** | 22 % | 21 % | 57 % | 6 % | 77 % | 17 % |

Table 5.10: Distributions of the resulting behavior classes, per stream (rounded). As velocity and yaw rate are predicted in parallel, they sum up to 100 % each.

Notice the importance of only fitting the behavior classes to the inner-city urban parts of each stream, rather than to the entire stream data. As each stream consists of urban, rural, and highway parts, characterized by their own ranges of velocities and yaw rates, respectively, the latter would tend to yield a single class for each of these scene types, which is more like classical image categorization and has been done before [49]. By fitting the behavior classes on the inner-city urban parts, in contrast, we can ensure that they indeed correspond to different actions of the human driver in an otherwise similar urban traffic environment, which is consistent with the general objective of our thesis work (see Chapter 1). After the fitting, of course, the resulting behavior class thresholds can then be applied to the entire stream data, giving rise to the behavior class distributions as shown in Table 5.10. To illustrate their adequacy for practical applications, qualitative examples of our behavior classes are depicted in Figure 5.10 and in Figure 5.11, for the velocity domain and for the yaw rate domain, respectively.

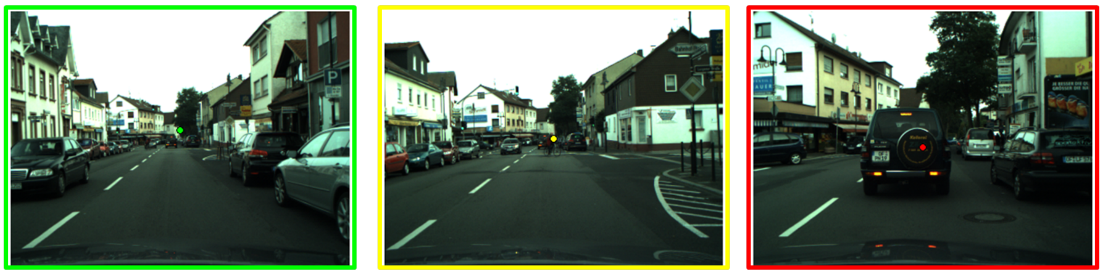


Figure 5.10: Qualitative example situations to illustrate our velocity classes. Their semantic labels are “fast” (green), “medium” (yellow), and “slow” (red).

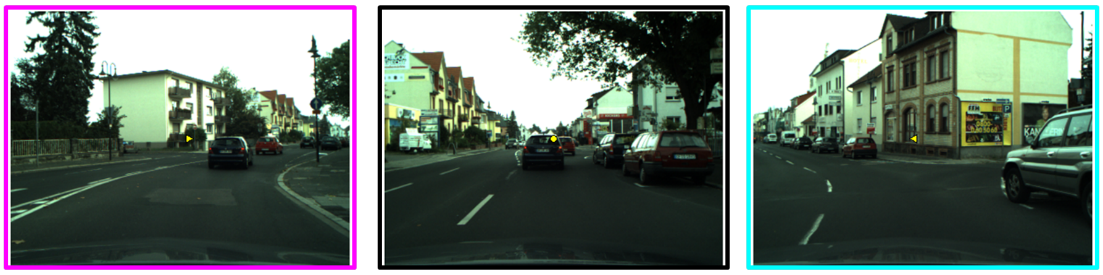


Figure 5.11: Qualitative example situations to illustrate our yaw rate classes. Their semantic labels are “right” (magenta), “straight” (black), and “left” (cyan).

### 5.3.2 Instantaneous Prediction

The behavior classifiers are trained on the HRI streams like the object classifiers on the CamVid dataset, with the following distinction: As we are interested in the effect of different weather conditions, encapsulated in the different HRI streams, training and testing is done separately for each stream. This situation resembles the CamVid dusk condition, where the TP stream was split in two blocks for disjoint training and test sets. The behavior classifiers need more training images than the object classifiers, however, because a single training image provides a high number of example patches, but only one velocity and yaw rate example. To compensate, we split the HRI streams in ten blocks and perform a blockwise cross-validation. As each block consists of 1 - 2 minutes of continuous driving, adjacent blocks contain different traffic situations despite their overall similarity. In particular, we always test on previously unseen parts of the route this way.

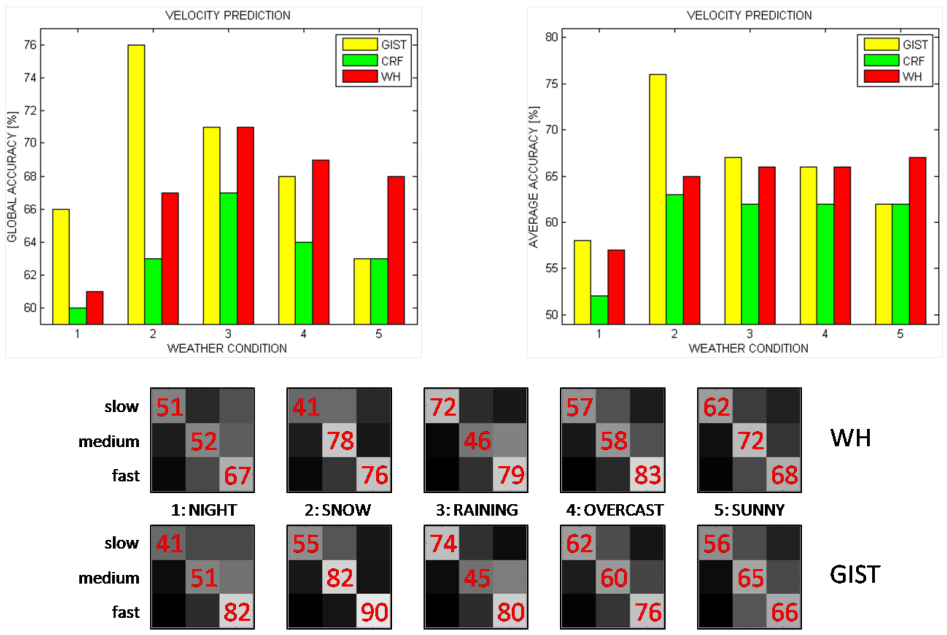


Figure 5.12: Global (top left) and average accuracies (top right) for WH-, CRF-, and GIST-based velocity prediction. Confusion matrices are shown below.

By concatenating the single-frame predictions of the individual test blocks, we obtain a full sequence of velocity and yaw rate predictions for each stream. A frame-wise comparison to the actual velocity and yaw rate data enables us to compute their confusion matrices, global and average accuracies, separated by weather conditions and traffic scene representations. Looking at the graphs for velocity prediction in Figure 5.12, we see that our CRF and WH implementations are rather similarly affected by the different weather conditions, but the latter is consistently superior. Limitations of this representation arise in night conditions where the accuracies are lowest, presumably because the visual appearance of traffic scenes at night is changed the most. The GIST representation, in contrast, is affected differently by the weather conditions: Its accuracies are highest for snow scenes, but they quickly degrade in more standard conditions. Particularly, sunny weather causes the GIST performance to drop to CRF level, while our WH implementation is able to maintain a considerably higher accuracy. We attribute this weakness of the GIST implementation to its oriented edge filters, which are apparently sensitive to the strong shadows that are typical of sunny conditions.

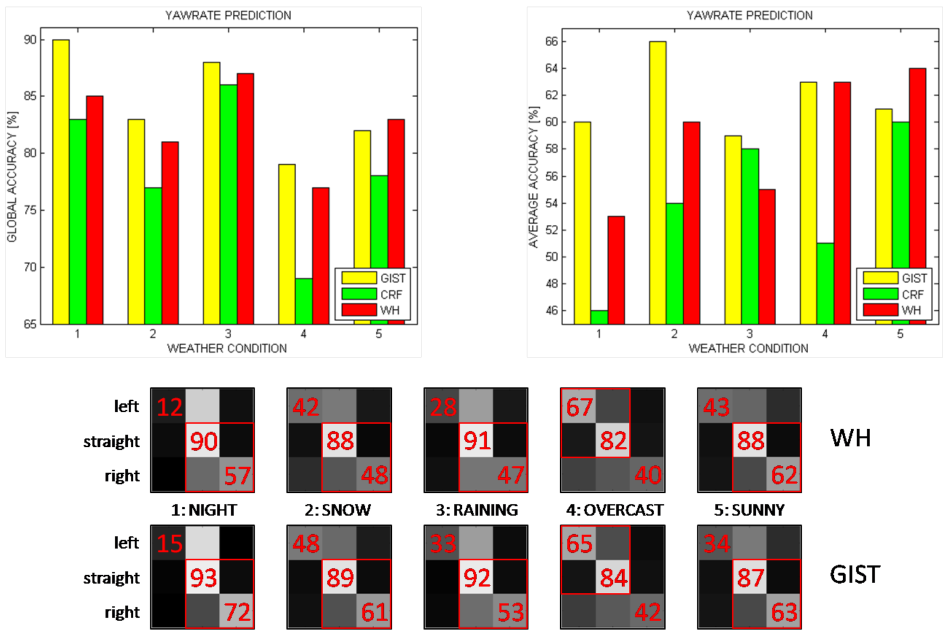


Figure 5.13: Global (top left) and average accuracies (top right) for WH-, CRF-, and GIST-based yaw rate prediction. Note the asymmetric confusion matrices.

As for the yaw rate prediction results (see Figure 5.13), the graphs show that all representations are similarly affected by the different weather conditions. With the only exception being the average accuracy of the CRF implementation in raining conditions, we again find that our WH implementation is performing consistently better. Also, the GIST representation does not reach the accuracy of our WH implementation in sunny conditions, but the difference is much less pronounced here. Overall, our WH implementation achieves a similar performance as the GIST representation, with the difference becoming more prominent in the non-standard conditions like night and snow. In contrast to the results for velocity prediction, however, the average accuracies are considerably lower than the global accuracies, which indicates that not all yaw rate classes were learned equally well. A closer look at the confusion matrices shows that the predictions are generally accurate for steering in the direction in which the route was followed (see rectangles), but not for the opposite direction. This reveals a limitation of the HRI dataset whose streams do not provide enough training examples, hence we expect a more balanced dataset to yield higher average accuracies as well.

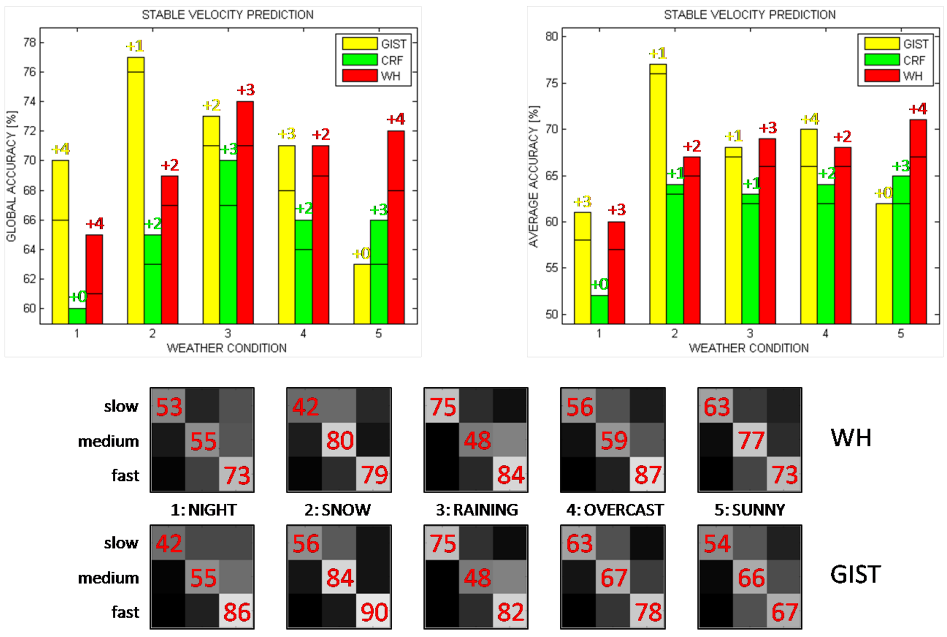


Figure 5.14: Global (top left) and average accuracies (top right) for WH-, CRF-, and GIST-based velocity prediction, with temporal stabilization by median filters.

### 5.3.3 Temporal Stabilization

While the single-frame responses are the basic output of our behavior classifiers, we have argued in Chapter 3 to stabilize them over time for noise reduction and to better match the driver's actions. We therefore apply a median filter to the discrete single-frame predictions, and also to the continuous responses from which they are computed. As filtering introduces a temporal lag as well, practical considerations impose a limit on the viable filter size. Given that human drivers are generally subject to a reaction time of about 1 second, we thus set the size of our filters correspondingly. Looking at the accuracy graphs for velocity prediction (see Figure 5.14), we observe an increase for all methods in nearly all conditions. The overall characteristics of the curves is nevertheless preserved, indicating that the effect of the filters is mainly restricted to noise reduction. Although the improvement may appear rather gentle in terms of numbers, it considerably contributes to the visual plausibility of the resulting predictions, as frequent oscillations are effectively suppressed. Note that, unlike our WH imple-

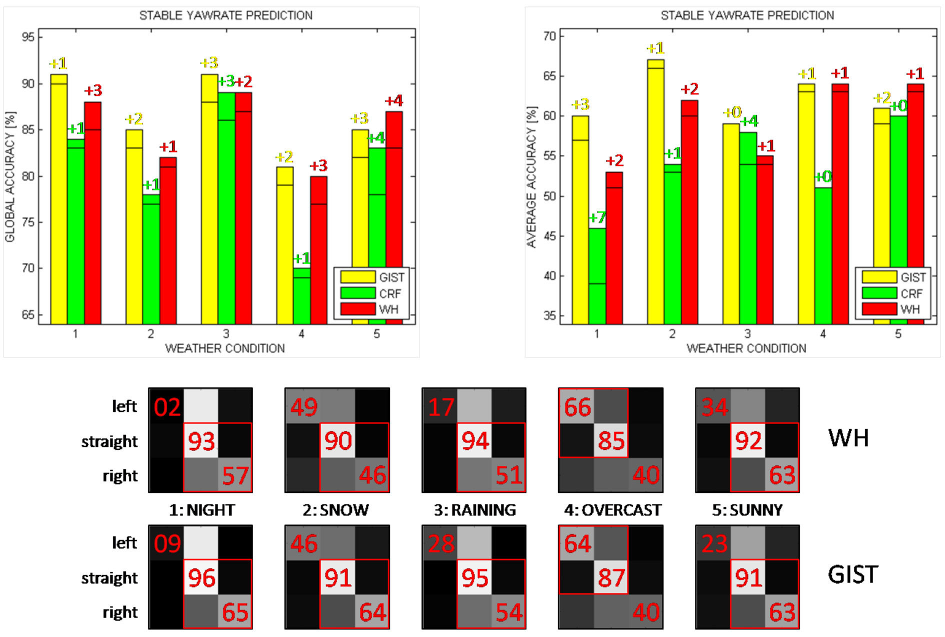
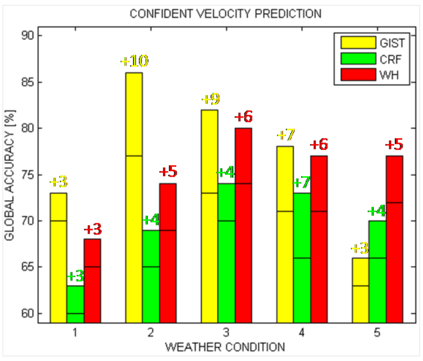
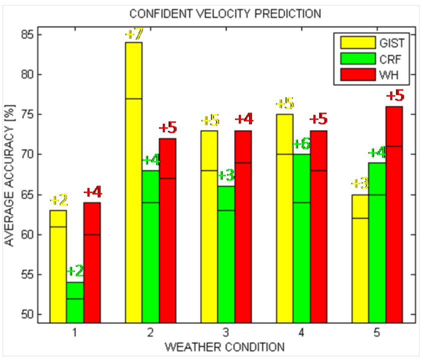


Figure 5.15: Global (top left) and average accuracies (top right) for WH-, CRF-, and GIST-based yaw rate prediction, using the same temporal stabilization.

mentation, the GIST representation is unable to benefit from the stabilization in sunny weather at all, which emphasizes the advantage of our method in this condition and also supports our view that the GIST representation suffers from a more fundamental limitation here that cannot be explained away by noise alone.

From the accuracy graphs of the stabilized yaw rate prediction (see Figure 5.15), a similar effect as in the stabilized velocity prediction can be observed: Without changing the overall characteristics of the curves themselves, the filtering improves the (global) accuracy of each method in all weather conditions. Its effect on the average accuracies, however, may appear surprising at first: Here, it leads to inconsistent results in that some values are slightly improved while others are slightly decreased, and the CRF implementation is even negatively affected. This finding indicates that the limits of the yaw rate prediction are not primarily imposed by the presence of noise but are more deeply rooted, which is consistent with our earlier observation that steering could not be learned equally well in both directions. This reason is also visible in the confusion matrices, where the dominant direction is slightly improved at the expense of the other direction.





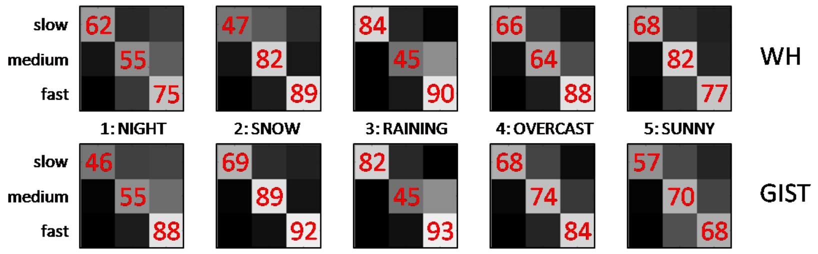


Figure 5.16: Global (top left) and average accuracies (top right) for WH-, CRF-, and GIST-based velocity prediction, with temporal stabilization and confidences.

### 5.3.4 Confidence Measures

Finally, we investigate into the effect of allowing the system to refrain from making a prediction if the confidence is not high enough. As explained in Chapter 3, we measure the confidence of a prediction as the ratio between the amplitudes of the winning and the second-winning responses of the behavior classifiers, for each frame. By applying a threshold to the resulting confidence values, the system is able to detect unreliable predictions by their sub-threshold confidences, and suppress their output. While the confidence mechanism generally improves the prediction accuracy, as unreliable predictions are no longer counted against the system, it also implies a certain share of effective "downtime" that impedes its usefulness if too long. To meet the anticipated requirements of practical applications, we therefore set the confidence threshold such that 10 % rejections are allowed here (autonomous navigation may require less whereas driver assistance may allow for more).

As for the velocity prediction results (see Figure 5.16), allowing 10 % rejec-

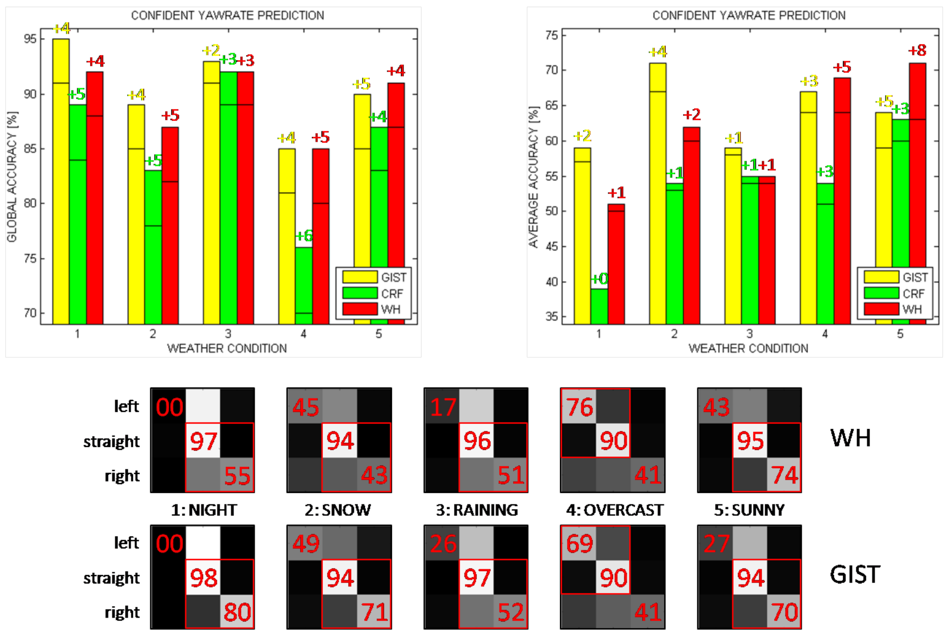


Figure 5.17: Global (top left) and average accuracies (top right) for WH-, CRF-, and GIST-based yaw rate prediction, with temporal stabilization and confidences.

tion rate considerably improves all accuracy values beyond what was possible by temporal stabilization, both for the global and the average accuracies. This observation indicates that the confidence mechanism is capable of dealing with limitations of the different methods that are not just the result of noise reduction or outlier removal. Also, we see that the overall characteristics of the graphs, again for both types of accuracies, is still preserved by the confidence mechanism, only the accuracies themselves are generally higher. In particular, our WH implementation still performs better than the CRF implementation for all weather conditions, and the GIST representation still excels in snow conditions while being strongly outperformed in sunny conditions (with similar results for overcast and raining). As the accuracies of both our WH implementation and the GIST representation are very similar for overcast and raining conditions, apart from their large difference in sunny conditions, we conclude that our WH implementation is more robust as far as the standard weather conditions are concerned.

The results for yaw rate prediction (see Figure 5.17) show a similar effect of the confidence mechanism on the prediction accuracies in that a considerable

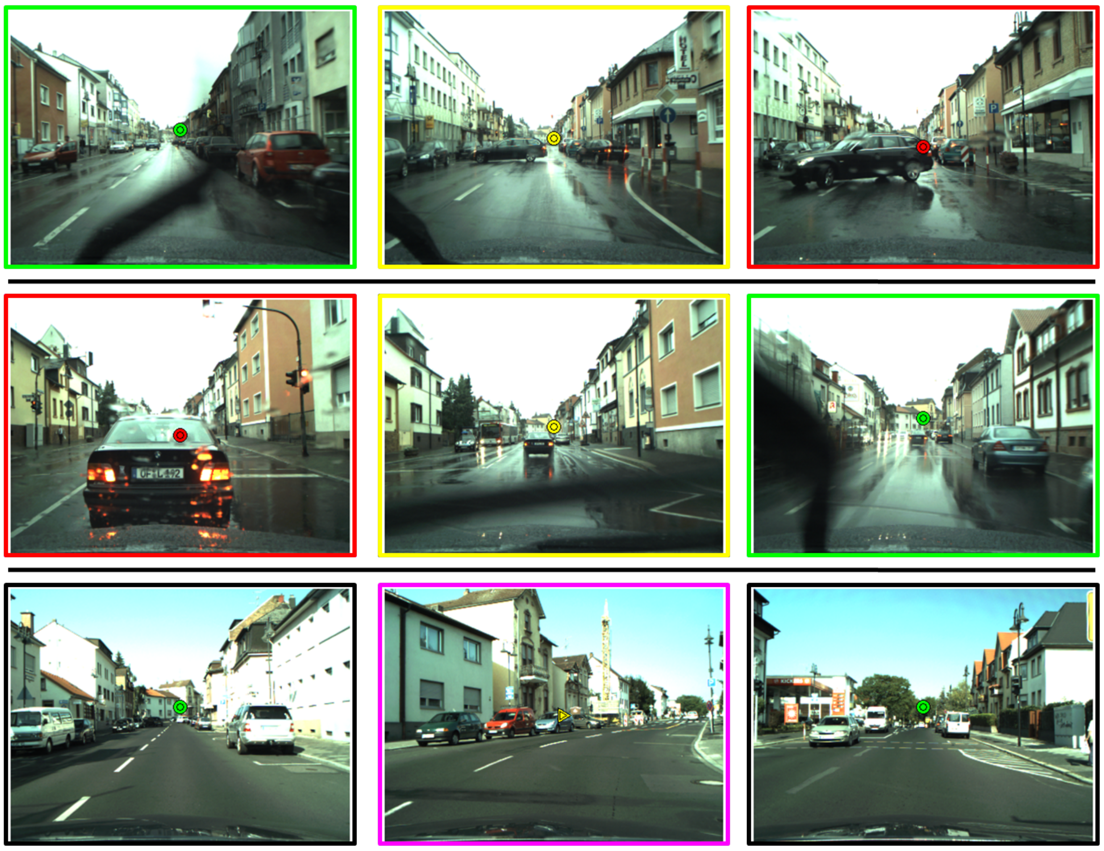


Figure 5.18: Qualitative example situations of our behavior prediction system in action. Top: gradually slowing down, center: speeding up again, bottom: driving a curve. All predictions (small dots) match the ground truth here (large dots).

improvement over the stabilized results of the previous section are achieved. Like in the previous section, however, these improvements are mainly limited to the global accuracies, without considerable benefits for the average accuracies (except for our WH implementation in sunny and overcast weather conditions). This result is to be expected, as neither the temporal stabilization nor the confidence-based suppression of unreliable predictions will be able to compensate for the general lack of training examples in the non-dominant steering direction of the HRI streams. To convey an intuitive impression of the final behavior prediction system in action, Figure 5.18 shows qualitative examples for different weather conditions, obtained when using the WH implementation.

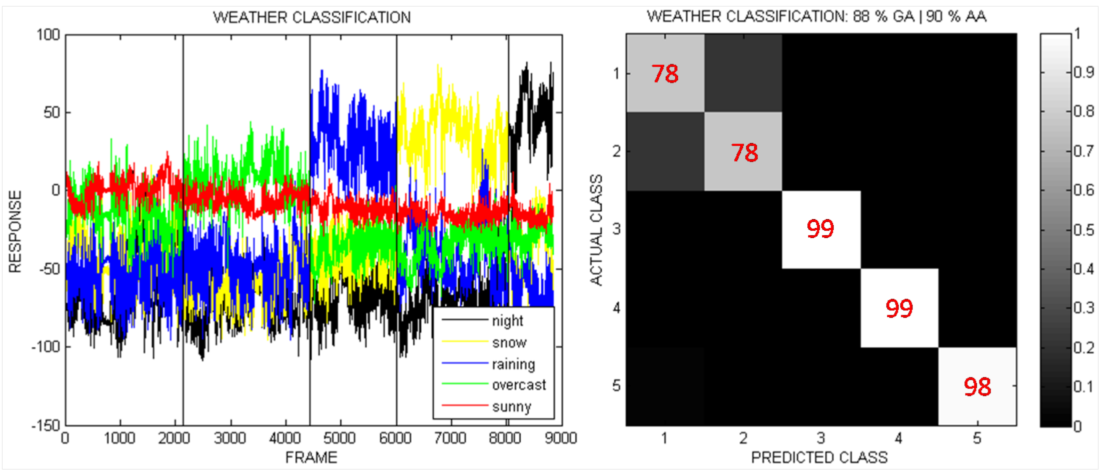


Figure 5.19: Raw weather recognition results (left: classifier responses, right: confusion matrix).

### 5.3.5 Weather Recognition

While we have considered the different weather conditions of the HRI dataset separately from each other, in our previous behavior prediction experiments, real applications cannot make such a rigid distinction because the weather condition might change during the ride. We therefore investigate whether it is possible to automatically determine which of the five weather conditions is actually present, at any given point in time. Being a classical scene categorization task such as the distinction between different scene types like "urban", "rural", and "highway" (see [49], for example), as global properties of the images are to be distinguished, we employ a standard GIST approach with the extension by Gaussian kernels and overlapping image grid cells as in the filter-based traffic scene representation [84], and represent all images of the HRI dataset by the corresponding feature vectors. After splitting each stream into a first and a second half for training and testing, respectively, we use the set of feature vectors corresponding to the training data to learn image classifiers for each stream in a one-versus-all manner, sub-sampling the feature vectors for speed such that only 1000 are used for each stream. The classifiers are then applied to the set of feature vectors corresponding to the test data, and the resulting classifier response curves are shown in Figure 5.19,

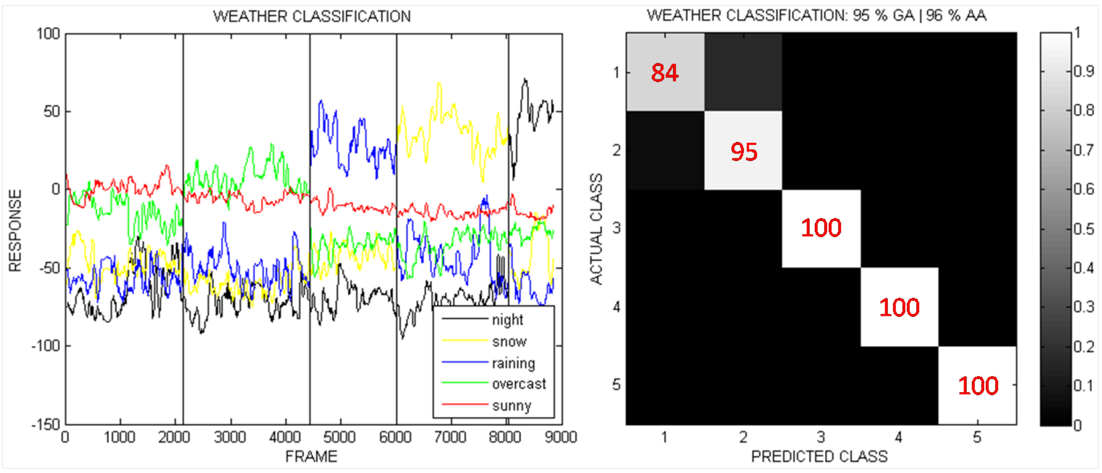


Figure 5.20: Stabilized weather recognition results (left: classifier responses, right: confusion matrix).

left. Note that the classifier with the maximum response value is considered the winner, for each frame, and consequently determines the predicted weather condition at that timestep. By computing the confusion matrix and the accuracies in the usual way as before, we obtain the results depicted in Figure 5.19, right. Apparently, the weather condition is recognized with a high accuracy, having almost no misclassifications for “night”, “snow”, and “raining”, and moderate misclassifications between “overcast” and “sunny” conditions.

As the weather condition can only change gradually, at a relatively slow frequency, the same considerations about temporal coherence between successive frames are applicable, and we therefore stabilize the single-frame predictions of the weather condition as well. While predictions of the appropriate velocity and yaw rate should be made as fast as possible, to be useful in real driver assistance or autonomous navigation applications, this is not as critical in the case of weather recognition, and we can therefore choose a slightly larger filter size despite the slightly longer temporal lag that comes with it. Specifically, we tolerate a 5-second lag instead of the previous 1-second lag, which leads to a considerable improvement of the weather recognition (see Figure 5.20). This information could thus be used to arbitrate between the parallel predictors, with little additional effort and largely transparent for the human driver.

# Chapter 6

# Conclusion

In this work, we have dealt with the problem of how to achieve that a technical system can develop a general scene understanding of its immediate environment, such that it is capable of recognizing, interpreting, and reacting to the current traffic situation in a basic manner, like a human driver. The purpose of such a technical system is to generate independent predictions about the appropriate driving behavior in a given traffic scene, separately from the actual behavior of the human driver, which can then be compared to each other to detect and warn the driver about potential mismatches, if not directly intervening temporarily.

To be able to handle the underlying learning problem in a technical sense, given the continuous video data stream of a car-mounted front stereo camera on the one hand, as well as the continuous behavior data stream of the driver from the CAN bus of the moving vehicle on the other hand, from which typical correlations should be extracted, we have developed a system architecture for driving behavior prediction that is an extension of our original approach [43], as described in Chapter 3. The basic idea is to subdivide the continuous-valued behavior space spanned by the CAN data of interest into discrete, semantically meaningful behavior classes, such that the resulting image categories in the visual data stream can be used to train a number of dedicated image classifiers in a supervised way. The prediction of the appropriate driving behavior for new images depicting traffic scenes is then obtained by the application of all image classifiers in parallel, and determining the corresponding behavior classes in a winner-take-all manner. In particular, the behavior classes are automatically fitted to example CAN data, such that a useful discretization of the behavior space in the context of the intended application is achieved without having to manually set the parameters. In addition, a property that distinguishes our system architecture for driving behavior prediction from other approaches [84] is the idea of approximating the continuous CAN data as required by the application, rather than considering binary quantities such as the pedal status only.

Of central importance is the way the traffic situations are visually represented for subsequent processing in our behavior prediction architecture. This is because the behavior data from the CAN bus actually spans a relatively low-dimensional space of interest that can be handled well by the aforementioned sub-division into behavior classes, whereas the visual domain is of much higher complexity. In particular, our focus on urban traffic situations rather than highway or off-road environments strongly contributes to the difficulty of the learning problem, since these are generally more densely populated by other traffic participants, composed of a wide variety of different objects or scene elements, and exhibit higher fluctuations of different traffic situations with corresponding implications for the behavioral side, which is the reason why we have placed emphasis on the traffic scene representations in Chapter 4. Our approach consists in a thorough decomposition of each traffic scene into its constituent semantic entities, which is achieved by training an additional range of object classifiers. The underlying learning process again relies on training data, which is composed of manually annotated example images of traffic scenes that indicate the different objects by means of labels, and is therefore supervised as well. Unlike the behavior classifiers, which operate on entire images, these object classifiers operate on patches that are densely sampled from an image. While our original CRF implementation [43] was built upon the LM filter bank, with subsequent integration of the classifier responses in a state-of-the-art Conditional Random Field and thus operating at the level of segmentations, our current WH implementation is an extension that operates on the continuous-valued response maps of the object classifiers, which are now trained on efficient Walsh-Hadamard features. While the underlying techniques of these traffic scene representations are well-known state-of-the-art methods, and not a part of our own contribution, investigating their usefulness for driving behavior prediction has not been done before, and our qualitative comparison to the extended GIST representation as used by the most related approach to driving behavior prediction [84] has also not been conducted so far.

At the end of this thesis work, we have put the proposed techniques and frameworks to the test, evaluating them quantitatively under realistic conditions. In particular, we have conducted a detailed analysis of the accuracy with which the appropriate driving behavior can be predicted by our proposed system architecture from Chapter 3, directly comparing the predicted behavior classes to the correct ones as given by the ground truth CAN data of a human driver. In this context, we have also conducted a comparative evaluation of our two different implementations for the semantic object-level representation, namely, the CRF implementation and the WH implementation. Our experiments show that the latter consistently outperforms the former, which we attribute to the gray-world normalization of the individual color channels that arguably contributes to the robustness against illumination and color fluctuations, in conjunction with the systematic sampling of overlapping image patches at each location in the field of view. Moreover, the advantages of the CRF implementation in that a visually plausible segmentation of the traffic scene can be obtained did not outweigh the advantage of the WH implementation in that uncertainty is being preserved in the feature vectors, and our direct comparison of the two also suggests that the Walsh-Hadamard filter bank appears to be better than the LM filter bank, in this context. Importantly, we have always compared the prediction accuracies of our own implementations for the semantic object-level representation to the best traffic scene representation for driving behavior prediction currently used, which relies on raw image filter responses and is an extended version of the GIST descriptor [84]. Systematically evaluating the prediction accuracies of these methods in a variety of different weather conditions that are often encountered in real-world environments, we found our WH implementation to be more robust than the GIST-based representation, when considering velocity predictions, as sunny weather conditions led to a significant decrease in prediction accuracy for the latter while ours was able to maintain a high accuracy. For yaw rate prediction, both performed approximately equally well for standard weather conditions, and only when considering more extreme conditions such as night and snow, the WH implementation did not keep up with the GIST representation, which we attribute to the fact that our training data [14] did not contain any examples of such conditions. Importantly, we have also shown that temporal stabilization and the introduction of a confidence-based reject option considerably improves the prediction accuracy and are therefore useful for practical applications. Neither these practical considerations nor the quantitative comparison of different traffic scene representations for driving behavior prediction have been conducted before, and our systematic evaluation of different weather conditions is also unparalleled in the respective literature. It is our hope that future work on this topic be evaluated in a similar way, and that our performance analysis on public datasets can contribute to the formation of an evaluation standard that goes beyond the current practice of testing on non-accessible streams, to make the results comparable and thus advance the progress on this important problem.

Future work should address the explicit modeling of small but behaviorally relevant symbolic objects, beyond lane markings as considered in our work. Important examples are traffic signs and traffic lights, which need to be further distinguished by their respective types or states, as these have different semantic meanings. Specialized features may also be needed for their reliable detection, unlike standard filter banks. The incorporation of temporal dependencies is another important extension that should be investigated further. Such dependencies arise in the context of traffic signs, for example, which are valid long after leaving the field of view, but are also necessary to account for the inertia of the ego-vehicle when making predictions about the appropriate driving behavior, as the car always requires some time to change its current physical state. By further analyzing the behavior prediction performance in dependence of prototypical traffic situations, such as crossings, junctions, stop lights, construction sites, or traffic jams among others, more insight could be gained into which of these situations may require more specific techniques to be handled correctly. For example, our proposed approach explicitly focuses on learning typical reactions to frequently encountered situations, while construction sites, for example, are examples of non-standard situations that nevertheless must be dealt with. With these possibilities for future research in mind, we are convinced that our vision-based approach to driving behavior prediction by direct correlation learning will prove to be useful for serving as a fundamental building block.

# References

[1] Adams, R. P., and MacKay, D. J. Bayesian online changepoint detection, 2007. Technical report, University of Cambridge, UK. 23

[2] Agamennoni, G., Nieto, J. I., and Nebot, E. M. A bayesian approach for driving behavior inference. In *Proc. IV* (2011). 21

[3] Bacha, A., et al. Odin: Team VictorTango's entry in the DARPA urban challenge. *Journal of Field Robotics* (2008). 14

[4] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., and Szeliski, R. A database and evaluation methodology for optical flow. *International Journal of Computer Vision* (2011). 74

[5] Beal, M. Variational algorithms for approximate bayesian inference, 2003. PhD thesis, University College London, UK. 21

[6] Bohren, J., et al. Little Ben: The Ben Franklin racing team's entry in the 2007 DARPA urban challenge. *Journal of Field Robotics* (2008). 14

[7] Bosch, A., Zisserman, A., and Munoz, X. Scene classification via pLSA. In *Proc. ECCV* (2006). 31

[8] Boykov, Y., and Jolly, M. P. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *Computer Vision* (2001). 26, 27

[9] BRAESS, H. H., AND REICHART, G. Prometheus: Vision des 'intelligenten Automobils' auf 'intelligenter Straße'? Versuch einer kritischen Würdigung – Teil 1. In *ATZ Automobiltechnische Zeitschrift 97, Band 4* (1995). 11

[10] BRAESS, H. H., AND REICHART, G. Prometheus: Vision des 'intelligenten Automobils' auf 'intelligenter Straße'? Versuch einer kritischen Würdigung – Teil 2. In *ATZ Automobiltechnische Zeitschrift 97, Band 6* (1995). 11

[11] BREIMAN, L. Random forests. *Machine Learning* (2001). 24

[12] BROGGI, A., BERTOZZI, M., AND FASCIOLI, A. The 2000 km test of the ARGO vision-based autonomous vehicle. In *Proc. IEEE Intelligent Systems* (1999). 11

[13] BROGGI, A., BERTOZZI, M., FASCIOLI, A., AND CONTE, G. *Automatic vehicle guidance: The experience of the ARGO autonomous vehicle*. World Scientific Co., 1999. 10, 11

[14] BROSTOW, G. J., FAUQUEUR, J., AND CIPOLLA, R. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters* (2009). 61, 71, 73, 74, 75, 86, 104

[15] BROSTOW, G. J., SHOTTON, J., FAUQUEUR, J., AND CIPOLLA, R. Segmentation and recognition using structure from motion point clouds. In *Proc. ECCV* (2008). 28, 73, 82

[16] BUEHLER, M., IAGNEMMA, K., AND SINGH, S. *The 2005 DARPA Grand Challenge*. Springer, 2007. 10, 12

[17] BUEHLER, M., IAGNEMMA, K., AND SINGH, S. *The DARPA Urban Challenge*. Springer, 2010. 10, 14

[18] CAO, L., AND FEI-FEI, L. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *Proc. ICCV* (2007). 32

[19] CASELLA, G., AND ROBERT, C. P. Rao-blackwellization of sampling schemes. *Biometrika* (1996). 23

[20] DARPA. Darpa Grand Challenge: Autonomous ground vehicles, 2004. http://archive.darpa.mil/grandchallenge04/. 12

[21] Dean, T., and Kanazawa, K. A model for reasoning about persistence and causation. *Computational Intelligence* (1989). 19

[22] Demcenko, A., Tamosiunaite, M., Vidugiriene, A., and Jakevicius, L. Lane marker parameters correlative to vehicle's steering signal. In *Proc. International Conference on Signal Processing, Robotics and Automation* (2009). 22

[23] Demcenko, A., Tamosiunaite, M., Vidugiriene, A., and Jakevicius, L. Lane marker parameters for vehicle's steering signal prediction. *Transactions on Systems* (2009). 22

[24] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B* (1977). 20

[25] Dickmanns, E. D. Vehicles capable of dynamic vision. In *Proc. IJCAI* (1997). 11

[26] Dickmanns, E. D. *Dynamic vision for perception and control of motion.* Springer, 2007. 10

[27] Dickmanns, E. D., Behringer, R., Dickmanns, D., Hildebrandt, T., Maurer, M., Thomanek, F., and Schiehlen, J. The seeing passenger car 'VaMoRs-P'. In *Proc. IV* (1994). 11

[28] Dickmanns, E. D., and Zapp, A. Autonomous high speed road vehicle guidance by computer vision. In *10th IFAC World Congress* (1987). 10

[29] Dollar, P., Wojek, C., Schiele, B., and Perona, P. Pedestrian detection: A benchmark. In *Proc. CVPR* (2009). 25, 74, 75

[30] Ess, A., Muller, T., Grabner, H., and van Gool, L. Segmentation-based urban traffic scene understanding. In *Proc. BMVC* (2009). 65, 71, 72

[31] Everingham, M., van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision* (2010). 74

[32] Fearnhead, P., and Liu, Z. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society, Series B* (2007). 23

[33] Fei-Fei, L., and Perona, P. A bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR* (2005). 31

[34] Fletcher, L., et al. The MIT-Cornell collision and why it happened. *Journal of Field Robotics* (2008). 14

[35] Gavrila, D. M., and Munder, S. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision* (2007). 74, 75

[36] Gepperth, A., Hasler, S., Rebhan, S., and Fritsch, J. Biased competition in visual processing hierarchies: An early fusion approach using multiple cues. *Cognitive Computation* (2011). 73, 74, 75, 80

[37] Gerdes, A. Automatic maneuver recognition in the automobile: The fusion of uncertain sensor values using bayesian models. In *Proc. WIT* (2006). 16, 18

[38] Gindele, T., Brechtel, S., and Dillmann, R. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In *Proc. ITSC* (2010). 20, 21

[39] Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Tech. rep., California Institute of Technology, 2007. 74

[40] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*. 2003. 28

[41] Haykin, S. *Neural networks: A comprehensive foundation*. Prentice Hall, 1999. 22

[42] Hel-Or, Y., and Hel-Or, H. Real time pattern matching using projection kernels. In *Proc. ICCV* (2003). 70

[43] Heracles, M., Martinelli, F., and Fritsch, J. Vision-based behavior prediction in urban traffic environments by scene categorization. In *Proc. BMVC* (2010). 6, 22, 24, 52, 62, 74, 89, 101, 102

[44] Hoiem, D. Seeing the world behind the image: Spatial layout for 3d scene understanding, 2007. PhD thesis, Carnegie Mellon University. 28

[45] Huang, X., Ariki, Y., and Jack, M. *Hidden markov models for speech recognition*. Edinburgh University Press, 1990. 17

[46] Jaakola, T. Tutorial on variational approximation methods. In *Advanced Mean Field Methods, Theory and Practice, MIT Press* (2000). 21

[47] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision* (1988). 25

[48] Kass, R., and Raftery, E. Bayes factors. *Journal of the American Statistical Association* (1995). 19, 23

[49] Kastner, R., Schneider, F., Michalke, T., Fritsch, J., and Goerick, C. Image-based classification of driving scenes by hierarchical principal component classification (HPCC). In *Proc. IV* (2009). 90, 99

[50] Kuge, N., Yamamura, T., Shimoyama, O., and Liu, A. A driver behavior recognition method based on a driver model framework. *SAE* (2000). 18, 21

[51] Kumagai, T., Sakaguchi, Y., Okuwa, M., and Akamatsu, M. Prediction of driving behavior through probabilistic inference. In *Proc. EANN* (2003). 17, 18

[52] Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML* (2001). 26

[53] Land, M. F., and Lee, D. N. Where we look when we steer. *Nature* (1994). 18

[54] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR* (2006). 31

[55] Lefevre, S., Ibanez-Guzman, J., and Laugier, C. Context-based estimation of driver intent at road intersections. In *Proc. ITSC* (2011). 21

[56] Leung, T., and Malik, J. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision* (2001). 67

[57] Li, L. J., Socher, R., and Fei-Fei, L. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Proc. CVPR* (2009). 32

[58] Liu, A., and Salvucci, D. Modeling and prediction of human driver behavior. In *Proc. HCI* (2001). 17, 18

[59] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* (2004). 23

[60] Martin, D., Fowlkes, C., and Malik, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence* (2004). 74

[61] Martinelli, F. Scene layout segmentation of traffic environments using a conditional random field, 2010. Master's thesis, University of Girona, Spain. 68, 69

[62] Maurer, M. Stadtpilot, Technische Universität Braunschweig, 2010. http://stadtpilot.tu-bs.de/en/stadtpilot/project/. 10, 15

[63] Maye, J., Triebel, R., Spinello, L., and Siegwart, R. Bayesian on-line learning of driving behaviors. In *Proc. ICRA* (2011). 23

[64] McCall, J. C., and Trivedi, M. M. Human behavior based predictive brake assistance. In *Proc. IV* (2006). 19, 22

[65] McCall, J. C., Wipf, D., Trivedi, M. M., and Rao, B. Lane change intent analysis using robust operators and sparse bayesian learning. *IEEE Transactions on Intelligent Transportation Systems* (2005). 22, 23

[66] Meyer-Delius, D., Plagemann, C., and Burgard, W. Probabilistic situation recognition for vehicular traffic scenarios. In *Proc. ICRA* (2009). 20

[67] Meyer-Delius, D., Plagemann, C., von Wichert, G., Feiten, W., Lawitzky, G., and Burgard, W. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *Proc. GFKL* (2007). 19

[68] Michon, J. A. A critical view of driver behaviour models: What do we know, what should we do? In *Human Behaviour and Traffic Safety, Plenum Press, New York* (1985). 16

[69] Miller, I., et al. Team Cornell's Skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics* (2008). 14

[70] Montemerlo, M., et al. Junior: The Stanford entry in the urban challenge. *Journal of Field Robotics* (2008). 14

[71] Moravec, H. Visual mapping by a robot rover. In *Proc. IJCAI* (1979). 10

[72] Moravec, H. The Stanford cart and the CMU rover. In *Proc. IEEE* (1983). 10

[73] Mourant, R. R., and Rockwell, T. H. Mapping eye movement patterns to the visual scene in driving: An exploratory study. *Human Factors* (1970). 18

[74] Murphy, K. P. Dynamic bayesian networks: Representation, inference and learning, 2002. PhD thesis, University of California, Berkeley, CA. 17, 20

[75] Nilsson, N. J. A mobile automaton: An application of artificial intelligence techniques. In *Proc. IJCAI* (1969). 10

[76] Oliva, A., and Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision* (2001). 54

[77] Oliver, N., and Pentland, A. P. Driver behavior recognition and prediction in a SmartCar. In *Proc. IV* (2000). 18

[78] Ommer, B., and Buhmann, J. M. Learning the compositional nature of visual objects. In *Proc. CVPR* (2007). 54

[79] Ortiz, M. G., Fritsch, J., Kummert, F., and Gepperth, A. Behavior prediction at multiple time-scales in inner-city scenarios. In *Proc. IV* (2011). 22

[80] Pentland, A., and Liu, A. Modeling and prediction of human behavior. *Neural Computation* (1999), 229–242. 17, 18

[81] Perronin, F., Dance, C., Csurka, G., and Bressan, M. Adapted vocabularies for generic visual categorization. In *Proc. ECCV* (2006). 30

[82] Pomerleau, D. A. ALVINN: An autonomous land vehicle in a neural network. In *Proc. NIPS* (1989). 12

[83] Pomerleau, D. A. Neural network perception for mobile robot guidance, 1992. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania. 10, 12

[84] Pugeault, N., and Bowden, R. Learning pre-attentive driving behaviour from holistic visual features. In *Proc. ECCV* (2010). 7, 22, 23, 24, 32, 52, 54, 55, 59, 73, 78, 88, 99, 102, 103

[85] Pugeault, N., and Bowden, R. Driving me around the bend: Learning to drive from visual gist. In *Proc. ICCV Workshop* (2011). 22, 24

[86] Quelhas, P., Monay, F., Odobez, J. M., Gatica-Perez, D., Tuytelaars, T., and van Gool, L. Modeling scenes with local descriptors and latent aspects. In *Proc. ICCV* (2005). 31

[87] Rabiner, L. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. IEEE* (1989). 20

[88] Rasmussen, J. Information processing and human-machine interaction: An approach to cognitive engineering. In *North-Holland, New York* (1986). 16

[89] Reed, R., and Il, R. M. *Neural smithing: Supervised learning in feed-forward artificial neural networks.* MIT Press, 1999. 22

[90] Roland, A., and Shiman, P. *Strategic computing: DARPA and the quest for machine intelligence, 1983 – 1993.* MIT Press, 2002. 10, 12

[91] Rother, C., Kolmogorov, V., and Blake, A. GrabCut – interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics* (2004). 25

[92] Russell, S., and Norvig, P. *Artificial intelligence: A modern approach.* Prentice-Hall, 2003. 20

[93] Salvucci, D. D., Mandalia, H. M., Kuge, N., and Yamamura, T. Lane-change detection using a computational driver model. *Human Factors* (2007). 18

[94] Scharstein, D., and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* (2002). 74

[95] Serrano, N., Savakis, A., and Luo, J. Improved scene classification using efficient low-level features and semantic cues. *Pattern Recognition* (2004). 30

[96] Shotton, J., Winn, J. M., Rother, C., and Criminisi, A. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. ECCV* (2006). 28, 67

[97] Sivic, J., and Zisserman, A. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV* (2003). 23, 30

[98] Stanford Racing Team (S. Thrun). Downloaded from http://cs.stanford.edu/group/roadrunner/old/presskit/high_res_pics/Stanley_Image2.jpg. 13

[99] Sturgess, P., Alahari, K., Ladicky, L., and Torr, P. H. S. Combining appearance and structure from motion features for road scene understanding. In *Proc. BMVC* (2009). 27, 28, 69, 71, 73, 82

[100] Szummer, M., and Picard, R. W. Indoor-outdoor image classification. In *ICCV Workshop on Content-Based Access of Image and Video Databases* (1998). 30

[101] Thorpe, C., Hebert, M. H., Kanade, T., and Shafer, S. A. Vision and navigation for the Carnegie-Mellon Navlab. In *PAMI* (1988). 10, 12

[102] Thrun, S. What we're driving at, Google Official Blog, 2010. http://googleblog.blogspot.de/2010/10/what-were-driving-at.html. 10, 15

[103] Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics*. MIT press, Cambridge, Massachusetts, 2005. 20

[104] Thrun, S., et al. Stanley: The robot that won the DARPA grand challenge. *Journal of Robotic Systems* (2006). 13

[105] TU Braunschweig / Institut für Regelungstechnik. Downloaded from http://stadtpilot.tu-bs.de/content/01-stadtpilot/01-project/scenario.png. 15

[106] TU Braunschweig / Presse und Kommunikation. Downloaded from https://www.tu-braunschweig.de/Medien-DB/presse/leonie/leonie2_tu-braunschweig.jpg. 15

[107] TU Braunschweig / Presse und Kommunikation. Downloaded from https://www.tu-braunschweig.de/Medien-DB/presse/leonie/leonie4_tu-braunschweig.jpg. 15

[108] Turk, M. A., Morgenthaler, D. G., Gremban, K. D., and Marra, M. Video road-following for the autonomous land vehicle. In *Proc. ICRA* (1987). 10

[109] Universität der Bundeswehr München (E. D. Dickmanns). http://www.unibw.de/lrt8/forschung-en/geschichte/marveye. 11

[110] Universität der Bundeswehr München (E. D. Dickmanns). http://www.unibw.de/lrt8/forschung/geschichte/hybrid_acc. 11

[111] Universität der Bundeswehr München (E. D. Dickmanns). http://www.unibw.de/lrt8/institut/mitarbeiter/prof_wuensche/vamp/image_original. 11

[112] Urmson, C., et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics* (2008). 14

[113] US Federal Government (DARPA). Downloaded from http://archive.darpa.mil/grandchallenge05/gcorg/index.html. 13

[114] US Federal Government (DARPA). Downloaded from http://commons.wikimedia.org/wiki/File:BeerBottlePass.JPG. 4

[115] US Federal Government (DARPA). Downloaded from http://www.darpa.mil/uploadedImages/Content/NewsEvents/Releases/2014/2005 Urban Challenge Entrant 6 - Hi Res.jpg. 13

[116] Vailaya, A., Figueiredo, A., Jain, A., and Zhang, H. Image classification for content-based indexing. *IEEE Transactions on Image Processing* (2001). 29

[117] Vailaya, A., Jain, A., and Zhang, H. On image classification: City vs. landscapes. *Pattern Recognition* (1998). 29

[118] van Gemert, J. C., Geusebroek, J. M., Veenman, C. J., and Smeulders, A. W. M. Kernel codebooks for scene categorization. In *Proc. ECCV* (2008). 32

[119] Veltri, L. Modelling eye movements in driving, 1995. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA. 18

[120] Vidugiriene, A., Demcenko, A., and Tamosiunaite, M. Steering angle prediction using neural networks and look-up table for different drivers. In *Proc. SPIE* (2009). 22

[121] Wang, Y., and Gong, S. Conditional random field for natural scene categorization. In *Proc. BMVC* (2008). 32

[122] Whittaker, R. Red team robot racing, Carnegie Mellon University, 2005. http://www.cs.cmu.edu/∼red/Red/index.html. 13

[123] Wojek, C., and Schiele, B. A dynamic conditional random field model for joint labeling of object and scene classes. In *Proc. ECCV* (2008). 27, 72