Lehrstuhl für Realzeit-Computer Systeme

Merging the Virtual and Real in a Car: In-Vehicle Augmented Reality

Qing RAO

Vollständiger Abdruck der von der
Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität München zur Erlangung des akademischen Grades
eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigten Dissertation.

Vorsitzende/-r: Prof. Dr.-Ing. habil. Gerhard Rigoll

Prüfende/-r der Dissertation:

1. Prof. Dr. sc. Samarjit Chakraborty

2. Prof. Dr. Jian-Jia Chen

Die Dissertation wurde am 18.04.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 08.03.2019 angenommen.

INSTITUTE FOR REAL-TIME COMPUTER SYSTEMS
TECHNISCHE UNIVERSITÄT MÜNCHEN
PROFESSOR SAMARJIT CHAKRABORTY

Merging the Virtual and Real in a Car: In-Vehicle Augmented Reality

Qing Rao

Dissertation

Acknowledgments

First and foremost, I would like to express my sincere gratitude and appreciation to my advisor Prof. Dr. sc. Samarjit Chakraborty for his continuous guidance and support, especially during the time period when I had difficulty continuing this thesis. This thesis would not have been possible without his encouragement, enthusiasm, and dedication in teaching me how to conduct research and write scientific articles. It has been a great honor for me to have him as the advisor and mentor throughout these years.

My sincere thanks to Prof. Jian-Jia Chen for his willingness to be the second advisor and reviewer of this thesis, for his encouragement, as well as his insightful comments.

I am deeply grateful to Joachim Gloger for accepting me in his research group during the internal reorganization at Daimler AG. I greatly appreciate the freedom and the trust he gave me to continue conducting my research.

Also, I would like to thank Dr. Clemens Rabe, Dr. David Pfeiffer, and Dr. Lars Krüger for opening the door to image understanding to me, for giving me valuable advices throughout my work, and for showing me the way of being an exemplary professional engineer that I wished to be.

My special thanks to Markus Braun and Fabian Flohr for sharing the knowledge of object detection with me and for the great joint work on Pose-RCNN.

Last but not least, I wish to thank my parents and my family for their never-ending support and encouragement throughout my life. It is thanks to their upbringing throughout my childhood that I became what I am today.

For My Parents and Grandparents.
For Family Rao.

Beijing, Spring 1990.

Contents

List of Figures

List of Tables

List of Symbols

ADAS	Advanced Driver Assistance System
AI	Artificial Intelligence
AOS	Average Orientation Similarity
AP	Average Precision
AR	Augmented Reality
CAD	Computer-Aided Design
CAN	Controller Area Network
CCD	Charge Couple Device
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
CTRV	Constant Turn Rate and Velocity
DCT	Discrete Cosine Transform
DP	Dynamic Programming
DTAM	Dense Tracking and Mapping
E/E	Electrical / Electronic
ECU	Electronic Control Unit
EKF	Extended Kalman Filter
FCN	Fully Convolutional Network
FOV	Field Of View
FPGA	Field Programmable Gate Array
GIF	Graphics Interchange Format
GP	Gaussian Process
GPLVM	Gaussian Process Latent Variable Model
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HMD	Head Mounted Display
HMI	Human Machine Interface
HUD	Head Up Display
IMU	Inertial Measurement Unit
LIN	Local Interconnect Network
LSQ	Least SQuares
MAP	Maximum A Posteriori
MLE	Maximum Likelihood Estimation

MOST	Media Orient Systems Transport
MR	Mixed Reality
NDC	Normalized Device Coordinate
OEM	Original Equipment Manufacturer
OLED	Organic Light-Emitting Diode
PCA	Principal Component Analysis
PMF	Progressive Morphological Filter
PNG	Portable Network Graphics
POI	Point Of Interests
RanSaC	Random Sample Consensus
R-CNN	Region-based Convolutional Neural Network
RCS	Real-time Computer Systems
RD	Research & Development
RNN	Recurrent Neural Network
ROI	Region Of Interest
SCG	Scaled Conjugate Gradient
SDF	Signed Distance Function
SGM	Semi-Global Matching
SLAM	Simultaneous Localization And Mapping
TTC	Time To Collision
TUM	Technical University of Munich
VR	Virtual Reality

Abstract

Augmented Reality (AR) is a technology that superimposes a computer-generated virtual image on a user's view of the real world, thus merging the *Virtual* and the *Real*. Over the last few decades, AR applications have expanded from the very first military, industrial, and medical applications, into entertainment, education, and gaming industry, gradually entering our daily life. The ever-growing importance of augmented reality has awakened special interest of the automotive industry to also apply it to a production car.

Today, despite the success of augmented reality in consumer electronics, AR in a car is still difficult to realize than AR on a mobile device. First of all, the requirements on functional safety, real-time capability, and robustness of an AR function (algorithm) in the automotive industry is in another dimension. Also, an in-vehicle AR application needs to be distributed into different domains of the in-vehicle Electric/Electronic (E/E) system, which, however, does not support transmitting massive amount of data today. In order to incorporate augmented reality in a production car, both of the existing AR functions and the in-vehicle E/E architecture have to be refined or redesigned.

This thesis presents a roadmap toward practical implementations of in-vehicle augmented reality. It discusses in detail how fundamental understandings, including *depth understanding* and *scene understanding*, enable a series of AR functions such as localization, tracking, and rendering for in-vehicle augmented reality. In particular, this thesis makes three following scientific contributions.

First, within the scope of depth understanding, we propose a compression scheme for a compact mid-level representation of depth information named *Stixel*. The proposed Stixel compression algorithm reduces both temporal and spatial redundancies in Stixel data using a combination of predictive modeling and entropy coding. It enables transmitting depth information through a reasonably priced in-vehicle bus, which solves a key problem for incorporating in-vehicle augmented reality.

Second, within the scope of scene understanding, we propose a workflow to 3D shape reconstruction named *3D Shaping* as the first step towards building a three-dimensional environmental model at an object level, which is essential for augmented reality. The 3D Shaping workflow adapts an existing silhouette-based reconstruction algorithm to jointly estimate the pose, the scale, and the 3D geometry of an object that is detected in an input image. The estimation step takes advantage of an extremely low-dimensional representation of 3D geometries, which allows distributing the workflow into different domains of the in-vehicle E/E architecture without significantly increasing the payload requirement on the in-vehicle communication medium. We additionally propose two extensions of 3D Shap-

ing in order to make it more practical for real traffic scenes. Relying on the proposed 3D Shaping and its extensions, truly 3D augmentation of the real world becomes possible for in-vehicle AR.

Last but not least, within the scope of system design and integration, we propose three different designs of in-vehicle AR systems, respectively for the current generation, next generation, and future generation of production cars. We continuously integrate our research findings into the designed AR systems, and we fully exploit depth understanding and scene understanding for in-vehicle AR in our designs. We expect our proposed future AR system to be seriously considered by industrial decision makers and eventually adopted in series production cars in the future.

The value of in-vehicle AR will continuously grow in the future, especially when drivers do not have to drive anymore by themselves. Relying on new sensor technology, display technology, and Artificial Intelligence (AI), in-vehicle augmented reality will become more connected, more sophisticated, and more "real". We hope this thesis could catch the attention of our peers in both academic research and industry so that more researchers and developers would devote themselves to AR. We firmly believe in a bright future of in-vehicle augmented reality.

1 Introduction

Now I do not know whether it was then I dreamed I was a butterfly, or whether I am now a butterfly dreaming I am a man.

Zhuang Zhou
c. 369 BC - c. 286 BC

Mankind has never ceased to question the reality of the world they live in. As early as the 4th century BC, the ancient Chinese philosopher Zhuang Zhou [Wat03] already raised the doubt about his existence after he dreamed about himself being a butterfly fluttering in the virtual world in his dream. In a general sense, the term *Virtual* involves everything that does not exist in the physical world, from dreams, myths, to fictional universes. It reveals the desire of human beings to emancipate themselves from the *Real*, which is most of the time perceived as being boring, banal, and monotonous. This desire has motivated countless researchers and scientists to keep exploring new possibilities for closing the gap between the virtual and the real. As an inevitable result of their efforts, *Augmented Reality* (AR) emerged.

This chapter introduces the background, the key problems, the way of thinking, the structure, and the key contributions of the thesis. We briefly review the history and related work of augmented reality in Section 1.1 and Section 1.2, respectively. In Section 1.3, we provide an overview of in-vehicle E/E architecture, with the main focus on the telematics domain and the Advanced Driver Assistance System (ADAS) domain, which are the most relevant domains for incorporating in-vehicle AR. In Section 1.4, we explain the key problems that make it impossible for us to realize in-vehicle augmented reality without any modification of the E/E architecture. Three key problems are defined in this section, which we address later in the main body of the thesis. In Section 1.5, we introduce our way of thinking, i.e., how we approach the objective of designing and integrating an AR system in a production car. Instead of focusing on individual functions in an AR system, we rather put emphasis on two fundamental understandings, namely *depth understanding* and *scene understanding*, based on which AR functions and systems are built. We present the outline and the contributions of this thesis in the last two sections, Section 1.6 and Section 1.7, respectively.

1.1 History of Augmented Reality

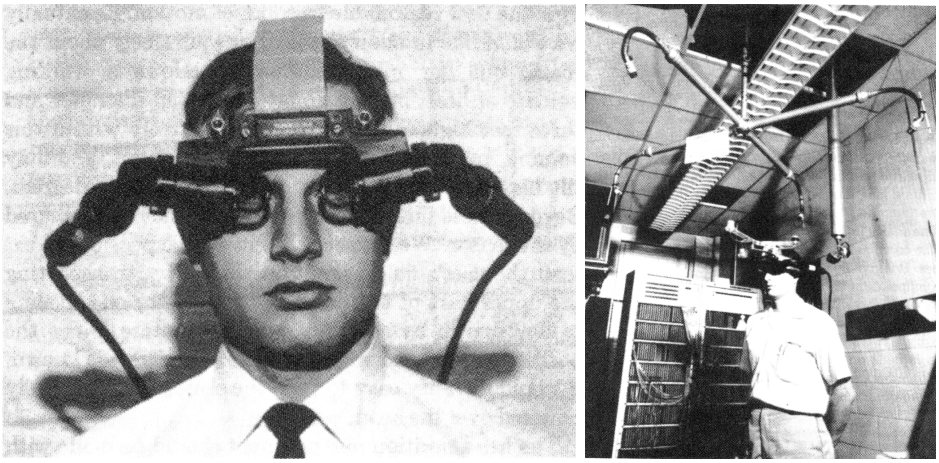


Figure 1.1: Head-mounted display created by Sutherland [Sut68] in 1968. Images are extracted from [Sut68] and adapted by the author.

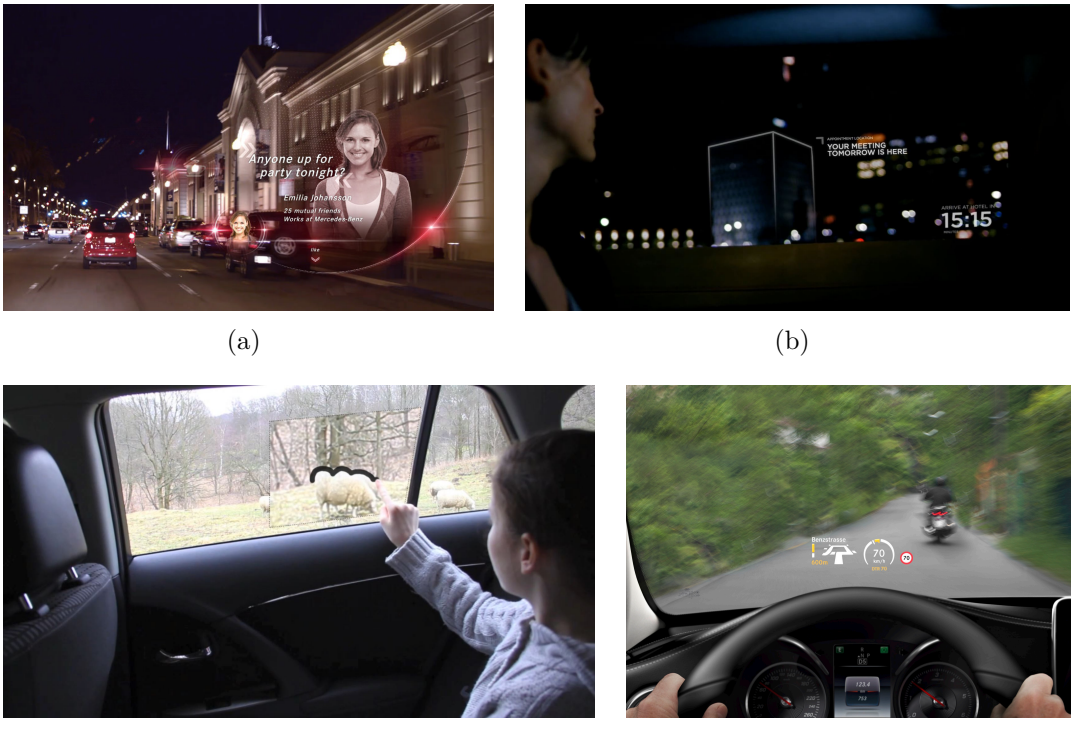
The first use of the term "Augmented Reality" was attributed to the former Boeing researcher T. Caudell [JMC93] in 1990. He used it to describe the enhancement of human perception of reality through computer generated virtual objects. The first mention of this kind of idea was much earlier, as L. Frank Baum [Fra01] introduced the "character maker" in his novel *The Master Key* in 1901. The character maker comprises a set of spectacles that overlay a label onto a person's forehead. The label reflects the persons character, e.g., 'G' for *Good* or 'E' for *Evil*. About 60 years later, in 1968, the idea in [Fra01] was realized by Sir Sutherland [Sut68]. He created a new type of three-dimensional display (Fig. 1.1), known as Head-Mounted Display (HMD) today, which is literally mounted on the head of a user. The HMD shows a virtual scene that changes along with the position and the orientation of the head. This gives a user the impression as if he or she were "inside" the virtual scene. However, quite the opposite of a modern day HMD, the wearer back then was rather tethered to a workstation confined to a fixed location instead of really wearing the HMD. Another decade later, in 1980, S. Mann [Man97] built a wearable computer with a stereo HMD, which is considered the world's first augmented reality wearable device. These are the pioneers who made fundamental contributions to augmented reality in the early years.

Relying on fast booming hardware and software technologies in the past three decades, augmented reality has experienced a rapid development. Today, there is a wide range of AR applications in many different fields including video games [Nia12, Lyt15, Nia16], commerce [L'O15], education [Ter16, Exp16], industrial design and assembly [VSH07, WON16], military [CDA+05], and tourism [BM06]. Google Glass [X D15] and Microsoft Hololens [Mic16] belong to the most well-known commercially available AR wearable devices. In 2016, the AR game Pokémon Go [Nia16] swept the globe by lifting the players off their couches and forcing them to go outside in order to catch Pokémons, a virtual creature who only appears on the screen if the player moves him- or herself physically close

to some specific GPS locations in the real world, known as "spawn points" of Pokémons. Augmented reality has already started to change the way how people acquire information, play games, and design product among others. The gap between the virtual and real is gradually closing.

1.1.1 AR in the Automotive Industry

Early AR projects in the automotive industry date back to the 90's. ARVIKA [Fri02] was one of the most influential pioneer projects. Supported by the German Federal Ministry of Education and Research, the primary objective of this project was to implement an augmented reality system for mobile use in industrial applications in a wide range of fields spanning from development to production and service. Germany's three major car manufacturers, namely Daimler, BMW, and VW were involved in this project. Since then, a number of AR projects were spun out of the automotive industry [RBW05]. AR applications such as engine maintenance, wiring, and safety drive training were prototyped and demonstrated.



(a) (b)

(c) (d)

Figure 1.2: In-vehicle AR concepts, demonstrators, and products. (a) Daimler AR demonstrator DICE [Dai12]. (b) Microsoft productivity future vision [Mic11]. (c) Toyota AR concept Window to the World [CT11]. (d) Head-up display available in Mercedes-Benz C-Class 2014 [Dai13]. Image source: (a)(d) Daimler AG, (b) Microsoft Corporation, (c) Toyota Motor Corporation.

Coming into the modern era, in early 2012, Mercedes-Benz [Dai12] exhibited the Dynamic and Intuitive Control Experience (DICE) demonstrator, where the entire windscreen of a car was conceptualized as a single augmented reality Head-Up Display (HUD). Only gestures were needed for controlling and interacting with the system. One year later, Mercedes-Benz [FPR+13] again presented a prototype vehicle, which drove itself "without a real driver" on the historical route from Mannheim to Pforzheim in Germany. This project, although not directly schemed for augmented reality, opened up new possibilities for in-vehicle AR to provide the "engineering view" to the driver and thus enabling him or her to monitor, view, and understand the decisions of a self-driving car.

Despite the growing popularity of in-vehicle augmented reality in the press (Fig. 1.2), AR has still not been truly integrated into vehicle platforms. There are certain specific automotive requirements that have to be met during the stage of system design, such as real-time and robustness requirements on the algorithms, power consumption and communication bandwidth constraints on the Electrical/Electronic (E/E) systems, etc. The automotive industry is still seeking new solutions to incorporate AR in a production car. On the one hand, AR technologies have to be further developed in order to improve their robustness and reliability. On the other hand, the in-vehicle E/E architecture needs to be modified or redesigned in order to support computationally expensive AR algorithms and enable transmitting data across different in-vehicle domains. These challenging tasks still remained unaddressed at the time point when we started this thesis.

1.2 Related Work

In 2014, Gabbard et al. [GFK14] published a comprehensive review of challenges and opportunities for incorporating augmented reality into automotive applications. According to [GFK14], the opportunities of in-vehicle AR lie above all in supporting interactions between the driver and the car, e.g., navigation aids through a windshield HUD, whereas a number of critical technical challenges including tracking, depth perception, and 3D scene registration need to be overcome.

Opportunities As in [GFK14], most of the existing studies of in-vehicle augmented reality [TFO+94, DCT09, KD09, PDT+09, CPMA11, KWGP13, MCJS13] focused on AR user interaction using a head-up display. The advantage to augment additional information directly on the windshield of a car is, that the driver does not have to shift his or her attention away from driving anymore while accomplishing additional driving related tasks in the same time, e.g., finding the next way point or checking the speed limit. Several examples of using AR based navigation and driver assistance systems are presented in [DCT09, KD09, KWGP13, MCJS13]. Also, AR navigation and driver assistance at night [Ber08] is of great interest and importance to the automotive industry.

Currently, commercially available in-vehicle AR systems are already able to provide the afore-mentioned features to some extent. Several Original Equipment Manufacturers

(OEMs) including Daimler [Dai13] and BMW [BMW11a, BMW11b] already integrated head-up displays in their premium segment production cars. Automotive suppliers such as Pioneer [Pio12] and Harman [Har16] also offer external HUDs that can be mounted overhead or to the dashboard. Nevertheless, the current generation of HUDs are only able to show static overlays instead of contact-analog augmentations.

In the meanwhile, the number of sensors equipped in a production car has gradually increased. Additional sensors including camera, Radar, and Lidar are already widely used in pilot projects of self-driving cars [FPR+13, DAK+15, Way16]. These sensors would be eventually available in future generation of series production cars, enabling them to capture the 360° surroundings (Fig. 1.3). This offers another opportunity to in-vehicle AR by providing more precise and comprehensive measurements of the world.

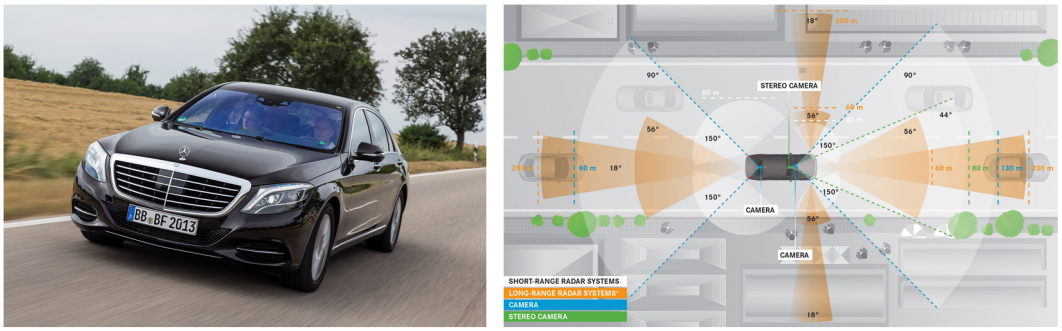


Figure 1.3: Sensor setup of the prototype vehicle "Bertha" [DAK+15] comprising a forward-looking stereo-vision system, a backward-looking monocular vision system, and eight Radar sensors covering 360° surrounding environment of the ego-car up to 200 meters. Image source: Daimler AG.

Challenges Despite the variety of automotive AR products being offered, augmented reality itself is still not a mature technology, let alone to be integrated into a vehicle platform.

In recent years, progresses have been made in AR enabling technologies including tracking [Bad04, NNB04, YKN06, How08], localization [DNC+01, LT10], depth estimation [Hir05, GEM09], 3D scene reconstruction and registration [PSR12, DPRR13, KTCM15], head pose and gaze point estimation [TKBO11, ZLR12], and realistic rendering [HDH03, UIS+09, dLLT12]. There are also a number of studies [Tuf97, HU04, GSZW10, GTFC12, TMH+12, GBR14] focusing on applying these technologies in a car. These studies revealed some of the challenges in transferring indoor AR technologies, e.g., tracking, to a large-scale outdoor environment. The hardware platform carrying indoor AR applications is usually a smartphone or a see-trough head-mounted display, where head tracking and gaze point estimation are not required. In a car, however, head and gaze point tracking is critical to the user experience while augmenting contact-analog information on the windshield, e.g., blending a navigation arrow on the surface of the road in the real world. Due to the large variety of traffic scenes, weather conditions, and driving styles, existing

AR technologies cannot be easily transferred into a car without improving the algorithms or redesigning the hardware. This is one of the biggest technical challenges in realizing in-vehicle AR.

Despite those afore-mentioned technical challenges, another hurdle of incorporating AR in a car finds itself in the high-demanding automotive requirements. The state-of-the-art AR enabling technologies are often computationally expensive, produce large amount of 3D data, and are not real-time capable. In contrast, the automotive E/E system is characterized through its heterogeneity, computational efficiency, and low-bandwidth communication media. As long as no significant upgrade is applied that enables data-intense computing and transmission on the in-vehicle E/E architecture, the trade-off has to made between the performance of the algorithm and the cost of modifying the specification of the in-vehicle E/E systems. From the point of view of system architects, an entire AR workflow has to be distributed to different Electronic Control Units (ECUs) in a car. In addition, the amount of data that has to be transmitted through the in-vehicle bus needs to be analyzed, and the latency from the sensors to the AR displays has to be minimized. These questions still remain open to the automotive industry.

In reviewing the literature of augmented reality, we realized that at the time point when we started this thesis, there was very limited research concerning both optimizing AR technologies and modifying or redesigning the in-vehicle E/E architecture for in-vehicle AR. To our best knowledge, our work in this thesis is the first of its kind that addresses these issues.

1.3 In-Vehicle E/E Architecture

In-vehicle Electric/Electronic (E/E) architectures are heterogeneous and distributed systems comprising a large number of *Electronic Control Units* (ECUs) and the communication systems connecting them. In some flagship vehicles, there are more than 100 ECUs controlling the engine, the brake, the seats, and the infotainment system. Figure 1.4 shows a typical E/E architecture of a modern vehicle. The ECUs in a car form four major domains according to their main functionalities. These include *body* domain that provides comfort related functions, *powertrain* domain that controls the engine, *ADAS* domain that helps improve driver safety, and *telematics* domain for in-vehicle telecommunication and infotainment. All of them share a central gateway which relays signal from one domain to another.

The communication between ECUs is carried out through a specialized communication network named *in-vehicle buses*. Since its first introduction in the late 80's, various communication standards were developed in order to meet special requirements of different domains. For example, data transmission in the ADAS domain is crucial to the safety of the driver or the passengers and therefore has to be hard real-time capable. In contrast, transmitting a control signal within the body domain is more tolerate to transmission error or packet loss. There are four most widely deployed vehicle bus standards today,

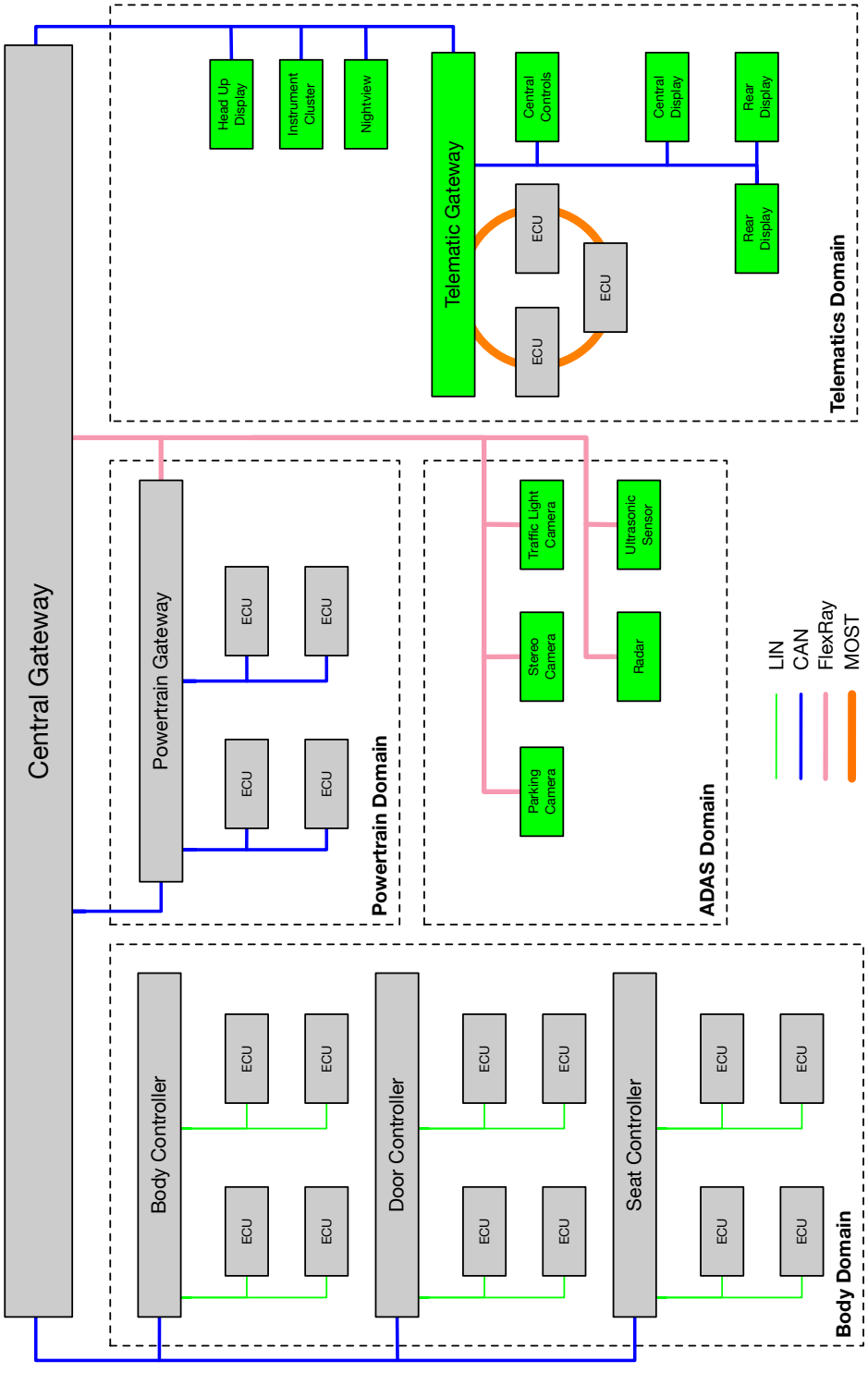


Figure 1.4: Typical E/E architecture of a modern vehicle. ECUs related to in-vehicle AR are marked green. Direct connections between cameras and displays are not illustrated.

namely Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay, and Media Orient Systems Transport (MOST). Table 1.1 gives a brief comparison of them. For more details of in-vehicle buses, please refer to [ZS11].

Table 1.1: Comparison of in-vehicle bus standards [ZS11]. Data source: [ZS11].

Bus	Bandwidth	Application Domain	Example Use Case	Cost
LIN	$\leq$ 20 kbps	Body	Door lock	Low
CAN	$\leq$ 500 kbps	Powertrain	Engine control	Medium
FlexRay	$\leq$ 10 Mbps	ADAS	X-by-wire	High
MOST	$\leq$ 25 Mbps	Telematics	Video streaming	High

1.3.1 Telematics Domain

The French origin télématique of the term telematics merges télécommunication and informatique, which refers to the transfer of information via telecommunications. In a car, it involves satellite navigation, radio, Bluetooth telephony, and wireless communication. More generally, in-vehicle telematics also includes input and output hardware devices for information transfer such as wheel buttons, number pads, and various in-vehicle displays.

The telematics domain provides an ideal basis for applying in-vehicle augmented reality. First, virtual contents to be augmented, e.g., virtual points of interest, need to be constantly updated. This requires wireless communication between a car and online content providers, which are provided by the telematics domain. Second, there are abundant options in the telematics domain available for designing the Human-Machine-Interface (HMI) of an AR system. It is more intuitive to use buttons in the telematics domain, e.g., a wheel button or a control element in the central panel, to interact with the AR system instead of using a seat button. Last but not least, the core functionalities of several AR customer features, e.g., navigation, are already available in the telematics domain. Hence, we do not need to re-implement them from the scratch. These give certain advantages to the telematics domain against other domains while considering the carrying domain for in-vehicle AR.

1.3.2 ADAS Domain

ADAS domain comprises ECUs, in-vehicle sensors, and actuators that help a driver in accomplishing various driving tasks and preventing traffic accidents. Several well-known ADAS applications include adaptive cruise control, parking assist, and lane-keeping warning. Traditional ADAS uses cameras, ultrasonic sensors, and Radars to perceive the environment and detect obstacles. Recent research projects also added Lidar to the ADAS in order to capture precise 3D measurements of the surrounding environment.

The ADAS domain is also important for in-vehicle augmented reality. One of the keys to merging the virtual and the real is an accurate alignment of real and virtual objects.

The better the three-dimensional environment is understood and reconstructed, the more realistic the blended output can be. In other words, an AR system requires algorithms that detect, classify, and track real-world objects in real time. In the ADAS domain, a number of algorithms to object (car, pedestrian, and traffic sign) detection and tracking are already available. These could be reused by an in-vehicle AR system.

1.4 Problem Statement

AR pioneers Azuma et al. [ABB+01] defined an AR system, independent of its carrying platform, to share the following three properties.

- Blends real and virtual, in a real environment.
- Real-time interactive.
- Registered in 3D.

There are several issues of the current generation of in-vehicle E/E architecture which makes it impossible to incorporate an AR system that meets all of the three properties. For example, the in-vehicle front-looking camera is originally designed for driver assistance and does not directly stream images to the displays. Instead, it evaluates input images through computer vision algorithms and only sends out the processed information, e.g., recognized traffic signs or road surface markings, to the vehicle bus. Without modifying the E/E architecture, there is no input source of the *Real* available to the AR system, let alone blending the Real and the Virtual. Besides, 3D registration is still an active research area in both academic research and industry. Robust and large-scale registration algorithms that are applicable to an in-vehicle system are yet to be developed. Even if an algorithm was ready to be applied, it could easily overwhelm the computational and memory capacity of any ECUs by its complexity. In addition, the requirement of real-time interactivity makes it even more difficult to incorporate AR in a car. It would be for example a poor interaction experience if zooming on a map using AR navigation would take more than a second.

By analyzing the current generation of the in-vehicle E/E architecture, we crystallized three key problems for in-vehicle augmented reality which we address in the main body of this thesis. These include the transmission of depth information through in-vehicle bus, object-level 3D reconstruction of the surrounding environment, and the engineering problem of integrating designed AR systems into prototype vehicles.

1.4.1 Key Problem: Transmission of Depth Information

Vision-based advanced driver assistance systems are increasingly being used in modern production cars. Towards this, stereo vision systems are used to capture the 3D surroundings of the ego-cars. Using stereo matching algorithms [Hir05, GEM09], depth information

is generated, processed, or stored for a short period of time in the vision processing system for further use, such as building an environmental model for in-vehicle augmented reality (Fig. 1.5).

(a) Original image. (b) Disparity image. (c) Point cloud.

Figure 1.5: Representations of depth information.

Stixel [BFP09, PF10, PF11, PGS13] is a data structure that was introduced about ten years ago to model depth information in traffic scenes. The term "Stixel" was created by combining "Stick" and "Pixel", which reflects the fact that a Stixel frames a thin rectangular area (like a stick) in a depth image with similar disparity (pixel) values (Fig. 1.6). Based on the assumption that the geometry in common traffic scenes comprises predominantly vertical and horizontal surfaces, the Stixel representation finds a balance between the compactness and informativeness of the depth information.

Figure 1.6: Stixel representation of a traffic scene. Distances are encoded in color, from red being close to green being far away. Image source: [PF11].

Even using the Stixel representation, the depth data volume of the complete scene is still too huge for a low- or medium-bandwidth in-vehicle communication system, e.g., a CAN [Int03] bus. In other words, in today's commercially available cars, depth information still remains in the in-vehicle vision system, without being accessible to the other components in the car due to cost reasons. This limits the development of a number of applications such as augmented reality that requires depth information of the surrounding environment. Hence, the automotive industry is still seeking a solution that makes depth

information available to other in-vehicle processing units via a less expensive communication bus. This is the first key technical problem that needs to be solved for in-vehicle augmented reality.

1.4.2 Key Problem: Object-level 3D Reconstruction

Recall that Azuma et al. [ABB+01] required an AR system to be able to register the 3D environment. A key approach to 3D registration is referred to as *object-level 3D reconstruction* [RBM+07, RMB+09], which aims at building a 3D environmental model at an object level. Relying on object-level 3D reconstruction, in-vehicle AR applications are able to truly augment the 3D world. Figure 1.7 shows a rendered example of object-level reconstruction of a car's surroundings. In this example, the driver of the ego-car entering the intersection cannot see the other car behind the bush. If the 3D pose (position and orientation) and 3D geometry of every objects in this scene are precisely reconstructed and registered, we can overlay a warning sign onto the hidden car through AR and thus bring it back to the driver's notice.

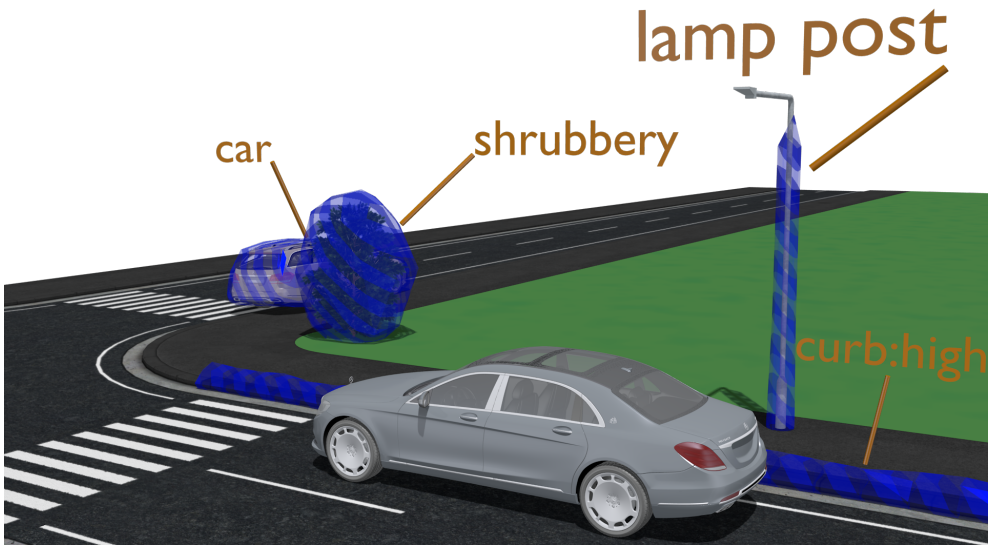


Figure 1.7: Example object-level 3D reconstruction of a car's surroundings.

There are four stages in building such an object-level environmental model. The first stage (Fig. 1.8(b)), referred to as *2D Detections*, aims at detecting one or multiple targeted objects in a 2D input image. It is also known as *object bounding box detection*. The second stage *2D Semantics* (Fig. 1.8(c)), also known as *image segmentation*, *pixel-level classification*, or *instance segmentation*, classifies every pixel in a 2D input image. This requires a more detailed understanding of the scene in the image, i.e., how the scene is structured. The third stage *3D Semantics* (Fig. 1.8(d)) extends 2D semantics to the three-dimensional world with the help of additional sensors such as stereo camera or Lidar. By back projecting image cues to the 3D space and combining them with 3D measurements,

a comprehensive reconstruction of the scene can be achieved, enabling further processing such as occlusion reasoning and surface reconstruction among others.

The fourth and the most important stage is referred to as *Geometry Modeling*, which fits 3D geometries (shapes) of the detected objects into the reconstructed 3D scene. Now, the 3D scene can be represented by a list of objects with their 3D poses and shapes instead of a huge amount of 3D points – hence, the name of object-level reconstruction. This enables a more compacted representation of the 3D environment.

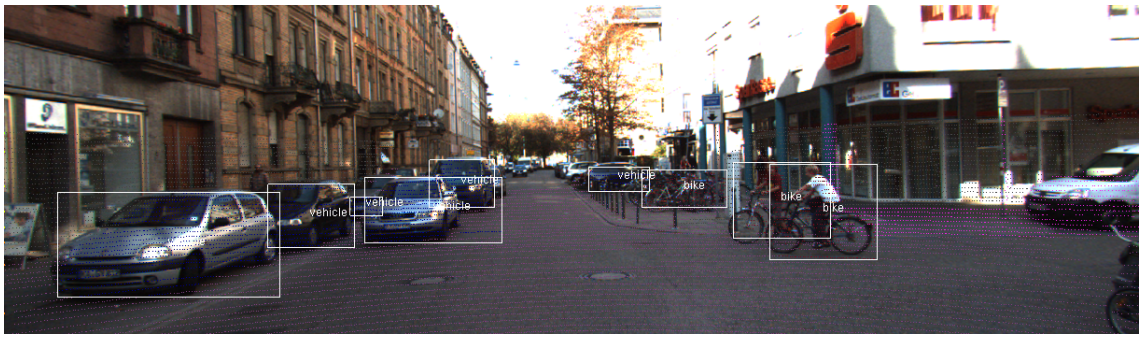
So far, all core functionalities for object-level 3D reconstruction are provided by the ADAS domain in a car. However, as already elaborated in Section 1.3.1, the hardware components for HMI (displays, buttons) are provided by the telematics domain. Hence, we need a sophisticated solution for shipping 3D geometries through the in-vehicle communication system. This is the second key technical problem for in-vehicle augmented reality.

1.4.3 Key Problem: In-Vehicle Proof of Concept

The modern in-vehicle E/E architecture is a highly complex system consisting of a considerable number of sensors, actuators, and ECUs, as well as the in-vehicle buses connecting them. Decades of research have been conducted in order to enable all of the in-vehicle components functioning as a whole. Several related research topics in the area of system design, modeling, and simulation include arithmetic expressions, heterogeneous composition, clock synchronization, and modeling delays. Beyond that, the validation, integration, and testing of a designed and implemented system in a test vehicle are also proven to be a huge challenge that requires years of engineering experiences.

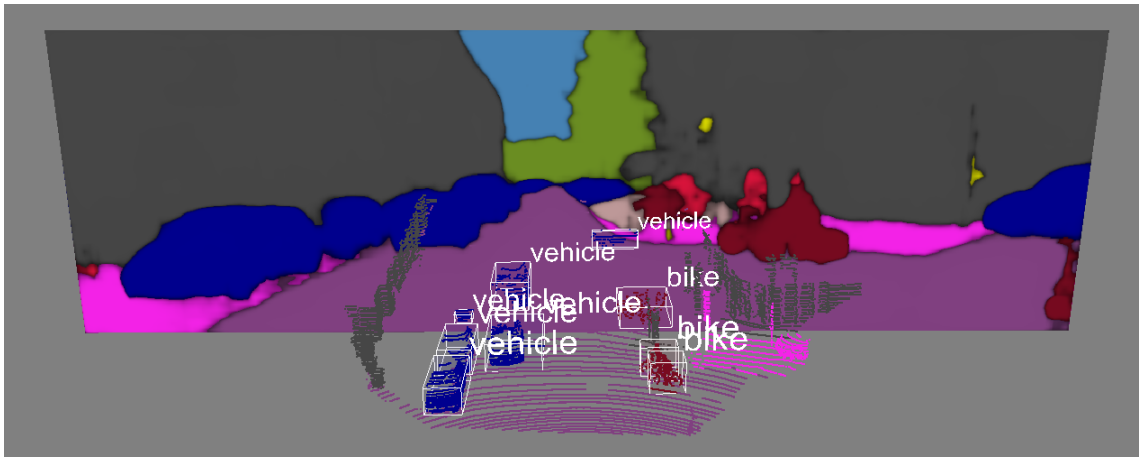
So far, in the automotive industry, it typically takes five to seven years for a new feature from the research phase to the actual production launch. In the early phase of this time span, prototypes are usually built for proof of concept. Based on the evaluation of the prototypes, the decision makers would decide whether to continue the development at all or how much resource should be further allocated if decided to continue. These kinds of decisions are typically hard to taken since it might have serious strategic consequences influencing the competitiveness of the manufacturer. Specifically, for in-vehicle augmented reality which is a technology that is not considered mature yet, it is even harder to make an early decision regarding the upgrade of the E/E architecture and the customer features to be provided based thereon. On the one hand, the robustness of currently existing AR functions such as tracking and localization needs to be improved, in oder to meet the technical requirements of the automotive industry. This requires further research and algorithmic development in the fundamental understandings, e.g., depth understanding and scene understanding. On the other hand, the current generation of the E/E architecture needs to be updated or redesigned, in order to support an AR system that is yet to be developed. In other words, at the current time point, it is impossible for a system architect to fix the specification for the in-vehicle E/E architecture for the next five years while the technology of AR itself is still under development.

(a) Input image.



(b) First stage: 2D detections.

(c) Second stage: 2D semantics.



(d) Third stage: 3D semantics.

Figure 1.8: The first three stages of building an object-level environmental model.

Given all the afore-mentioned constraints and determining factors, it is clearly infeasible to address all of the afore-mentioned research topics within the scope of this thesis. Therefore, we define the key problem here from the following two perspectives. First, we aim to prove the basic concept of AR in a test vehicle using the currently available technologies as far as possible. In parallel, we intend to continuously integrate new findings during the research on depth and scene understanding into an AR prototype system. This engineering problem is the last and the most important key problem of the thesis.

1.5 Think Model

This thesis aims at enabling in-vehicle augmented reality in series production cars. To be more precise, it does not focus on individual algorithms or technologies, referred to as *AR functions*, that support in-vehicle AR. Since AR functions cover a variety of subjects from 3D geometry, computer vision, computer graphics, to navigation, and optical physics, it is impossible to attack all of them and make a breakthrough in each AR function within the scope of this thesis. Also, from the view point of an OEM, our task consists in specification and requirements engineering instead of developing and implementing single functions. These are the reasons why we decided to focus on a more fundamental layer of AR in this thesis and tackle the afore-mentioned key problems which are critical for incorporating augmented reality in a car.

Nevertheless, it is impossible to discuss the fundamental understandings without introducing AR functions. Therefore, in Section 1.5.1, we begin with three of the most important AR functions for in-vehicle augmented reality, namely *localization*, *tracking*, and *rendering*. Subsequently, in Section 1.5.2, we stress the importance of depth understanding and scene understanding for AR and show how these two fundamental understandings support the introduced AR functions.

1.5.1 AR Functions

Localization Localization aims at answering the question "Where am I?". It applies both global and local positioning techniques in order to achieve certain required localization precision. It is essential that an in-vehicle AR system be able to determine the precise global position of the ego-car, since virtual objects such as virtual Point Of Interests (POIs) provided by most of the content providers are defined in the global coordinate system. According to [U.S17], as of late 2016, Standard Positioning Service (SPS) GPS receivers only provide a horizontal accuracy around 1.9 meters. The vertical accuracy is twice as much. In order to increase the GPS accuracy, enhanced systems such as Differential GPS (DGPS) and Real-Time Kinematic GPS (RTK-GPS) are deployed, enabling a car to navigate at lane level [DB08] or in areas with poor or without GPS reception [LGSB12]. Yet in urban areas, especially in areas with tall buildings, the performance of pure GPS-based localization methods are usually unsatisfying due to multipath infer-

ences. Recently, hybrid localization methods based on Simultaneous Localization And Mapping (SLAM) [DNC+01, LT10] successfully solved this problem by combining classic localization techniques with (visual) tracking.

Tracking The problem of tracking involves self-tracking and tracking of external objects. It enables retrieving and updating the relative pose between the ego-car and surrounding objects. A straightforward solution for self-tracking accumulates data from motion sensors, e.g., wheel odometer. However, wheel odometry is vulnerable against wheel drift, since once the vehicle loses track at one point, it will never be able to correct itself again. The drifting effect can be compensated using visual odometry [NNB04, How08], where feature points [LK81] extracted from consecutive image frames are first matched, and optical flows [BB95] are then calculated based on the matched feature points. In other words, drifts are detected and compensated visually using visual odometry. The robustness of visual odometry is increased by eliminating outliers using RANSAC [FB81] or by using a Kalman Filter [Kal60]. Although a monocular camera is proven to be sufficient for self-tracking [YKN06], a stereo camera is still preferred in systems that require more precise tracking results. The advantage of using a stereo camera is, that it is able to track feature points directly in the three-dimensional space [Bad04].

Figure 1.9: Example of 6D Vision. The left image shows a typical traffic scene. The right image draws the dense motion field of the surroundings estimated using 6D Vision. Velocities are encoded in color from green being slow to red being fast. Images are extracted from [RMWF10] and adapted by the author.

Also, computer vision plays an important role in tracking external objects. A vision-based method usually initializes a tracking procedure based on object detection, e.g., vehicle [CVT+12] or pedestrian [XLF05] detection. Common visual cues for object tracking include color, edges, optical flow, and texture [YJS06]. Recently, a new approach known as 6D Vision [FRBG05, RMWF10] enabled motion tracking of dense feature points in real-time using a stereo camera. The 6D Vision approach allowed a stereo camera to track the entire scene in its field of view (Fig. 1.9). In addition, the results of 6D Vision could be fused with other sensors available in a car such as Radar [GSDB07, BF12] or ultrasonics [KCV13], yielding more promising and robust tracking results.

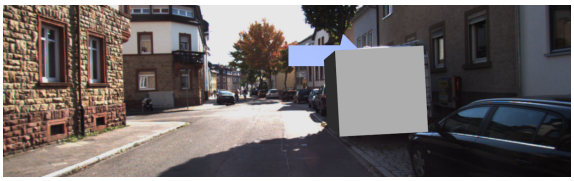
Rendering AR rendering aims at seamlessly integrating computer rendered objects into the real world. In order to achieve a high degree of visual coherence [KK12], global illuminations [PML+10] must be calculated accurately and the light interaction between virtual and real objects has to be simulated correctly. Adding shadowing, reflection, and refraction effects [HDH03, UIS+09, dLLT12] will further improve the visual realism of virtual objects. Image-based rendering techniques [KLZN03] that illuminate computer generated objects with measurements of real world lighting are especially helpful by various environment illuminations under different weather conditions.

Other AR functions such as spatial AR [BR05] and light field [MIT15] are less applicable in a car. For more details of AR functions, please refer to [ABB+01, ZDB08].

1.5.2 Fundamental Understandings

Depth Understanding Figure 1.10 shows an example of the use of depth understanding for AR navigation. Assume our navigation destination to be the building behind the caravan on the right side of the image. If we directly overlay a navigation arrow at the position where the building appears in the image, we will most probably get confused whether we should stop in front of the caravan or not (Fig. 1.10(a)). Now, if we replace the caravan with a dummy 3D cube and place it in front of the navigation arrow (Fig. 1.10(b)), the destination in the rendered image (Fig. 1.10(c)) becomes much more clear. This simple example shows how depth understanding enables realistic blending between virtual objects and the real world.

(a) AR overlay without depth reasoning.



(b) A dummy depth mask.

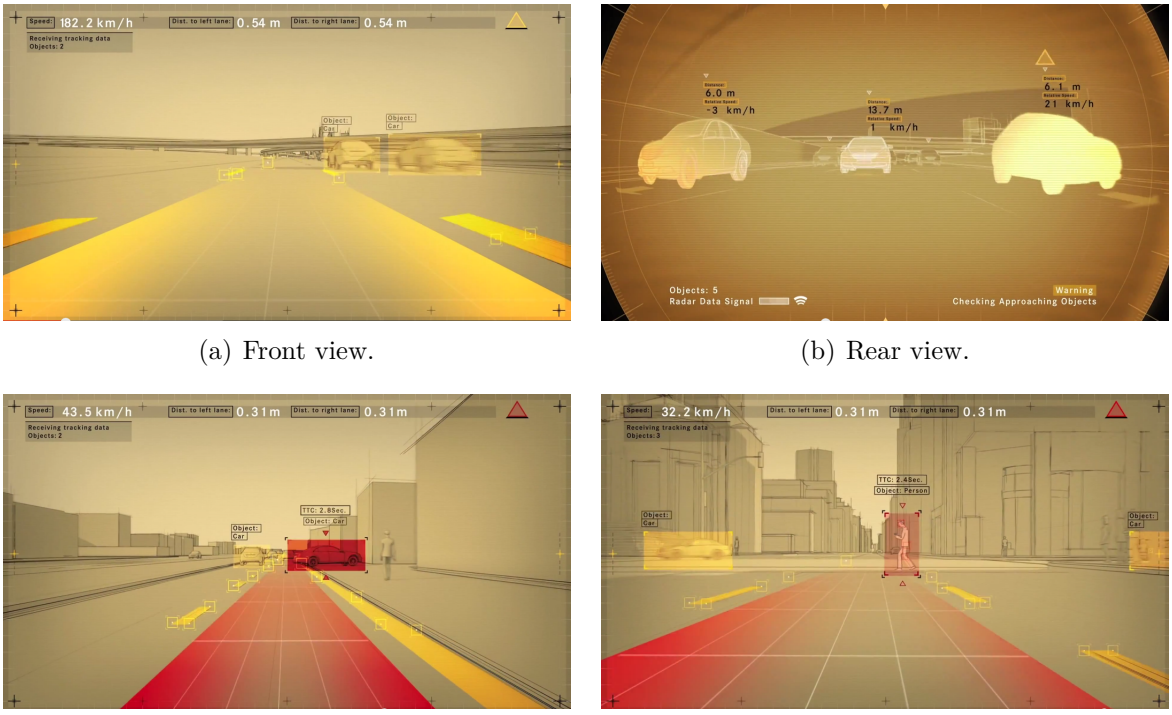
(c) AR overlay with depth reasoning.

Figure 1.10: Exploiting depth information for AR rendering.

Depth understanding supports AR functions by all means. We are able to localize the ego-car more precisely in the 3D surroundings using depth measurements, e.g., point clouds captured by a Lidar or disparity maps generated by a stereo camera. For tracking, depth understanding helps us separate target objects according to their distances to the ego-car. AR rendering becomes more realistic by applying a depth mask to virtual objects, as

already shown in Fig. 1.10. Hence, we need to understand the depth in order to augment the real world.

Scene Understanding Several conceptual graphical interfaces of AR driver assistance applications are shown in Fig 1.11. The speed of the ego-car, the speed of surrounding cars, and the distance to the left and right lane, are shown to the driver, in order to keep him or her informed about the surrounding traffic situation. If an obstacle appears in front of the ego-car, the obstacle and the road on which the ego-car drives turn red in order to give an early warning to the driver. In addition, an estimated Time To Collision (TTC) is augmented. In case someone attempts to overtake the ego-car, the overtaking vehicle is rendered differently in the rear view image, in order to attract the attention of the driver. All of these conceptual designs are based on the assumption that roads and lane markings are flawlessly detected, surroundings cars are robustly tracked, and pedestrians are recognized with their attentions being predicted in advance. In other words, the surrounding scene has to be fully understood for in-vehicle AR.



(a) Front view.

(b) Rear view.

(c) Crossing vehicle.

(d) Crossing pedestrian.

Figure 1.11: Conceptual graphical interfaces of AR driver assistance. Image source: Daimler AG.

Scene understanding is fundamental to AR functions. First, the accuracy of localization can be improved significantly through a combination of GPS and lane markings detection. While GPS receivers only provide a horizontal accuracy around 1.9 meters [U.S17], a typical lane marking aided localization technique operates at centimeter level [GBR14].

Second, by separating dynamic objects from a static background, tracking can be carried out more efficiently. There is no need to track objects that are irrelevant to traffic scenes, such as sky, buildings, or vegetations. The tracking algorithm would have a clear focus if these irrelevant objects could be detected and removed in a preprocessing step. Last but not least, understanding the entire scene provides abundant possibilities to redesign the AR graphical interface. Examples are already shown in Fig. 1.11(b). Here, classification is carried out at a pixel level, enabling the AR renderer to overlay a pixel-precise augmentation onto the overtaking car instead of a simple rectangular bounding box.

1.6 Organization of the Thesis

We present the roadmap of this thesis in Fig. 1.12. It illustrates how the knowledge is built up in order to reach the top of the pyramid – our goal to enable in-vehicle augmented reality in series production cars.

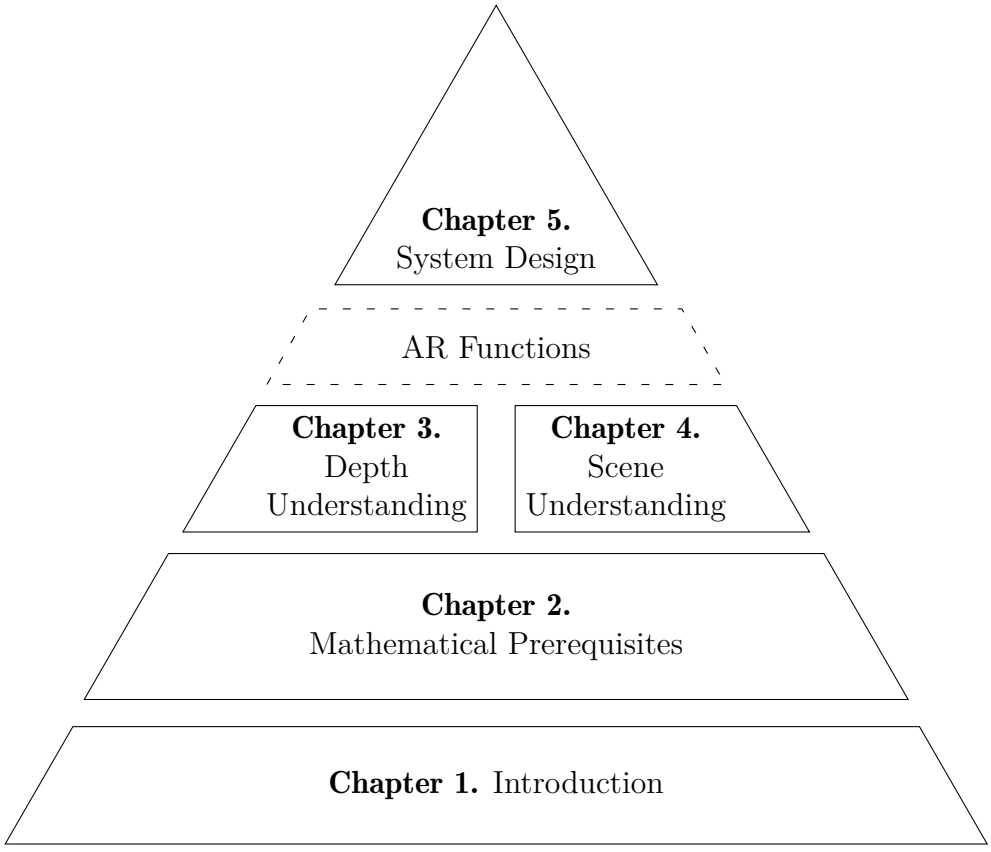


Figure 1.12: Roadmap of the thesis.

In Chapter 2, we introduce essential mathematical prerequisites which helps in better understanding the rest of this thesis from a mathematical point of view. These include

basic mathematical models of virtual and real cameras, a brief introduction to motions, and definitions of different coordinate systems relevant to in-vehicle AR.

In Chapter 3, we comprehensively discuss one of the fundamental understandings for in-vehicle augmented reality: *Depth Understanding*. The major challenge in enabling depth understanding in a car lies above all in, as already mentioned in Section 1.4.1, the bandwidth of the in-vehicle communication system. We propose a self-developed lossless compression scheme for Stixel in this chapter to solve this problem. We believe that the proposed compression scheme is a key enabler for in-vehicle augmented reality.

In Chapter 4, we provide an in-depth discussion of another fundamental understandings for in-vehicle AR: *Scene Understanding*. As introduced in Section 1.4.2, the main challenge here is how to reconstruct the three-dimensional surroundings of the ego-car at an object level. We present a novel 3D reconstruction workflow named *3D Shaping* in this chapter, that is able to reconstruct the 3D shape of an object using only a single frame from a monocular camera. In addition, we propose two extensions to 3D Shaping in order to make it more practical for real traffic scenes. Our proposed 3D Shaping workflow together with its extensions serve as the first step toward object-level 3D reconstruction, which solves another key problem for incorporating augmented reality in a car.

Based on the fundamental understandings discussed in Chapter 3 and Chapter 4, a wide range of AR functions can be realized. However, as already elaborated at the beginning of Section 1.5, addressing each AR function in the finest detail is out of scope of this thesis. Therefore, instead of dedicating an entire chapter specifically for AR functions, we implicitly immerse the discussion of the three most important functions into Chapter 5 and explain how they are supported and improved through depth understanding and scene understanding. These functions include *road Surface Estimation*, *ego-Motion Estimation*, and *depth Culling*, each as a representative of localization, tracking, and rendering, respectively.

In Chapter 5, we tackle the final key problem of this thesis – the proof of concept of AR in a test vehicle. Here, we put the emphasis on how our new findings in depth and scene understanding would enable augmented reality in a close-to-production vehicle platform. We present three different in-vehicle AR systems, respectively designed for the current generation, the next generation, and future generation of productions cars. The current generation AR system is integrated into a middle-class prototype vehicle, whereas the next generation into a modified upper-class production car with more advanced sensor technology. Both of them are demonstrated in normal road traffic. We show the advantage of using Stixel compression for the transmission of depth information in the next generation AR system, and we design the future AR system in such a way as to fully exploit 3D Shaping for object-level 3D reconstruction. We expect our designed AR systems to be seriously considered by the industrial decision makers and eventually adopted in series production cars in the future.

In the last chapter of this thesis, Chapter 6, we conclude this thesis and give a brief outlook on the future of in-vehicle AR.

1.7 Key Contributions

The key contribution of this thesis, in a nutshell, is that *it brings in-vehicle augmented reality one step closer to practical implementation.* As already stated in Section 1.2, this thesis is, to our best knowledge, the first of its kind which jointly concerns fundamental understandings and in-vehicle E/E architecture design in order to enable augmented reality in series production cars. We propose solutions in the thesis which resolves two technical key issues for in-vehicle AR, namely transmission of depth information and object-level 3D reconstruction, respectively. In addition, we tackle the key engineering problem to prove the concept of in-vehicle AR by presenting three designed AR systems for the current generation, the next generation, and future generation of production cars, based on our findings during the research on depth understanding and scene understanding.

From a technical point of view, the contributions of this thesis are three-fold.

In Chapter 3, we develop an efficient lossless compression scheme for Stixel which enables the transmission of depth information through a reasonably priced in-vehicle communication system. The proposed compression scheme is, to our best understanding, the first of its kind in the context of depth compression. We design the compression algorithm in such a way that both spatial and temporal redundancies in Stixel are removed. The removal of redundancies is achieved by using a combination of predictive modeling and entropy coding, which is inspired by the prediction step of image compression. Compared to general purpose compression algorithms, our approach takes advantage of continuous motion of objects in a common traffic scene for predictive modeling and therefore outperforms them significantly. Evaluation shows that our proposed algorithm outperforms a well-known general purpose compression method, namely *zlib*, by more than 60 percent in space savings. More importantly, we prove that using our proposed Stixel compression scheme, depth data could be transmitted through a reasonably priced CAN bus, whereas a more expensive FlexRay bus is needed for uncompressed Stixel data or compressed data through general compression methods. This finding has great relevance for the cost-sensitive automotive industry to incorporate augmented reality in a car.

In Chapter 4, we present an approach named 3D Shaping to reconstruct 3D objects using only a single frame from a monocular camera. The proposed approach exploits an existing silhouette-based reconstruction technique, which takes advantage of an extremely low-dimensional latent space to represent 3D geometries. We combine the latent geometry representation with a deep-learning-based appearance detection step, which achieves nearly 20 percent performance gain by 3D reconstruction in viewpoint accuracy. Furthermore, we present two extensions to 3D Shaping in order to make it more practical for real traffic scenes. The proposed extensions take advantage of additional 3D sensors in a car including stereo camera and Lidar. In the first extension, we propose a novel neural network architecture called Pose-RCNN, which is able to jointly detect objects and estimate a viewpoint angle for each detected object. We attach a small viewpoint regression network on top of the ROI pooling layer of a Region-based Convolutional Neural Network (R-CNN), yielding the architecture of the proposed Pose-RCNN. We show competitive

results of our proposed Pose-RCNN against other state-of-the-art approaches. In the second extension, we use an additional point cloud energy for 3D Shaping, which enables us to optimize 3D geometries directly in the 3D space. By using 3D measurements from a Lidar, we are able to reconstruct multiple objects within the same class by heavy self-occlusion. In summary, the proposed 3D Shaping approach and its extensions serve as the first step towards object-level 3D reconstruction of the surroundings, based on which a number of AR functions can be further improved.

In Chapter 5, we present three different in-vehicle AR systems, respectively designed for the current generation, the next generation, and future generation of productions cars, in order to prove the concept of in-vehicle augmented reality. First, we present a rather modest design of an in-vehicle AR system aiming at only using currently available technologies in a middle-class commercially available car with least possible modifications. Under these constraints, we decide that the AR system shall be purely GPS-based. We propose a low-cost GPS-smoothing algorithm, which is able to track the ego-car to some extent but requires much less computational resources compared to an Extended Kalman Filter or visual tracking algorithms. In addition, we use a "clever" HMI design and a sophisticated data retrieval mechanism to visually compensate the drifting effect in case the tracking is unstable. We demonstrate the first AR system with two customer features, namely AR driver assistance and AR passenger infotainment. Second, based on the feedback on our first AR prototype system, we present the next generation of in-vehicle AR system which takes advantage of the most advanced automotive sensors back then in a modified upper-class production car. We adjust our design requirements according to the additionally available resources in the test vehicle and decide that the AR system shall be distributed, in order to simulate different vehicle domains. Stixel compression introduced in Chapter 3 is applied here in order to simulate the transmission of depth information from the ADAS domain to the telematics domain in the in-vehicle E/E architecture. With depth information available, we fully exploit depth understanding in order to support AR functions including road estimation, ego-motion estimation, and depth culling in the next generation AR system. We demonstrate these AR functions in the test vehicle in normal daily traffic. Last but not least, we propose a future generation AR system aiming at future series production. We present the functional modules, software components, and the E/E architecture of our designed future AR system in detail. Here, 3D Shaping introduced in Chapter 4 is integrated, which enables 3D reconstruction of the surroundings at an object level. The performance of road estimation, ego-motion estimation, and depth culling can be boosted using 3D Shaping. We believe that our designed AR systems will bring in-vehicle augmented reality one-step closer to everybody's daily life.

Note that parts of this work have already been published in [RGHC14a, RGHC14b, RTG+14, BRWF16, RKD16a, RKD16b, RC19a], and submitted to [RC19b], respectively.

The main contribution presented in Chapter 3 is published in

- Q. Rao, C. Grünler, M. Hammori, S. Chakraborty: *Stixel on the Bus: An Efficient Lossless Compression Scheme for Depth Information in Traffic Scenarios*. In Proceedings of the 20th Anniversary International Conference on MultiMedia Modeling (MMM), 2014.

and

- Q. Rao, S. Chakraborty: *Efficient Lossless Compression for Depth Information in Traffic Scenarios*. Springer Multimedia Systems, Feb. 2019.

The main contribution presented in Chapter 4 appeared in the following publications

- Q. Rao, L. Krüger, K. Dietmayer: *Monocular 3D Shape Reconstruction Using Deep Neural Networks*. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), 2016.
- M. Braun, Q. Rao, Y.K. Wang, F. Flohr: *Joint Object Detection and Pose Estimation Using 3D Object Proposals*. In Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016.
- Q. Rao, L. Krüger, K. Dietmayer: *3D Shape Reconstruction in Traffic Scenarios Using Monocular Camera and Lidar*. In Workshop Proceedings of the 13th Asian Conference on Computer Vision (ACCV), 2016.

and is submitted as the following manuscript for review.

- Q. Rao, S. Chakraborty: *In-Vehicle Object-level 3D Reconstruction of Traffic Scenes*. Submitted.

The main contribution presented in Chapter 5 appeared in the following publications.

- Q. Rao, T. Tropper, C. Grünler, M. Hammori, S. Chakraborty: *AR-IVI - Implementation of In-Vehicle Augmented Reality*. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2014.
- Q. Rao, C. Grünler, M. Hammori, S. Chakraborty: *Design Methods for Augmented Reality In-Vehicle Infotainment Systems*. In Proceedings of the 51st Annual Design Automation Conference (DAC), 2014.

2 Mathematical Prerequisites

There is no royal road to geometry.

Euclid
c. 325 BC - c. 265 BC

We understand the world through geometry, both the virtual world and the real. It is important to have a deeper mathematical insight into both worlds in order to better understand the problem of augmented reality. We provide the insight in this chapter.

We first introduce mathematical models of different type of cameras including virtual camera, pinhole camera, and stereo camera in Section 2.1. In Section 2.2, we briefly introduce a simple type of motion that models the motion of rigid bodies, which applies for most objects in a common traffic scene. In Section 2.3, we introduce the coordinate systems used by in-vehicle augmented reality. In the last section, Section 2.4, we explain the major challenges of AR from a mathematical perspective.

2.1 Camera Model

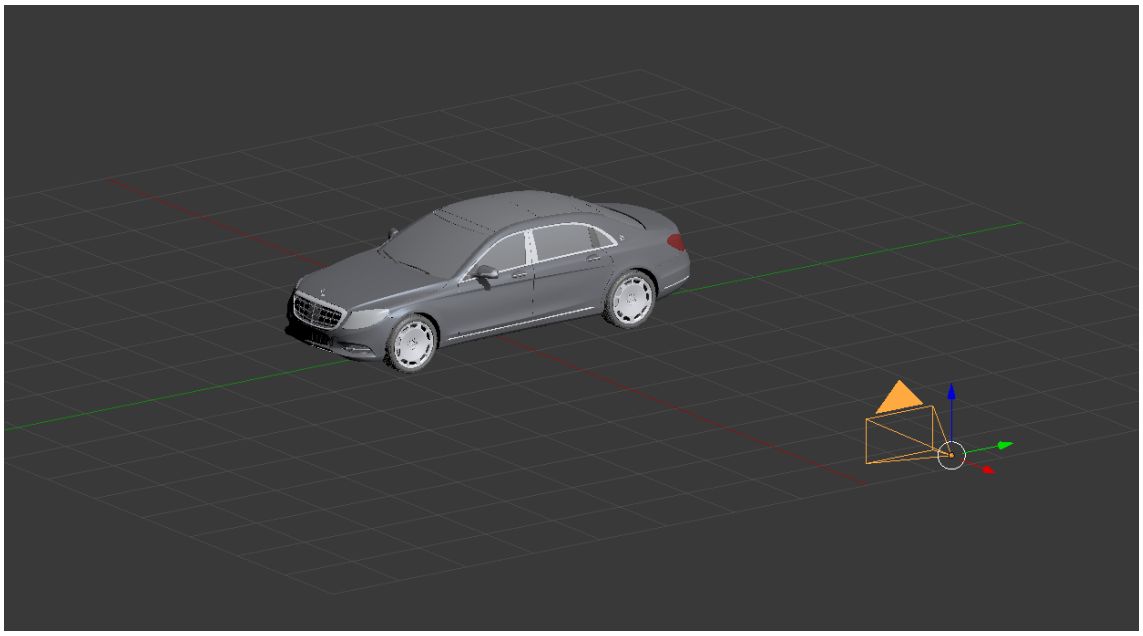
2.1.1 Virtual Camera

In the virtual world, a virtual camera renders a 3D scene onto an image. In mathematical terms, it first transforms a point from the camera coordinate system into the Normalized Device Coordinate (NDC) cube, where coordinates are normalized within $[-1, 1]$ (Eq. 2.1). Then, the NDC coordinates are rasterized according to the size of the image, yielding pixel coordinates. The matrix P that transforms a camera point ${}^{c}\mathbf{X}$ to an NDC point ${}^{ndc}\mathbf{X}$ is referred to as *projection matrix*. The parameters of the projection matrix influence the type of the projection. Common projection types include orthogonal projection and perspective projection, whose projection matrices are given in Eq. 2.2 and Eq. 2.3, respectively. Their view frustums are illustrated in Fig. 2.2, and rendered examples of both projections are shown in Fig. 2.1. Readers might note the tiny difference at the grill of the car in Fig. 2.1

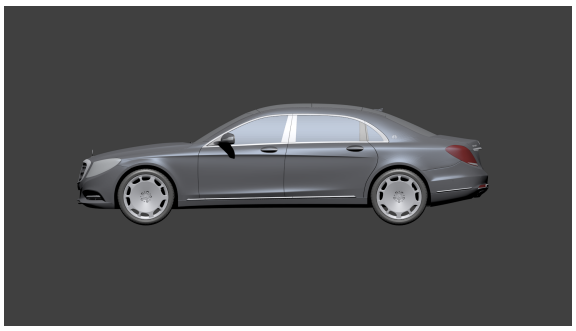
$$ {}^{ndc}\mathbf{X} = P\,{}^{c}\mathbf{X} \tag{2.1} $$

$$P_o = \begin{bmatrix} \frac{1}{w} & 0 & 0 & 0 \\ 0 & \frac{1}{h} & 0 & 0 \\ 0 & 0 & -\frac{2}{Z_f - Z_n} & -\frac{Z_f + Z_n}{Z_f - Z_n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

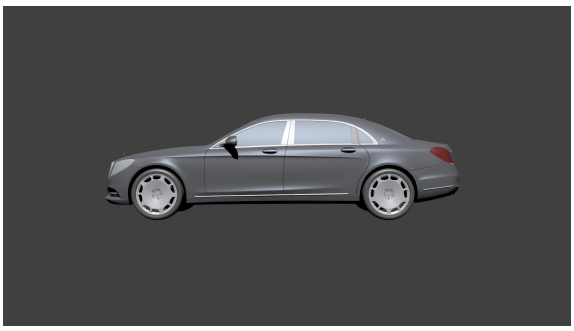
$$P_p = \begin{bmatrix} \arctan(\frac{\alpha_x}{2}) & 0 & 0 & 0 \\ 0 & \arctan(\frac{\alpha_y}{2}) & 0 & 0 \\ 0 & 0 & -\frac{Z_f + Z_n}{Z_f - Z_n} & -\frac{2Z_f Z_n}{Z_f - Z_n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{2.3}$$



(a) 3D scene.

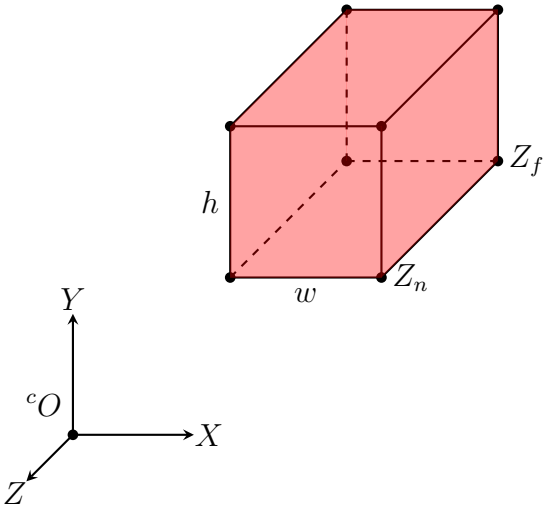


(b) Orthogonal projection.

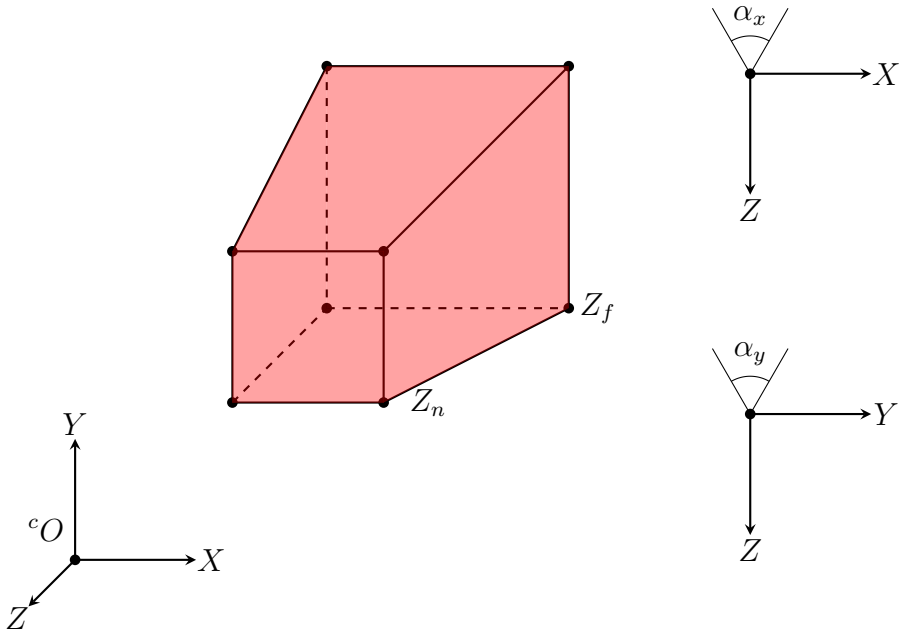


(c) Perspective projection.

Figure 2.1: Orthogonal vs. perspective projection. A part of the grill is only visible to the orthogonal camera.



(a) Orthogonal camera. w and h denote the sensor width and height. Z_n and Z_f are the nearest and farthest clipping distance.



(b) Perspective camera. α_x and α_y denote the horizontal and vertical Field Of View (FOV) angle. Z_n and Z_f are the nearest and farthest clipping distance.

Figure 2.2: View frustum of a virtual camera.

2.1.2 Pinhole Camera

A pinhole camera (Fig. 2.3) is one of the simplest mathematical models used to model a camera in the real world. It assumes that the thickness of the lens is infinitely close to zero so that all rays remain undeflected after passing through it.

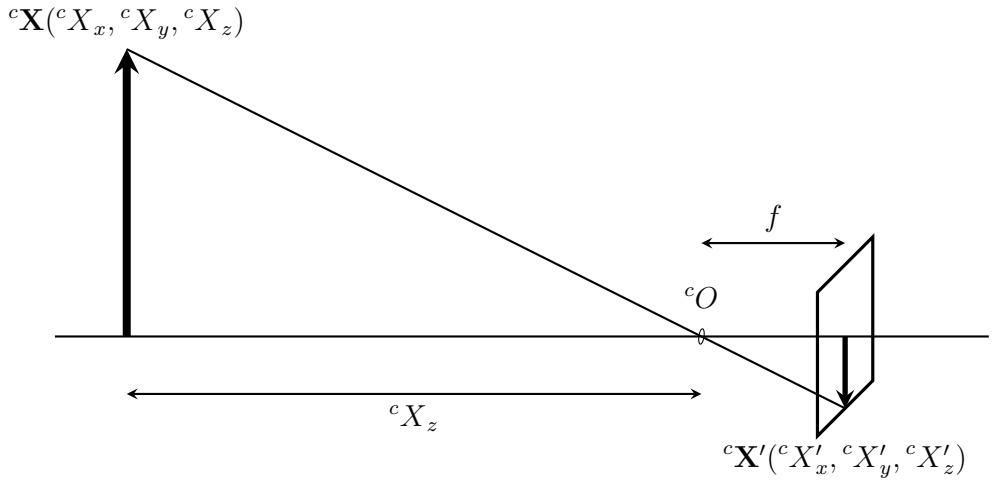


Figure 2.3: Pinhole camera model.

Similar to a virtual perspective camera, a pinhole camera projects a point in the camera coordinate system onto an image through perspective projection. First, the camera coordinates are projected through Eq. 2.4, where f denotes the focal length of the camera. ${}^cX'_x$ and ${}^cX'_y$ are normalized metric coordinates on the image plane. They are then transformed into pixel coordinates through Eq. 2.5, where (s_x, s_y) denote the scale factors and (c_x, c_y) the image center.

$$\begin{aligned} {}^cX'_x &= f\frac{{}^cX_x}{{}^cX_z} \\ {}^cX'_y &= f\frac{{}^cX_y}{{}^cX_z} \end{aligned} \tag{2.4}$$

$$\begin{aligned} {}^{im}x_x &= s_x\,{}^cX_x + c_x \\ {}^{im}x_y &= s_y\,{}^cX_y + c_y \end{aligned} \tag{2.5}$$

Equation 2.6 expresses the complete projection equation in matrix form. s_θ is an additional parameter, which is introduced to model the skew factor of the Charge Couple Device (CCD) sensor in a camera if it is not perfectly rectangular. The depth ${}^cX_z$ is lost through the projection.

$$\begin{bmatrix} {}^{im}x_x \\ {}^{im}x_y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fs_x & fs_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} {}^cX_x \\ {}^cX_y \\ {}^cX_z \\ 1 \end{bmatrix} \tag{2.6}$$

Equation 2.7 writes a more compact form of the projection equation. The matrix K containing projection parameters is referred to as camera *intrinsic parameters*.

$$ {}^{im}\mathbf{x} \sim K \left[\; I \; \middle| \; \mathbf{0} \; \right] {}^{c}\mathbf{X} \tag{2.7} $$

2.1.3 Stereo Camera

As the name suggests, a stereo camera comprises two (horizontally) aligned pinhole cameras whose fields of view overlap. Every point in the overlapped FOV is seen twice by the stereo camera: once by the left camera and once by the right, as depicted in Fig. 2.4. The difference between its horizontal coordinate on the left and on the right image is referred to as *disparity*, which is a common measure of depth information in the context of stereo vision.

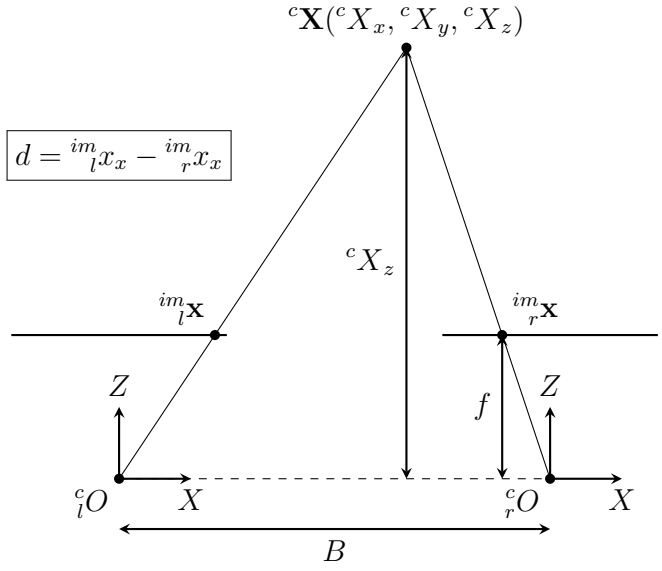


Figure 2.4: Stereo camera model.

Given the metric baseline of the stereo camera B and the focal length f in pixels, the relation between the disparity d and the depth ${}^{c}X_z$ of a point is formulated in Eq. 2.8.

$$ {}^{c}X_z = f\frac{B}{d} \tag{2.8} $$

After the depth ${}^{c}X_z$ is reconstructed, the other two coordinates ${}^{c}X_x$ and ${}^{c}X_y$ of the point ${}^{c}\mathbf{X}$ can be determined through Eq. 2.9, where (c_x, c_y) indicates the image center. Therefore, we can use a stereo camera to reconstruct the 3D point cloud of the scene.

$$ \begin{aligned} {}^{c}X_x &= ({}^{im}x_x - c_x)\frac{{}^{c}X_z}{f} \\ {}^{c}X_y &= ({}^{im}x_y - c_y)\frac{{}^{c}X_z}{f} \end{aligned} \tag{2.9} $$

2.2 Motion

2.2.1 Model View Matrix

In the virtual world, objects have their own coordinate systems where their geometries (3D shapes) are defined, commonly using vertices and faces. A point (vertex) in an object coordinate system is transformed to a camera coordinate system through a model transformation and a view transformation, as expressed in Eq. 2.10.

$$^{c}\mathbf{X} = V \cdot M \cdot {}^{o}\mathbf{X} \tag{2.10}$$

The model matrix M first transforms object coordinates into world coordinates through a concatenation of scaling, rotation, and translation, as formulated in Eq. 2.11. Then, the world coordinates are mapped to camera coordinates through the view matrix V, which is defined using the right, up, and forward vector of the camera, as well as the position of the camera in the world, as expressed in Eq. 2.12.

$$M = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & 0 \\ r_{10} & r_{11} & r_{12} & 0 \\ r_{20} & r_{21} & r_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

$$V = \begin{bmatrix} right_x & up_x & forward_x & position_x \\ right_y & up_y & forward_y & position_y \\ right_z & up_z & forward_z & position_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

The production of the model matrix and the view matrix is often referred to as modelview matrix as a one-step transformation, if the world coordinates are not interested. By combining Eq. 2.12 and Eq. 2.1, we have the complete transformation from an object point to an NDC point, as expressed in Eq. 2.13.

$$^{ndc}\mathbf{X} = P \cdot V \cdot M \cdot {}^{o}\mathbf{X} \tag{2.13}$$

2.2.2 Rigid Body Motion

The motion of solid and rigid objects in the real world such as cameras or cars can be modeled through *rigid body motion*, where object deformation is neglected. In other words, the distance between any two points on a rigid body remains constant as it moves in the world. Similar to the modelview matrix in the virtual world, a matrix M describing a rigid body motion in the real world can be decomposed into a rotation matrix R and a translation vector $\mathbf{T}$, as expressed in Eq. 2.14.

$$M = \begin{bmatrix} R & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

If we attach the world coordinate system to the camera coordinate system at an initial time point and track the motion of the camera (Fig. 2.5) throughout the time, we can build the transformation from a world point ${}^{w}\mathbf{X}$ to a camera point ${}^{c}\mathbf{X}$, as expressed in Eq. 2.15. ${}^{c}_{w}M$ denotes the rigid body motion from the camera to the world, which is often referred to as the camera *extrinsic parameters*.

$$ {}^{c}\mathbf{X} = {}^{c}_{w}M \cdot {}^{w}\mathbf{X} \tag{2.15} $$

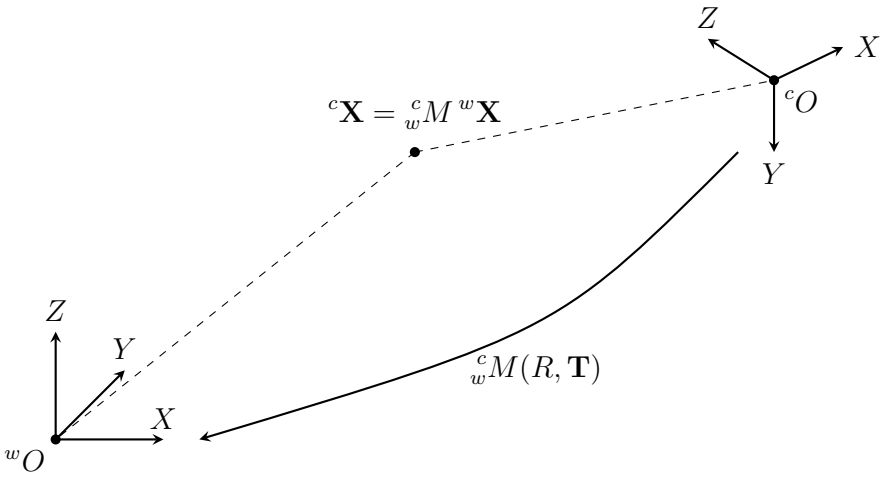


Figure 2.5: Transformation from camera to world coordinate system.

By combining the intrinsic parameters in Eq. 2.7 and the extrinsic parameters in Eq. 2.15, we can model the complete projection in the real world using only one projection matrix P, as expressed in Eq. 2.16.

$$ {}^{im}\mathbf{x} \sim P \cdot {}^{w}\mathbf{X} = K \left[\; I \;\middle|\; \mathbf{0} \; \right] {}^{c}_{w}M \, {}^{w}\mathbf{X} \tag{2.16} $$

Representation of Rotations

There are different mathematical models for representing 3D rotations, two of which are introduced here, namely quaternion and Euler angles. Both of them are widely used in context of computer vision or robotics for modeling 3D rotation.

Quaternion A quaternion is an extension of a complex number, which takes the form in Eq. 2.17, where $i^2 = -1$, $j^2 = -1$, and $i \cdot j = -j \cdot i$.

$$ q = q_0 + q_1 i + q_2 j + q_3 ij, \quad q_0, q_1, q_2, q_3 \in \mathbb{R} \tag{2.17} $$

A rotation can be equally mapped to a special sub-group of the set of quaternion $\mathbb{H}$, i.e., the group of unit quaternions $\mathbb{S}^3$, as defined in Eq. 2.18.

$$ \mathbb{S}^3 = \{ q \in \mathbb{H} \mid \|q\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \} \tag{2.18} $$

The conversion from quaternion to rotation matrix and vice versa are given in Eq. 2.19 and Eq. 2.20, respectively.

$$R = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & 1-2q_1^2-2q_2^2 \end{bmatrix} \tag{2.19}$$

$$\begin{aligned} q_0 &= \sqrt{\frac{1+r_{00}+r_{11}+r_{22}}{2}} \\ q_1 &= \frac{r_{21}-r_{12}}{4q_0} \\ q_2 &= \frac{r_{02}-r_{20}}{4q_0} \\ q_3 &= \frac{r_{10}-r_{01}}{4q_0} \end{aligned} \tag{2.20}$$

Euler angles A three-dimensional rotation can be decomposed to a concatenation of three rotations around each axis, as expressed in Eq. 2.21. α, β, and γ are the rotation angles around the x-, y-, and z-axis, which are referred to as roll, pitch, and yaw, respectively. Together, they are called *Euler angles*.

$$\begin{aligned} R_{zyx}(\gamma,\beta,\alpha) &= R_z(\gamma)R_y(\beta)R_x(\alpha) \\ R_z(\gamma) &= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ R_y(\beta) &= \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \\ R_x(\alpha) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \end{aligned} \tag{2.21}$$

We use Eq. 2.22 to calculate Euler angles from a rotation matrix.

$$\begin{aligned} \tan\alpha &= \frac{r_{21}}{r_{22}} \\ \tan\beta &= \frac{-r_{20}}{r_{21}\sin\alpha + r_{22}\cos\alpha} \\ \tan\gamma &= \frac{r_{10}}{r_{00}} \end{aligned} \tag{2.22}$$

In case of a *gimbal lock*, i.e., when r_{00}, r_{10}, r_{21} and r_{22} are zero, the yaw angle γ is set to be zero and the roll angle α is given by $\alpha = \arctan(r_{01}/r_{11})$.

For more details of rotation matrix, quaternion, and Euler angles, please refer to [Sho85].

2.3 Coordinate Systems

Typically, we use multiple coordinate systems to describe the motion of a complex system such as a car, as shown in Fig. 2.6. Each coordinate system is responsible for describing the motion of a specific component of the car, such as wheels, chairs, and the chassis. In the context of in-vehicle augmented reality, we are specifically interested in the vehicle (ego-car) coordinate system and the transformations from different sensor coordinate systems to it.

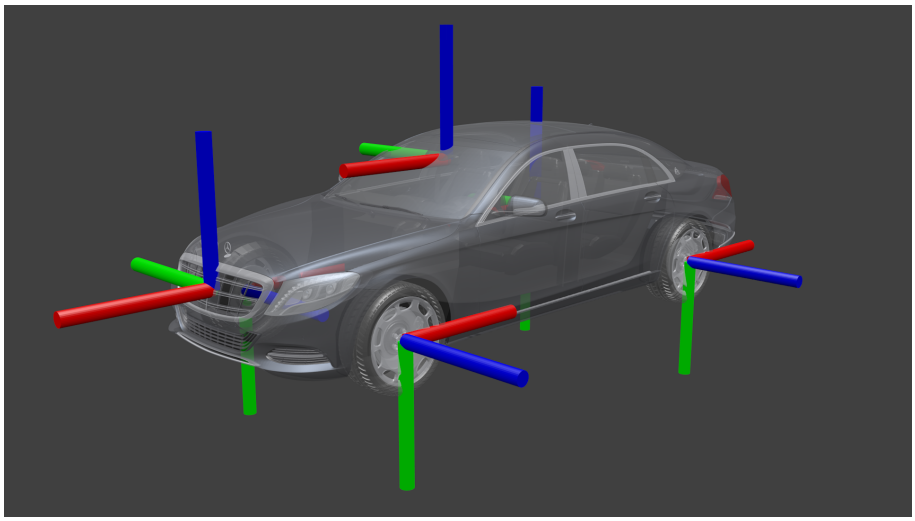


Figure 2.6: Example coordinate systems in a car.

Figure 2.7 illustrates typical coordinate system transformations for in-vehicle augmented reality. Real world objects in the surroundings are tracked by different sensors in a car, i.e., their motions are first estimated in the corresponding sensor coordinate systems, such as a camera, Radar, or Lidar coordinate system. Since sensors equipped in a car are moving with the car, it is more practical to define a car coordinate system to describe the motion of the ego-car in the world instead of directly tracking the motion of each sensor. The estimation of ego-motion can be carried out through wheel odometry, visual odometry, GPS tracking, or combinations of them.

The car coordinate system is attached to the middle of the rear axle, with the x-axis pointing to the driving direction, y-axis pointing to the left looking from the driver seat, and z-axis pointing up. Assuming the car drives at a constant yaw rate ω and speed v in a short time interval Δt, the motion of the car can be described by Eq. 2.23, where r, l, and ϕ denotes the radius, the length, and the angle of the arc driven by the car (Fig. 2.8) in the short period of time. If the yaw rate is positive, the car drives to the left.

$$\begin{aligned} \phi &= \omega \Delta t \\ l &= v \Delta t \\ r &= \frac{l}{\phi} = \frac{v}{\omega} \end{aligned} \tag{2.23}$$

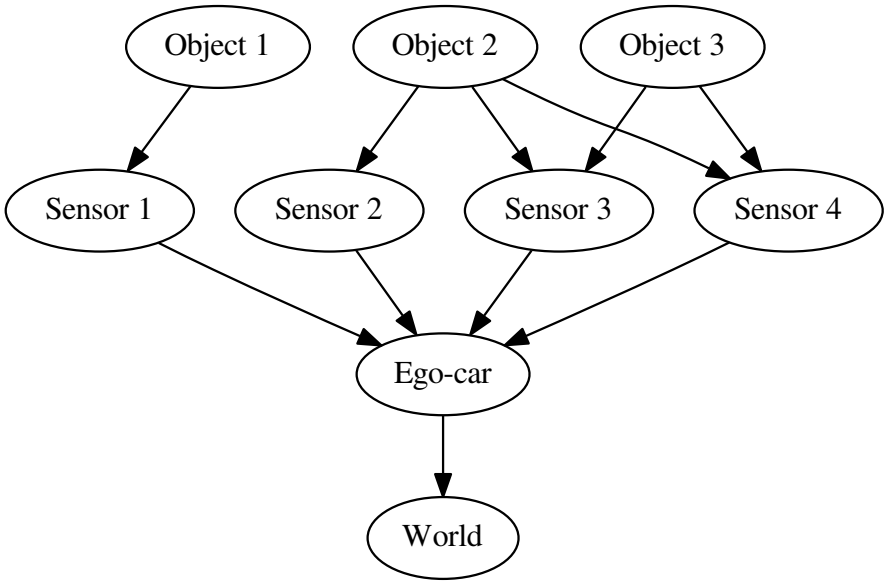


Figure 2.7: Typical coordinate system transformations used by in-vehicle AR.

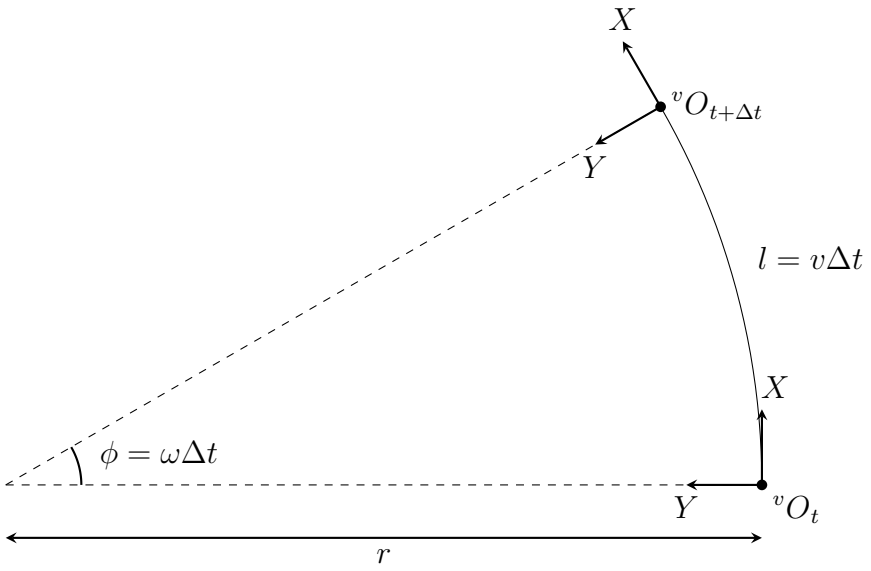


Figure 2.8: Movement of the vehicle coordinate system.

The translation vector $\mathbf{T}$ and the rotation matrix R of the car during the time interval are given by Eq. 2.24 and Eq. 2.25. Note that they are defined in the car coordinate system *before* the movement.

$$\mathbf{T} = \begin{bmatrix} r\sin\phi \\ r(1-\cos\phi) \\ 0 \end{bmatrix} \tag{2.24}$$

$$R = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

Equation 2.26 describes the inverse transformation of a fixed point in the world with respect to the movement of the car.

$$\begin{aligned} {}^{v}\mathbf{X}_{t+\Delta t} &= R_{t+\Delta t|t}({}^{v}\mathbf{X}_t - T_{t+\Delta t|t}) \\ &= R_{t+\Delta t|t}\,{}^{v}\mathbf{X}_t - R_{t+\Delta t|t}T_{t+\Delta t|t} \\ &= \left[\begin{array}{ccc|c} \cos\phi & \sin\phi & 0 & -r\sin\phi \\ -\sin\phi & \cos\phi & 0 & r(1-\cos\phi) \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] {}^{v}\mathbf{X}_t \\ &= {}^{v}M_{t+\Delta t|t}\,{}^{v}\mathbf{X}_t \end{aligned} \tag{2.26}$$

The motion matrix ${}^{v}M_{t+\Delta t|t}$ transforms a point from the previous car coordinate system into the current. Assuming a camera is perfectly calibrated to the car and the extrinsic matrix ${}^{c}_{v}M$ is known, we can derive the transformation of the point in the camera coordinate system using Eq. 2.27. Transformations in the camera coordinate system are more interested in the context of AR since virtual objects are eventually augmented in the virtual camera instead of the car coordinate system.

$$\begin{aligned} {}^{c}\mathbf{X}_{t+\Delta t} &= {}^{c}_{v}M\,{}^{v}\mathbf{X}_{t+\Delta t} \\ &= {}^{c}_{v}M\,{}^{v}M_{t+\Delta t|t}\,{}^{v}\mathbf{X}_t \\ &= {}^{c}_{v}M\,{}^{v}M_{t+\Delta t|t}\,{}^{c}_{v}M^{-1}\,{}^{c}\mathbf{X}_t \end{aligned} \tag{2.27}$$

The result matrix ${}^{c}M_{t+\Delta t|t}$ is written in Eq. 2.28.

$${}^{c}M_{t+\Delta t|t} = {}^{c}_{v}M\,{}^{v}M_{t+\Delta t|t}\,{}^{c}_{v}M^{-1} \tag{2.28}$$

Camera to car calibration, as well as Radar or Lidar to car calibration, can be carried out through hand-eye calibration techniques. For more details, please refer to [SH06].

2.4 Augmented Reality

We illustrate the procedure of merging the *Virtual* and the *Real* using a V-shape graph, as shown in Fig. 2.9. In the real world, objects are detected and tracked, i.e., the rigid body

motion (Section 2.2.2) of each surrounding object and the ego-car is estimated, based on which a three-dimensional model of the environment is reconstructed. In the virtual world, each virtual object to be augmented is modeled by its geometry, which goes through the model-view-projection transformation (Section 2.2.1) and is eventually rendered on a 2D canvas. Finally, the rendered image and the real image are superimposed, yielding an augmented reality blend.

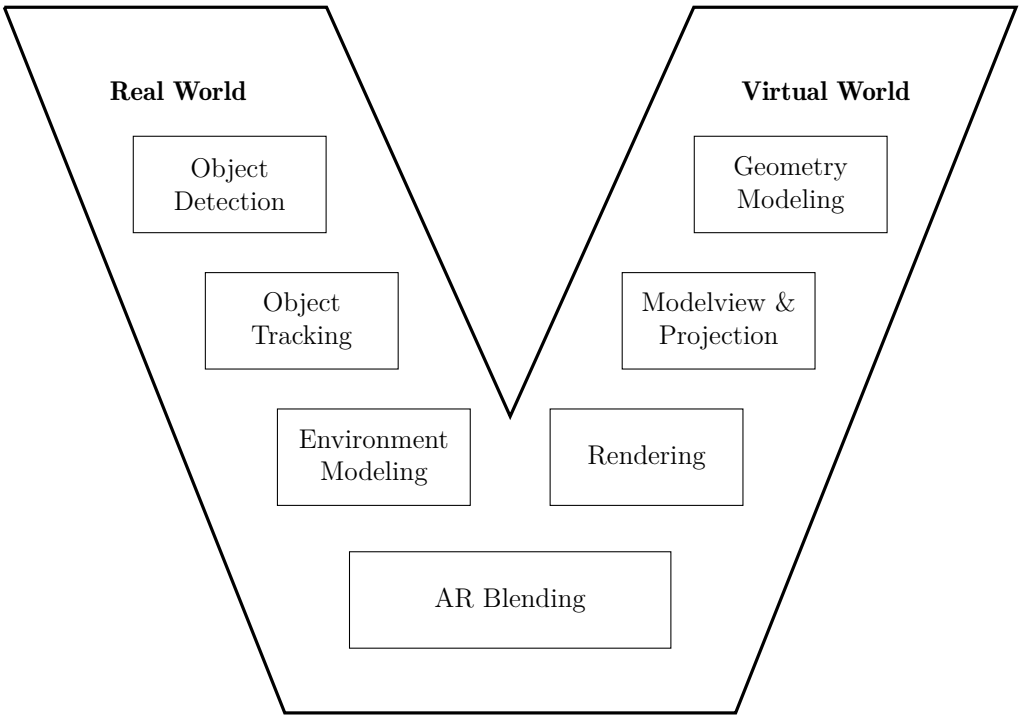


Figure 2.9: V-model of augmented reality.

The quality of AR blending is mainly affected by the following three factors. First, intrinsic calibration, i.e., the process of determining the camera intrinsic parameters of a pinhole camera in the real world. The better a real-world pinhole camera is calibrated, the more realistic a virtual camera can be parameterized in order to project objects in the virtual world in the same way as in the real. Second, tracking, i.e., the procedure of estimating of motion matrices of the ego-car and surrounding objects. In case we need to attach a virtual object on a moving object in the real world, the modelview matrix of the virtual object has to be updated corresponding to the motion of the real-world object. It is therefore critical for AR to estimate the motion matrix of a moving real-world object as precise as possible. Third, 3D environmental model, based on which depth or occlusion reasoning is performed. This has a direct influence on the final output of AR blending, as already shown in the example in Fig. 1.10. Without a properly reconstructed 3D environmental model, the output is only a superposition of real and rendered images which would most of the time only distract the driver instead of adding values to in-vehicle augmented reality.

3 Depth Understanding

In the land of the blind, the one-eyed man is king.

Desiderius Erasmus Roterodamus
1466 - 1536

The Dutch Renaissance humanist Erasmus [Des23] revealed the importance of the ability to perceive the world with eyes, even with only one. Here, we extend his famous proverb quoted at the beginning of this chapter by adding another eye. It becomes: *In the land of one-eyed men, the binocular man is king.* With two eyes, we are able to perceive the three-dimensional world. Our skill to reason the depth from different views seems so natural that we simply take it for granted.

Today, as a monocular forward looking camera already became a standard equipment in a car, the automotive industry gradually start to deploy stereo camera for more advanced driver assistance functions. A car equipped with a stereo camera is enabled to fully understand the depth. Consequently, new research questions such as how to represent, transmit, and reprocess the depth information retrieved by an in-vehicle stereo camera are raised. The answers still remain open.

In this chapter, we focus on a self-developed algorithm for compressing *Stixels*, a mid-level representation of depth information. We begin with an introduction of depth understanding and related work in Section 3.1 and Section 3.2, respectively. In Section 3.3, we discuss the redundancy in Stixel data from a mathematical point of view. In Section 3.4, we explain the workflow of our proposed compression algorithm and present experimental results. We summarize this chapter in Section 3.5 and outline directions for future work.

3.1 Introduction

Vision-based advanced driver assistance systems are increasingly being used in modern production cars. Towards this, stereo vision systems are used to capture the 3D surroundings of the ego-cars. Using stereo matching algorithms [Hir05, GEM09], depth information is generated, processed, or stored for a short period of time in the vision processing system for further use, such as building an environmental model for in-vehicle augmented reality.

Stixel [BFP09, PF10, PF11, PGS13] is a data structure that was introduced about ten years ago to model depth information in traffic scenes. The term “Stixel” was created by

combining “Stick” and “Pixel”, which reflects the fact that a Stixel frames a thin rectangular area (like a stick) in a depth image with similar disparity (pixel) values (Fig. 3.1). Based on the assumption that the geometry in common traffic scenes comprises predominantly vertical and horizontal surfaces, the Stixel representation finds a balance between the compactness and informativeness of the depth information.

Even using the Stixel representation, the depth data volume of the complete scene is still too huge for a low- or medium-bandwidth in-vehicle communication system, e.g., a CAN [Int03] bus. In other words, in today’s commercially available cars, depth information still remains in the in-vehicle vision system, without being accessible to the other components in the car due to cost reasons. This limits the development of a number of applications such as augmented reality that requires depth information of the surrounding environment. Hence, the automotive industry is still seeking for a solution that makes depth information available to other in-vehicle processing units via a less expensive communication bus.

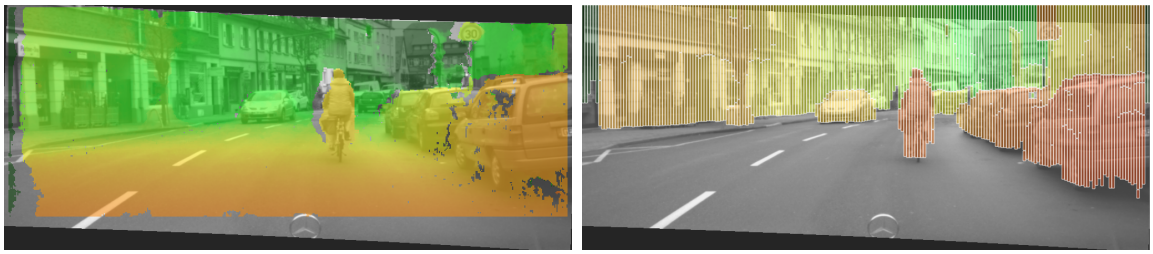


Figure 3.1: Stixel representation of a typical traffic scene. Distances are encoded in color, from red being close to green being far away. A half million disparities in the left image are represented by a few hundred Stixels in the right image. Each colored rectangle in the right image represents a Stixel. Image source: [PF11].

In reviewing the literature of Stixel, we found that there is a considerable amount of redundancy in Stixel data caused by several assumptions of independence introduced by Pfeiffer et al. [PF11] in order to improve the computational performance. These redundancies can be removed through a sophisticatedly designed compression scheme. This motivated us to develop a compression algorithm for Stixels and use it to solve one of the key problems for in-vehicle augmented reality: transmission of depth information.

3.1.1 Key Contributions

In this chapter, we present an efficient lossless compression scheme for Stixels, with the specific objective of transmitting compressed data through a less expensive in-vehicle bus. We are motivated by the fact that there is a considerable amount of redundancy in Stixel data, which originates from several assumptions of data independence [PF11]. These redundancies are removed in our proposed approach through a combination of predictive modeling and entropy coding, which is inspired by the prediction stage of image compression [WSS96, WSS00]. Compared to general purpose compression algorithms, our

approach takes advantage of continuous motion of objects in a common traffic scene for predictive modeling and therefore outperforms them significantly as the evaluation results show.

Our contributions of this chapter are threefold. First, we develop a lossless compression scheme for Stixels, which is, to our best understanding, the first of its kind in the context of depth compression. Second, we show that the compression performance of our proposed algorithm reaches its theoretical upper bound on our recorded dataset. Last but not least, we prove that through Stixel compression, in-vehicle transmission of depth data becomes possible via a more reasonably priced CAN bus. Without the proposed scheme, a more expensive FlexRay bus is needed, which makes the adoption of in-vehicle AR applications more difficult, given the highly cost sensitive nature of the automotive domain. Hence, we believe that our work in this chapter has great relevance for the automotive industry.

3.2 Related Work

Stereo Matching Stereo matching is an approach to *generate* depth information from a pair of aligned cameras. The depth data are encoded in *disparities*, which refers to the pixel difference between the same point in the left image and the right image, as already introduced in Section 2.1.3. The difference between the entire left and right images is referred to as *disparity map*, *disparity image*, or *depth image*.

Traditional approaches [Hir01, HIG02, DDWL10] focused on searching correspondences in local regions in the image, such as edges, corners, or salient objects where correspondences are easily identified. These approaches typically produce either sparse or dense disparity maps, also with blurred boundaries, as they assume a disparity value to be constant within a local window. In 2005, stereo matching was redefined by Hirschmüller [Hir05] as a global optimization problem, referred to as Semi-Global Matching (SGM). He introduced an energy function that comprises two terms representing pixel-wise matching cost and local smoothness, respectively. By minimizing this energy, a globally optimal solution can be found for every pixel. Thanks to hardware implementation [GEM09] of SGM, real-time access to depth information in a car became possible.

However, even if SGM itself was able to run in real-time, the subsequent applications using the output of SGM were still not real-time capable since the processing effort of a disparity map would increase corresponding to its density. This motivated Pfeiffer et al. [BFP09, PF10, PF11, PGS13] to define a novel representation of depth information which is more compact but still suitable for computer vision applications. They named the novel compact mid-level representation “Stixel”.

Stixel Stixels are rectangular slices that are used to “quantize” a disparity image – hence, the name Stixel as the combination of “Stick” and “Pixel”. This representation

turns over half million disparity measurements into only a few hundred Stixels [PF11], which reduces the data volume of depth information significantly.

From a computer science point of view, a Stixel is simply a data structure that defines the position, the size, and the depth of a rectangular slice in a disparity image. The computation of Stixels is formulated as a global segmentation problem [PF11], where each vertical column in the disparity image is segmented into two classes: *obstacle* and *ground*. Stixels are then generated only out of obstacle segments. In other words, Stixels in the disparity image implicitly indicate the existence of obstacles in the 3D world and encode the boundary between free spaces and obstacles. This property is further exploited by recent work [SCR+16, CRS+17] which generate a semantic environmental model for more challenging tasks in the automotive industry, such as in self-driving cars.

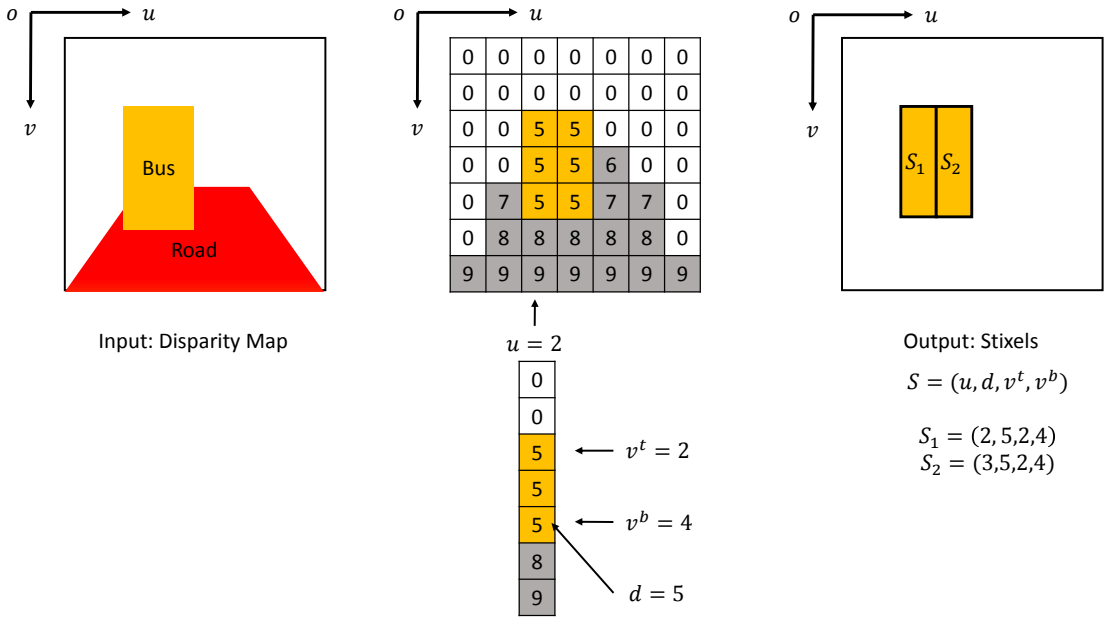


Figure 3.2: Example of Stixel computation.

Figure 3.2 shows an example how Stixels are generated from an input disparity map. In this example, there is only one object *bus* in the scene, which spans two columns in the input depth image. Each column is segmented into obstacle segments (gold) and ground segments (gray). In the third column with the horizontal coordinate[1)] $u = 2$, there is only one obstacle segment with the top vertical coordinate $v^t = 2$ and the bottom vertical coordinate $v^b = 4$. The disparity value of this segment is $d = 5$. Hence, one Stixel $\mathcal{S}(u, d, v^t, v^b)$ is generated in the third column with the symbols $\mathcal{S}_1 = (2, 5, 2, 4)$. Similarly, another Stixel $\mathcal{S}_2 = (3, 5, 2, 4)$ can be generated in the fourth column. From the input disparity map with $7 \times 7 = 49$ disparity values, two Stixels with in total $2 \times 4 = 8$ symbols are generated.

[1)] The horizontal and vertical coordinate start from zero.

Despite the fact that Stixels already represent depth information in a compact way, the communication bandwidth needed to transfer them through a low- or medium-bandwidth in-vehicle bus is still not available. Hence, a relevant research question is whether Stixel data is amenable for further compression to such an extent that they can be transferred via for example a CAN bus. Compression algorithms that work best for Stixels need to be further investigated and developed.

Data Compression In a narrow sense, data compression refers to the procedure to represent the same information using fewer bits. It can be either lossy or lossless. Lossy compression is a irreversible procedure that reduces data amount by removing less important information. In contrast, lossless compression only removes statistical redundancy of the original data which allows it to be perfectly reconstructed. In this work, we decide to focus on lossless compression techniques for the reason that new in-vehicle applications such as AR navigation (Fig. 1.10) require depth information as precise as possible and lossless compression enables transmitting the captured depth measurements without any information loss. ZIP [Int15], PNG [Int04], and GIF [Wor89] are some well-known file formats using lossless compression.

A typical lossless compression algorithm comprises a *modeling* stage which removes redundancies in the original data and an *entropy coding* stage which translates symbols into bit streams [ESU09, NS08, RAS13, Ste94]. Specifically in the area of image compression, redundancy removal during the modeling stage is typically realized through *prediction* [HV93, WSS96, WSS00, WM97], where pixels are predicted from their neighbors and only the residual between the original and the predicted pixels are encoded. If the prediction mechanism is designed properly, the entropy of the symbols to be encoded could be massively reduced, i.e., most of the symbols to be encoded would become zero. For example, the LOCO-I algorithm [WSS96, WSS00] used by the JPEG-LS standard uses three neighboring pixels (upper left, upper, and left) for prediction. The LOCO-I predictor is able to detect edges (discontinuities) in the image in a very simple way and eventually predicts a pixel value that is not on the detected edge.

After the prediction, the entropy coder replaces frequently occurring symbols with fewer bits and rare symbols with many bits. According to Shannon's source coding theorem [Sha48], the optimal code length for a symbol is $-\log P$ where P indicates the probability of the input symbol. Huffman coding [Huf52] and Golomb coding [Gol66] are among the most well-known entropy coding algorithms. The former is ideal if the probability of each input symbol is a power of half, whereas the latter is optimal if the source follows a one-sided geometric distribution.

The *Squash Compression Benchmark* [Nem18] provides a benchmark for data compression algorithms. The benchmark evaluates different codecs at every compression level on different datasets and continues publishing the results on the website. Currently, it contains more than 40 codecs including zpaq [Mah09], zlib [GA95], and brotli [AS16], comprising different derivatives of Lempel-Ziv (LZ77) [ZL77] and Huffman coding [Huf52]. Among them, zlib achieves good performance in both space savings ($\sim$ 80%) and compression

speed ($\sim$ 2 MB/s) on the test dataset of the benchmark. In comparison, zpaq (level-5) achieves the best compression performance with nearly 98% space savings but only operates at 8.5 KB/s, which is lower than the data rate of Stixels[2)] We choose zlib as the representative of general purpose compression algorithms and compare its performance on Stixels against our proposed compression scheme.

3.3 Redundancy in Stixel

The computation of Stixel in [PF11] results in both spatial and temporal redundancies in Stixel data. The spatial redundancy emerges from the assumption that adjacent columns in the disparity image are independent. The temporal redundancy occurs naturally by continuous motion in real traffic scenes.

We can observe spatial redundancy in Stixel data using the example in Fig. 3.2. Due to the fact that the Stixel $\mathcal{S}_1(2, 5, 2, 4)$ and $\mathcal{S}_2(3, 5, 2, 4)$ belong to the same object in the real world, the only difference between them is the horizontal position u. $\mathcal{S}_2$ can be predicted from $\mathcal{S}_1$. In addition, as an example of temporal redundancy, if the bus stays at the same position in the disparity map in the next frame, $\mathcal{S}_1$ and $\mathcal{S}_2$ will also remain the same, which can be predicted from the current frame straightforwardly.

In this section, we discuss the redundancy in Stixel data from a mathematical perspective. First, we briefly review the computation of multilayer Stixel as in [PF11]. Then, we introduce Stixel entropy that defines the theoretical performance limit of Stixel compression. Analytical results of the defined Stixel entropy are presented at the end of this section.

3.3.1 Stixel Computation

As introduced in Section 3.2, the computation of Stixel is formulated as a global segmentation problem where a disparity image is segmented into obstacle and ground. Mathematically, the segmentation is modeled as a Maximum A Posteriori (MAP) estimation problem expressed in Eq. 3.1, where L denotes a column-wise labeling that separates each column of an input disparity image into object segments and ground segments. The objective is to find the most likely labeling L^*, given the corresponding disparity D of the input image.

$$L^* = \arg\max_L P(L|D) \tag{3.1}$$

By applying the Bayes rule, the *a posteriori* probability $P(L|D)$ can be written as the product of the conditional probability $P(D|L)$ and the *a priori* probability $P(L)$, as expressed in Eq. 3.2. The former models the likelihood of the input disparity map by a determined labeling, the latter rates an a priori knowledge about the world model.

$$P(L|D) \sim P(D|L) \cdot P(L) \tag{3.2}$$

[2)] The payload analysis is given in Tab. 3.5.

In order to achieve real-time capability, [PF11] considered the following three types of probabilistic independence. First, neighboring columns are assumed to be independent, i.e., $P(L) = \prod_u P(L_u)$, where u denotes the horizontal coordinate. Second, disparity measurements are considered independent and identically distributed, allowing generalizing $P(D|L)$ to $\prod_u P(D_u|L)$. Third, the disparity measurements in the same column are assumed to be independent of all labeling in other columns, i.e., $P(D_u|L) = P(D_u|L_u)$. These result in a clean statistical separation of the columns, as expressed in Eq. 3.3.

$$P(L|D) \sim \prod_u P(D_u|L_u) \cdot P(L_u) \tag{3.3}$$

It is indeed reasonable to make these assumptions in order to improve computational efficiency. However, the assumptions do not truly reflect the real world. Large objects such as cars and trucks often span multiple columns in an image. The disparities and the labeling of a column can be easily predicted from neighboring columns, especially in traffic scenes where motions are considered continuous and smooth. These lay a theoretical foundation for us to further compress Stixel data.

3.3.2 Stixel Entropy

In order to quantitatively measure the redundancy in Stixel data, we calculate the Shannon entropy [Sha48] of Stixels. Here, we use a four-symbol tuple $\mathcal{S}(u, d, v^t, v^b)$ to encode a Stixel, where u denotes the horizontal coordinate, d the disparity, v^t and v^b the vertical coordinates of the top and the bottom, respectively. The width of a Stixel w is fixed to a constant value and thus not considered a symbol that contains any useful information within the scope of data compression.

Since the four symbols making up a Stixel are not mutually independent, we can only determine the lower bound and the upper bound of their joint entropy using the inequalities expressed in Eq. 3.4.

$$\begin{aligned} H(u, d, v^t, v^b) &\leq H(u) + H(d) + H(v^t) + H(v^b) \\ H(u, d, v^t, v^b) &\geq \max\Big(H(u), H(d), H(v^t), H(v^b)\Big) \end{aligned} \tag{3.4}$$

We analyze the statistical properties of Stixels using a real-world road record that contains more than 5.5 million Stixels. This dataset comprises different road situations such as industrial area and country roads among others. The resolution of an input image is 1024×440 and the minimum width of a Stixel is set to be 3 pixels, i.e., u ranges from 0 to $\lfloor 1024/3 \rfloor = 342$ and $v^t, v^b \in [0, 440]$. A disparity value ranges from 0 to 127, with a step size of 0.25. We remap it to $[0, 508]$ by multiplying the disparity value with factor 4. By doing so, we are able to encode each symbol with nine bits covering $[0, 511]$.

The statistical results of the dataset are presented in Tab. 3.1. The space savings as a measure of the performance of a compression algorithm is defined in Eq. 3.5. Since entropy

indicates the theoretically smallest average symbol length for lossless compression, we use it as the benchmark on this specific dataset to evaluate our proposed compression scheme. According to the results in Tab. 3.1, the average space savings of a compression algorithm for Stixels could theoretically reach 32% – 80% on our recorded dataset.

$$\rho = 1 - \frac{\text{Compressed Size}}{\text{Raw Size}} \tag{3.5}$$

Table 3.1: Example Shannon entropy of our recorded dataset.

Symbol	Entropy	Bit-Length	Space Savings
u	7.1775	9	20.25%
d	6.7558	9	24.94%
v^t	3.6888	9	59.01%
v^b	6.6690	9	25.90%
$\inf S(u, d, v^t, v^b)$	7.1775	36	80.06%
$\sup S(u, d, v^t, v^b)$	24.2911	36	32.52%

3.4 Stixel Compression

In this section, we propose a workflow for compressing Stixels. Recall that Stixel was originally introduced to reduce the data volume of disparity maps while keeping a certain level of data structure for further computer vision algorithms. In other words, the original focus of Stixel was not to compress the data to the greatest extent. In fact, Pfeiffer et al. [PF11] made a series of assumptions during the computation of Stixels which leaves a considerable amount of redundancies in the data, as already discussed in Section 3.3. This motivated us to design a compression scheme that removes the redundancies as far as possible so that we could possibly transfer it through a relatively low-bandwidth in-vehicle communication system like a CAN bus.

Inspired by image compression techniques, we also combine predictive modeling and entropy coding in our proposed compression scheme for Stixels. Typically for image compression, a predictor runs from left to right, top to bottom on the input image, takes pixels in a local neighborhood as references, and predicts the value of the next pixel based on the reference area. The residual between the predicted value and the input value is then encoded through an entropy coder. For Stixel compression, this cannot be directly adopted since unlike pixels in an image, Stixels are not in a rectangular grid. The reason for that is, in a column of an input disparity map, the number of obstacle segments is not determined, i.e., each column might contain different number of Stixels. Hence, we need to first design a grid representation for Stixel data.

To begin with, we first introduce our designed grid representation for Stixels in Section 3.4.1. Then, we explain the first stage of the encoder workflow, namely *Predictive Modeling*, in Section 3.4.2, followed by the second stage *Entropy Coding* in Section 3.4.3. The entire workflow of the encoder is depicted in Fig. 3.3. The encoding workflow ensures that all of the information necessary for a lossless reconstruction of the Stixel data is contained in the output bitstream. First, there is no operation in the encoding workflow such as quantization or low-pass filtering that introduces an irreversible loss of information. In addition, the stage of predictive modeling is designed to be the same in the encoder and the decoder so that the predicted Stixel column $\mathcal{C}_P$ from the same reference group remains the same. Last but not least, we also encode the *mode* of entropy encoding[3)] into a header structure which allows the decoder to determine the correct decoding mode.

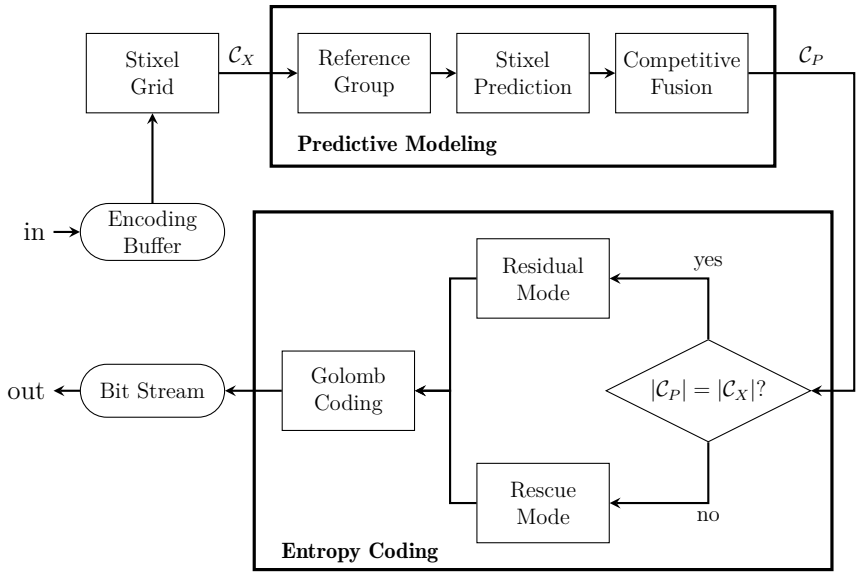


Figure 3.3: Encoding workflow of Stixel compression.

3.4.1 Stixel Grid

In general, neighboring pixels in an image have strong spatial correlations and therefore can be predicted based on a local reference area. As already shown in the example at the beginning of Section 3.3, Stixels have both temporal and spatial redundancies that can be removed using prediction. In order to do so, we need a grid representation for Stixels that incorporates both temporal and spatial domain. This is the reason why we propose the temporal-spatial neighborhood matrix shown in Fig. 3.5.

Concatenating all Stixels within the same column yields a Stixel Column $\mathcal{C} = \mathcal{S}_0\mathcal{S}_1 \ldots \mathcal{S}_{n-1}$, with n being the number of Stixels in that column. The vertical coordinates satisfy $v_0^t < v_0^b < v_1^t < v_1^b < \cdots < v_{n-1}^t < v_{n-1}^b$, i.e., Stixels are ordered from image top to image bottom, following the definition of a common image coordinate system. The Stixel

[3)] Residual mode or rescue mode. More details in Section 3.4.3.

Column is the canonical element in our proposed compression workflow. It can be differentiated by another Stixel Column if they contain the same number of Stixels. The difference between two Stixel Columns is referred to as Residual Column $\mathcal{R} = \partial\mathcal{S}_0\partial\mathcal{S}_1 \ldots \partial\mathcal{S}_{n-1}$.

As in [PF11], Stixels are not required to be vertically adjacent, i.e., a ground segment between two obstacle segments in the same column is allowed. Therefore, both v_t and v_b are needed to specify the vertical position of a Stixel. Now, we enforce Stixels in the same column to be adjacent by filling placeholder Stixels $\mathcal{S}_g$ into gaps between existing Stixels (Fig. 3.4). Be doing so, we can save one symbol for the vertical coordinate (either v^t or v^b) as one can then be derived from the other using Eq. 3.6.

$$v^b_{i-1} = v^t_i - 1, i = 1 \ldots n-1 \tag{3.6}$$

In the following text, v^t will be omitted and v^b will be denoted as v.

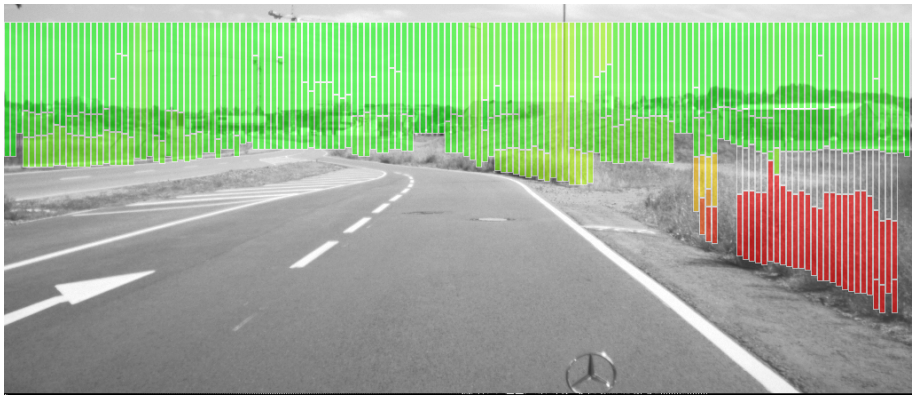


Figure 3.4: Ground Stixel as a place holder. Illustrated by the white rectangle frames without filling color.

We define a Stixel Frame $\mathcal{F} = \mathcal{C}_0\mathcal{C}_1 \ldots \mathcal{C}_{N-1}$ as the concatenation of all Stixel Columns from image left to image right. The number of Stixel Columns N within a Stixel Frame is a constant. A Stixel Frame encodes the scene in the spatial domain.

Stacking temporally adjacent Stixel Frames together yields the desired grid representation for Stixels, which we refer to as temporal-spatial neighborhood matrix (Fig. 3.5). Each row of the matrix represents a Stixel Frame, whereas each column of it records the temporal change of the same Stixel Column. Relying on this grid representation, we can define a reference area (bounded by the black box in Fig. 3.5) where Stixels are both spatially and temporally correlated. The next step is then to design a predictor that removes these redundancies.

3.4.2 Predictive Modeling

Figure 3.6 illustrates the complete scheme for predicting a Stixel Column based on the reference area. It comprises three major steps: building Stixel reference groups, predicting the next Stixel to be encoded, and fusing the predicted results competitively.

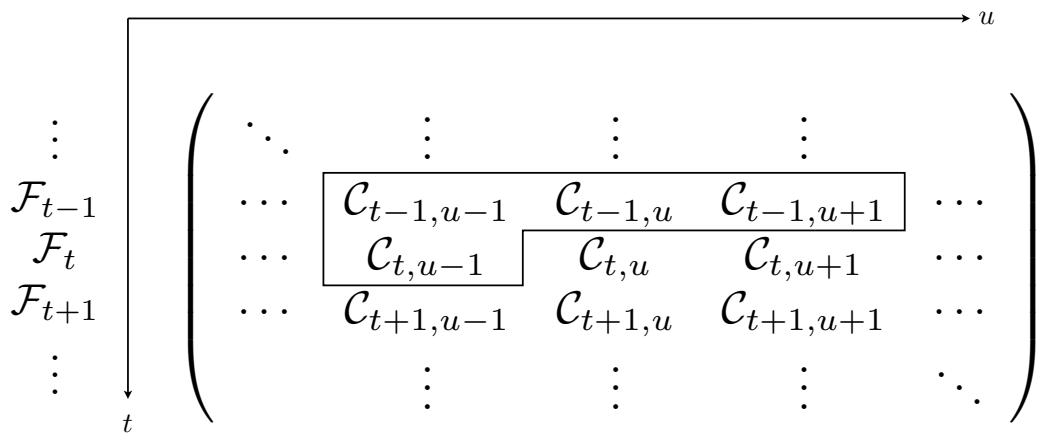


Figure 3.5: Temporal-spatial neighborhood matrix of Stixel Columns.

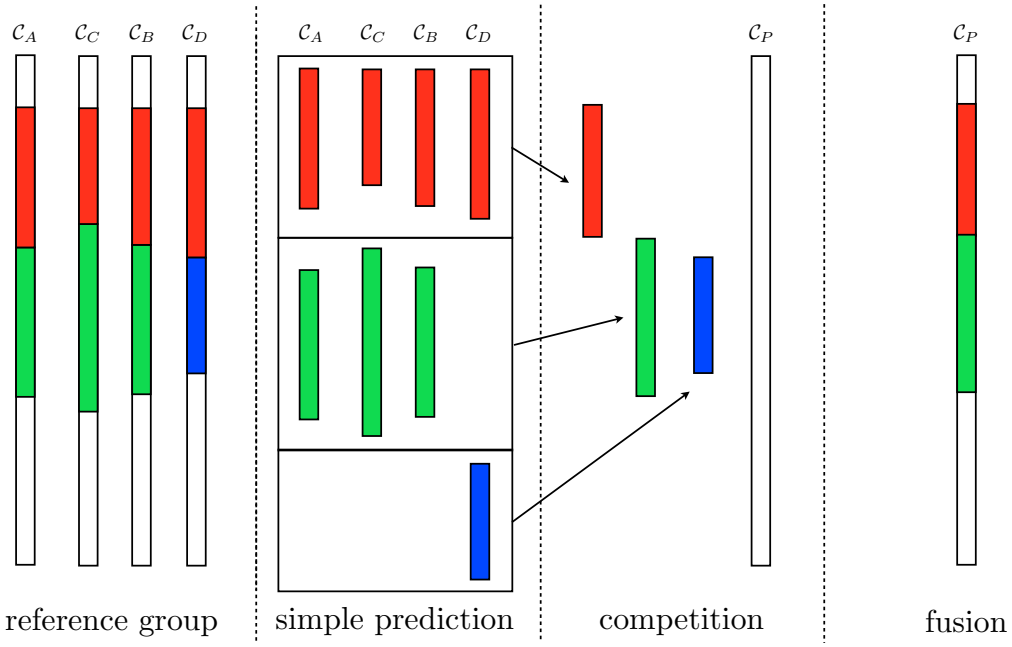


Figure 3.6: Scheme for Stixel Column prediction.

Reference Group We first cluster Stixels within a reference area into smaller groups according to their similarity. The purpose of clustering is to roughly identify Stixels that belong to the same object in the real world, since objects often span multiple columns and multiple frames in a traffic scene.

We calculate the dissimilarity between two Stixels using Eq. 3.7 where $l = v^b - v^t$ denotes the length of a Stixel. The three weighted terms from left to right measure the difference between their disparities, lengths, and vertical boundaries, respectively. The controlling weights are set to be $\lambda_1 > \lambda_2 > \lambda_3$, i.e., we put more emphasis on the difference between disparities over the other two factors. Two Stixels are grouped together if their dissimilarity indicator Δ is smaller than certain threshold τ_s.

$$\begin{aligned}\Delta\left(\mathcal{S}_1, \mathcal{S}_2\right) =& \lambda_1 |d_1 - d_2|^2 + \lambda_2 |l_1 - l_2|^2 \\ &+ \lambda_3 \left(|v_1^b - v_2^b|^2 + |v_1^t - v_2^t|^2\right)\end{aligned} \tag{3.7}$$

Stixel Prediction After the reference area is clustered, one Stixel is predicted from each reference group in a similar manner as the LOCO-I prediction introduced in Section 3.2. We prefer the LOCO-I algorithm over the others for the following reasons. First, the memory requirement of LOCO-I is simple to meet, as the LOCO-I predictor only takes three neighboring pixels as input. Second, LOCO-I only performs addition or subtraction operations, which allows very fast computation. Last but not least, the LOCO-I predictor is capable of detecting discontinuity within the reference area. In our case, this property turns truly useful, since a gap in depth between adjacent Stixels appears to be a common case in traffic scenes. The ability to detect discontinuities in depth allows us eliminating false references during the prediction stage and thus to improve the overall performance of the entire compression workflow.

In order to be consistent with LOCO-I, we use $\mathcal{C}_X$ to denote the current Stixel Column $\mathcal{C}_{t,u}$ in Fig. 3.5 in the rest of this section. Similarly, $\mathcal{C}_{t,u-1}$ is substituted by $\mathcal{C}_A$, $\mathcal{C}_{t-1,u-1}$ by $\mathcal{C}_B$, $\mathcal{C}_{t-1,u}$ by $\mathcal{C}_C$, and $\mathcal{C}_{t-1,u+1}$ is replaced by $\mathcal{C}_D$, as shown in Fig. 3.7. The predicted Stixel column is denoted as $\mathcal{C}_P$.

$$\begin{pmatrix} & & & & \\ \ddots & \vdots & \vdots & \vdots & \\ \cdots & \mathcal{C}_B & \mathcal{C}_C & \mathcal{C}_D & \cdots \\ \cdots & \mathcal{C}_A & \mathcal{C}_X & & \cdots \\ & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Figure 3.7: Simplified notation of the temporal-spatial neighborhood matrix.

Equation 3.8 defines the function used to predict the disparity. It takes 1) d_A if a temporal discontinuity (a huge gap between d_A and d_C) is detected; 2) d_B if a spatial discontinuity

(a huge gap between d_B and d_C) is detected; 3) the combination $d_A + d_B - d_C$ if no discontinuities are detected.

$$d_P = \begin{cases} \min(d_A, d_B), & \text{if } d_C \geq \max(d_A, d_B) \\ \max(d_A, d_B), & \text{if } d_C \leq \min(d_A, d_B) \\ d_A + d_B - d_C, & \text{otherwise} \end{cases} \tag{3.8}$$

The boundary value v_P is straightforwardly predicted through linear interpolation, based on the assumption that Stixels in the same reference group tend to form large rectangle or trapezoids in an image [PGS13]. This assumption is reasonable for objects with clear and sharp edges, such as cars or trucks.

If a reference group contains less than three Stixels, we take the average of the available references as the prediction.

Competitive Fusion Predicted Stixels have to compete against each other if they vertically overlap with more than τ_v pixels. A confidence score is assigned to each predicted Stixel according to the availability of its references during the prediction stage. The one with the higher confidence score wins the competition. For example, a Stixel predicted based on three reference Stixels from column $\mathcal{C}_A, \mathcal{C}_B$, and $\mathcal{C}_C$ is considered more trustworthy than that with only one Stixel reference from column $\mathcal{C}_D$.

In the last step, all remaining Stixels are fused together, yielding the predicted Stixel Column $\mathcal{C}_P$.

After experimenting with different combinations of the parameters (Fig. 3.8 and Fig. 3.9), we choose $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 1$, together with $\tau_s = 285$ and $\tau_v = 11$, which yields the best space saving factor on our recorded dataset. The results in Fig. 3.8 suggest that the dissimilarity measure puts more weight on the depth and length difference than the difference of vertical position. The results in Fig. 3.9 indicate that there exists a lower bound for both τ_s and τ_v. The influence on the final space savings become trivial once they are set to be bigger than the lower bounds.

3.4.3 Entropy Coding

The entropy coder at the end of the workflow has two different modes: *Residual* mode and *Rescue* mode. It runs residual coding if a differentiation between $\mathcal{C}_P$ and $\mathcal{C}_X$ is possible, i.e., the number of predicted Stixels is equal to the number of Stixels in the current Stixel Column. If the predicted column has more or less Stixels than the current one, the coder switches to rescue coding.

In residual mode, the residual column between $\mathcal{C}_P$ and $\mathcal{C}_X$ is encoded through a modified version of Golomb code. A source symbol $x \in \mathbb{N}$ is divided by a Golomb divisor $m \in \mathbb{N}^+$, yielding a quotient $q \in \mathbb{N}$ and a remainder $r \in \mathbb{N}$, i.e. $x = q \cdot m + r$. The quotient is

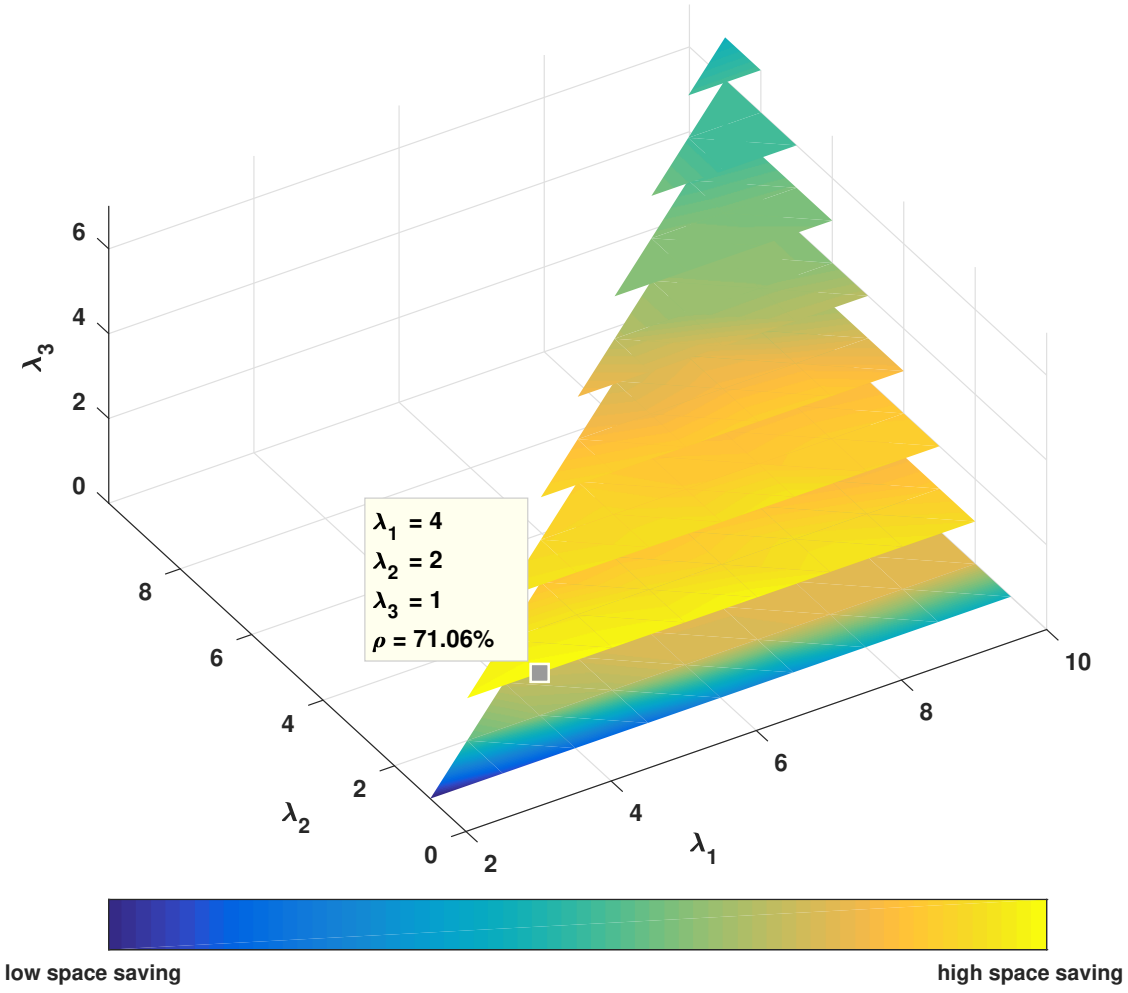


Figure 3.8: Experiment on different λ by Stixel compression. Fixing τ_s and τ_v, $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 1$ yield the best space saving results on our recorded dataset.

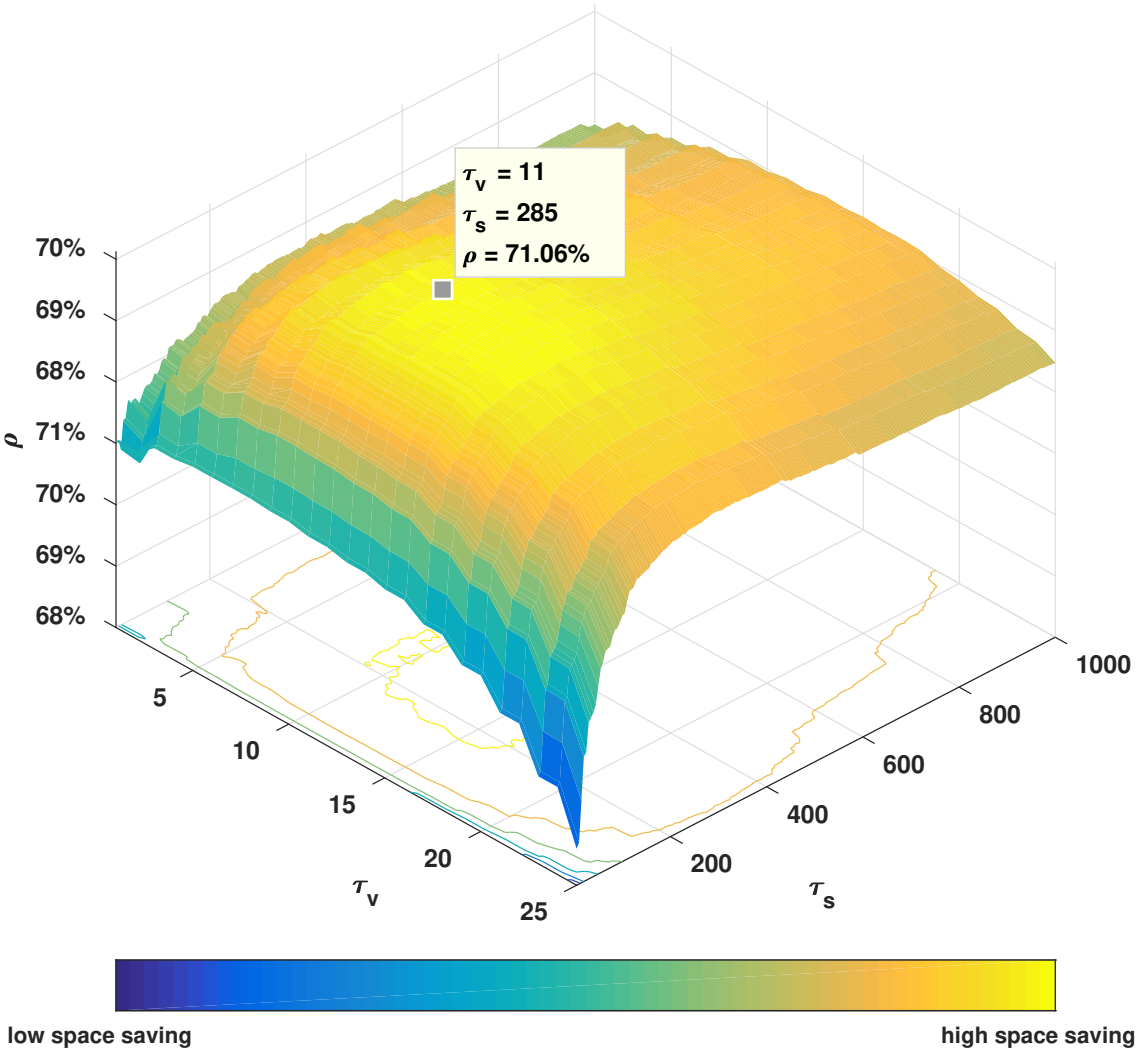


Figure 3.9: Experiment on different τ_s and τ_v by Stixel compression. Fixing λ_1, λ_2, and λ_3, $\tau_s = 285$ and $\tau_v = 11$ yield the best space saving results on our recorded dataset.

Table 3.2: Statistical properties of residuals for different reference combinations.

Combination	References				Occurrence	Entropy	
	$\mathcal{C}_A$	$\mathcal{C}_B$	$\mathcal{C}_C$	$\mathcal{C}_D$		$H(r_d)$	$H(r_v)$
#1				×	16.19%	6.2664	5.4195
#2			×		8.27%	6.0334	5.5463
#3			×	×	0.62%	4.9560	4.5351
#4		×			7.60%	5.5494	5.7342
#5		×		×	6.61%	2.7966	3.3201
#6		×	×		2.58%	3.9158	4.5833
#7		×	×	×	3.30%	4.0536	4.4018
#8	×				10.06%	4.5559	4.7061
#9	×			×	1.26%	3.4223	3.6910
#10	×		×		7.60%	4.4382	4.4595
#11	×		×	×	0.99%	3.3246	4.1816
#12	×	×			1.39%	3.6463	3.9729
#13	×	×		×	2.06%	2.9732	3.7330
#14	×	×	×		7.48%	2.5996	3.7083
#15	×	×	×	×	23.97%	2.0705	2.2218

encoded through an unary code, whereas the remainder is encoded through a modified binary code. Theoretically, there exists an optimal Golomb divisor m^* if the source follows a one-sided geometric distribution [Gv75]. In practice, m is usually rounded up to the next power to 2 so that the encoder can be efficiently implemented through binary arithmetic [WSS00, LQY$^+$10]. In order to find the optimal Golomb divisors by different reference combinations, we examined probability distributions of the residuals using our recorded dataset with 21 844 Stixel Frames. Table 3.2 shows the entropy of the residuals by different combination patterns according to the availability of Stixel references in a reference area. For example, the combination pattern #1 indicates that only one reference Stixel from Stixel Column $\mathcal{C}_D$ is available. Fig. 3.10 shows the distribution of the residuals by the three most frequent combination patterns: #15 where all four Stixel references are available, #1 where only one reference from column $\mathcal{C}_D$ is available, and #8 where only one reference from column $\mathcal{C}_A$ is available. Since the residuals are two-sided distributed, we use Eq. 3.9 to map the negative values back to the positive side.

$$x' = \begin{cases} 2x, & \text{if} \quad x \geq 0 \\ -2x - 1, & \text{if} \quad x < 0 \end{cases} \tag{3.9}$$

The Golomb divisor by different reference combinations is updated adaptively based on the "one-liner" implementation as suggested in [WSS96]. We also use different Golomb divisors for disparity residual and vertical boundary residual since their statistical properties are different.

In rescue mode, a Stixel is modeled by $\mathcal{S}(d, l)$ (l for the vertical length), and it is straight-

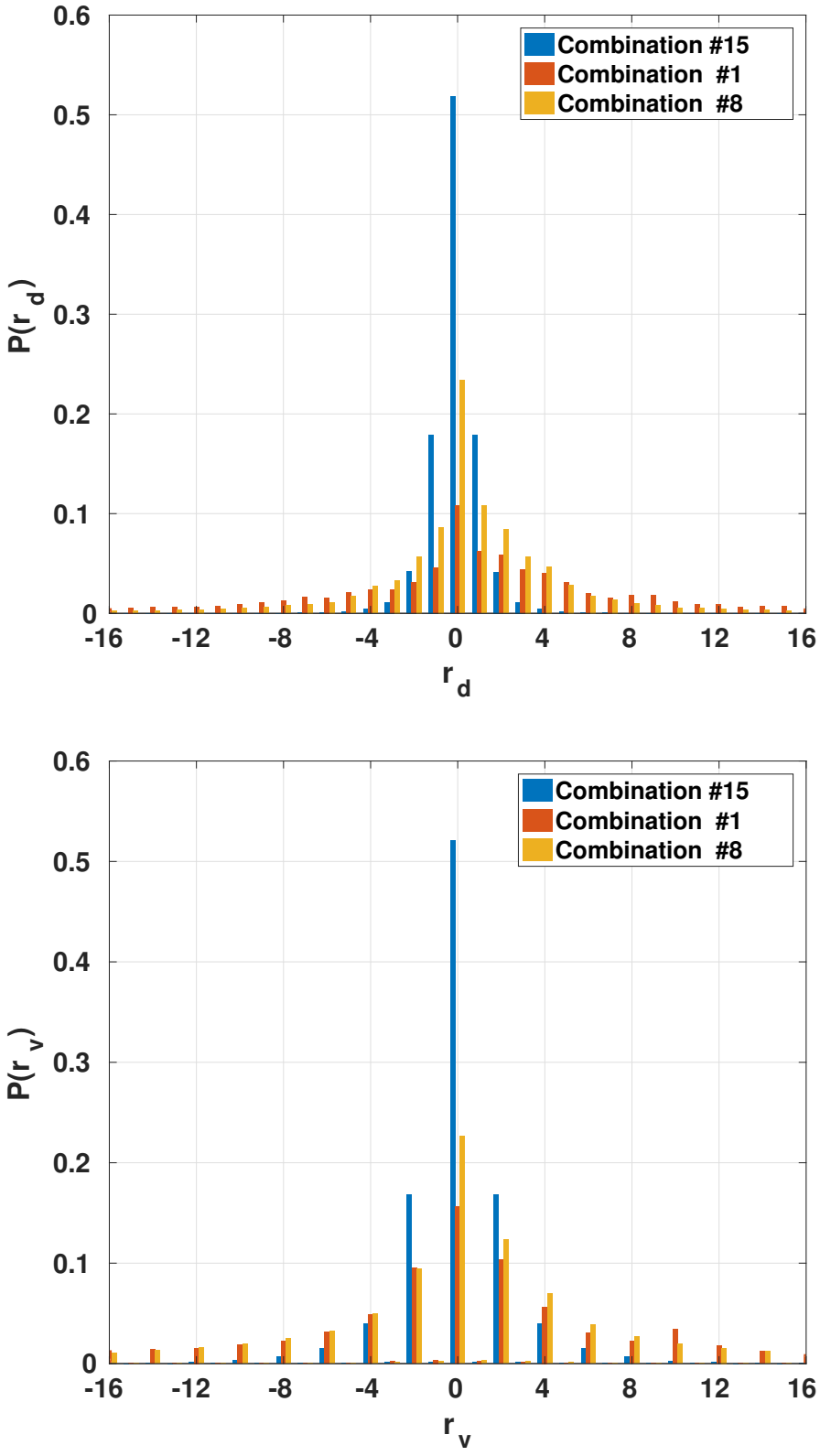


Figure 3.10: Probability distributions of disparity residuals (top) and vertical boundary residuals (bottom) for the most frequent reference combinations.

forwardly encoded through binary coding. The horizontal coordinate u is recorded in a header structure of the current Stixel Frame, which is necessary for the decoder to determine which columns are in rescue mode before it starts to decode.

3.4.4 Experimental Results

We evaluate the proposed compression algorithm for Stixel data using a 15-minute real world record of traffic scenes. The record contains 21 844 frames and more than 5.5 million Stixels. The proposed approach, referred to as *Space+Time* as predictions are made based on both spatial and temporal neighbors, is compared with a stand-alone *Space* approach as well as a stand-alone *Time* approach where no temporal or spatial references are taken, respectively. In addition, we considered including several general purpose compression algorithms into our comparison group, and we eventually decided to use *zlib* which achieved a good performance in both space saving and compression speed in the Squash compression benchmark.

Table 3.3: Space savings and processing time by Stixel compression. The compression Software is implemented using C++ without multi-threading. The processing time is recorded on an Intel Core i7-5960X CPU.

Approach	Space Savings			Processing Time (μs)		
	min	max	avg.	min	max	avg.
Space+Time	-0.77%	86.93%	71.95%	158	1348	327
Space	20.16%	80.92%	69.15%	25	269	34
Time	-86.50%	85.46%	70.45%	29	230	48
Zlib	0.91%	29.91%	8.55%	50	298	78

The results in Tab. 3.3 show that the Space+Time approach achieves the highest space saving ratio, which is better than the stand-alone Time approach as well as the stand-alone Space approach. Although it also takes the longest time to process, the Space+Time approach still meets the real-time requirement since our stereo vision system only operates at 25 frames per second.

The advantage of having a carefully designed prediction stage in our proposed compression algorithm is clearly proven through the evaluation results. All of our three approaches (Space, Time, and Space+Time) which incorporate predictive modeling significantly outperformed the general purpose compression approach zlib, where a combination of LZ-77 and Huffman Coding is carried out during compression. The major weakness of, not only zlib, but any kind of general compression algorithm is, that it does not necessarily have to understand how the original data is generated and is thus not sufficiently able to remove the redundancies before the entropy coding stage. In our proposed predictive modeling stage, we carefully considered the mathematical procedure how Stixels are generated, the physical shapes of other road users, and the nature of continuous motion in traffic scenes, which eventually led to our decisions to design a LOCO-I-like predictor for Stixels. These

are the main reasons why our proposed Stixel compression scheme is non-replaceable by a general purpose compression algorithm.

In addition, we picked up the best case scenario (Fig. 3.11) and the worst case scenario (Fig. 3.12) in the 15-minute record according to their instant compression performance (Tab. 3.4). In the best case scenario, the ego-car stopped in front of a crossroad, waiting for the traffic light. The scene in front of the camera remained almost the same, of which the temporal prediction takes full advantage. In the worst case, a truck drove from left to right in the image. It nearly covered the entire field-of-view of the camera, causing problems for Stixel computation. The Stixels appeared to be randomly segmented, which generated massive errors during the prediction stage, disturbing the probability distribution of the residuals. As a result, a great amount of residual bits were needed by the entropy encoder.

Table 3.4: Extreme cases by Stixel compression.

	Approach	Original Bits	Residual Bits	Rescue Bits	Space Savings
Worst Case	Space+Time	5868	5196	717	-0.77%
	Space	5868	4553	132	20.16%
	Time	5868	10143	801	-86.50%
Best Case	Space+Time	9540	1247	0	86.93%
	Space	6624	1066	198	80.92%
	Time	9576	1392	0	85.46%

Figure 3.13 and Fig. 3.14 illustrate instant compression performance of the Space+Time approach. The performance is heavily influenced by the traffic scenes. The curve of the compressed data volume in Fig. 3.13 remains flat when the ego-car drives straightforward, and it turns spiky when unexpected objects appear in the scene. Most of the time, the space savings in Fig. 3.13 stays close to the upper bound of the theoretical limit of Stixel compression, except the afore-mentioned worst case scenario.

Furthermore, we analyzed the payload[4)] requirement for transmitting Stixels through different in-vehicle buses (Tab. 3.5). Note that the requirement is influenced by the *maximum* of instant data volumes instead of the average value. Also, it is worth mentioning that depending on the structure of the data frames, the payload capacity of a in-vehicle bus generally only reaches a percentage of its maximum bandwidth. For example, a CAN data frame [Int03] has a maximum frame length of 128 bits, in which the actual data field is only up to 64 bits. In other words, only the half of its maximum bandwidth can be used for carrying the user data.

According to Tab. 1.1, the CAN bus has a maximum bandwidth of 500 kbps = 62.5 KB/s, which yields a maximum payload capacity of 62.5 KB/s × 50% = 31.25 KB/s. Therefore, we come to the conclusion that the more expensive FlexRay is required for transmitting

[4)] Data packets that carry user data rather than control information.

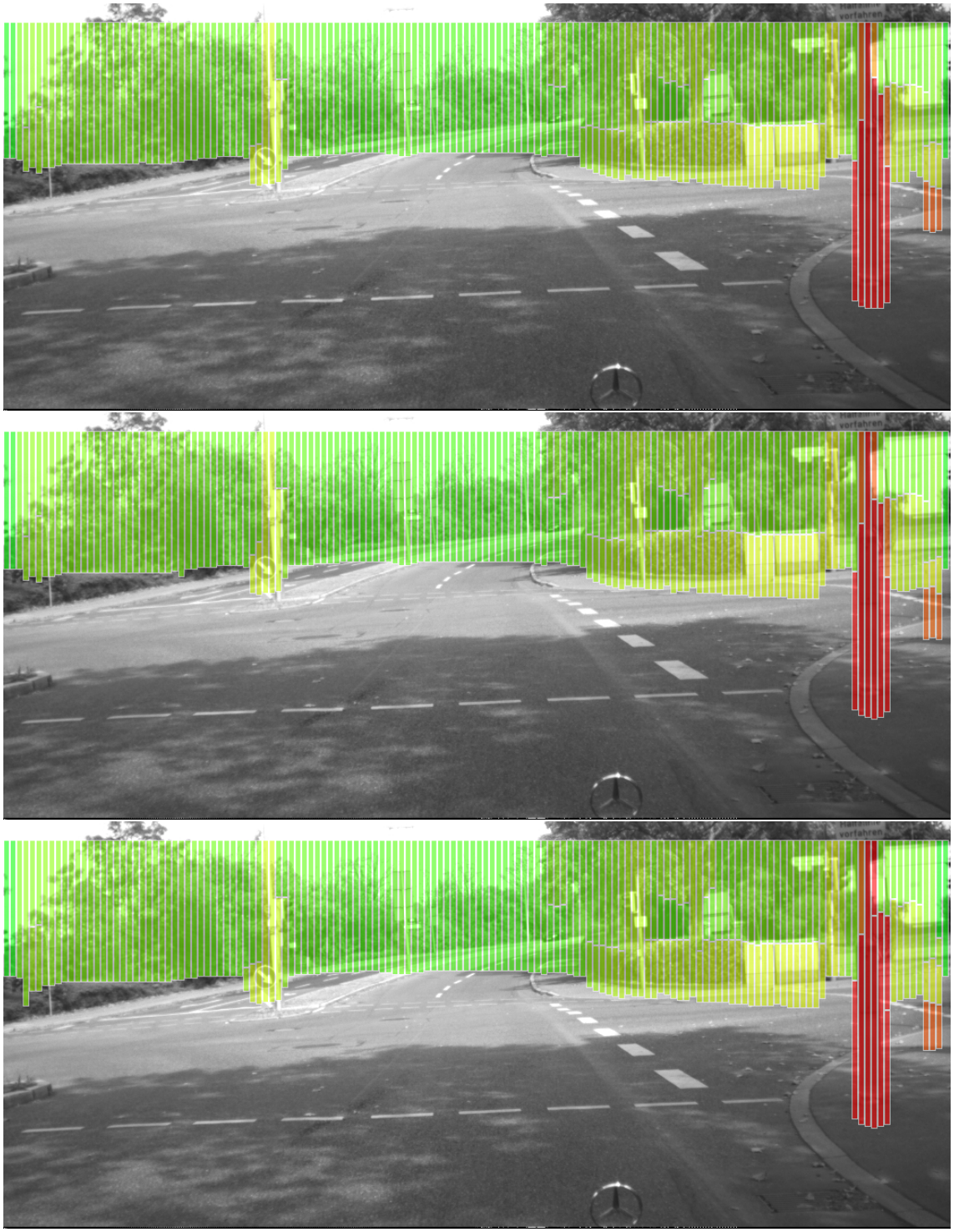


Figure 3.11: The best case scenario by Stixel compression.

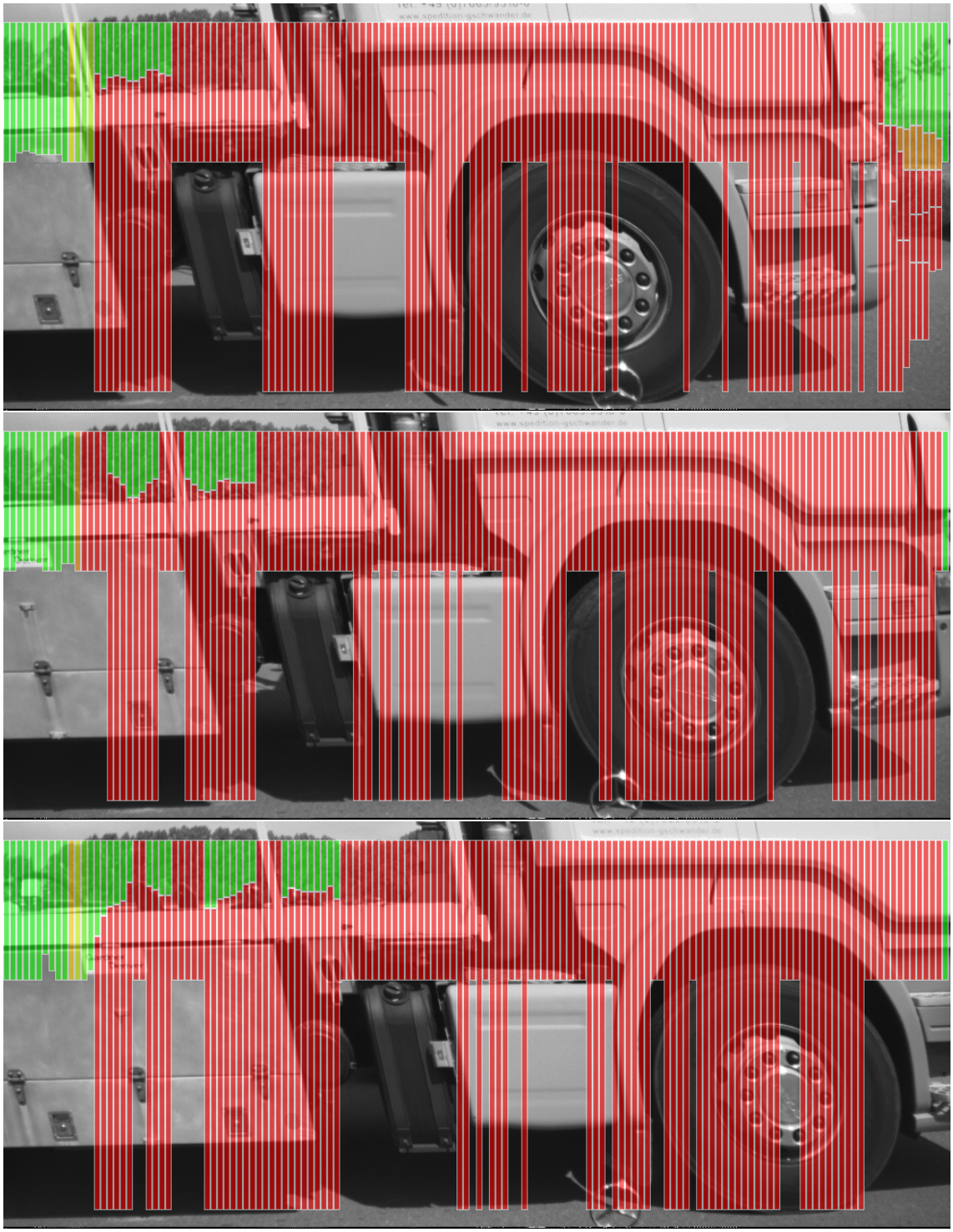


Figure 3.12: The worst case scenario by Stixel compression.

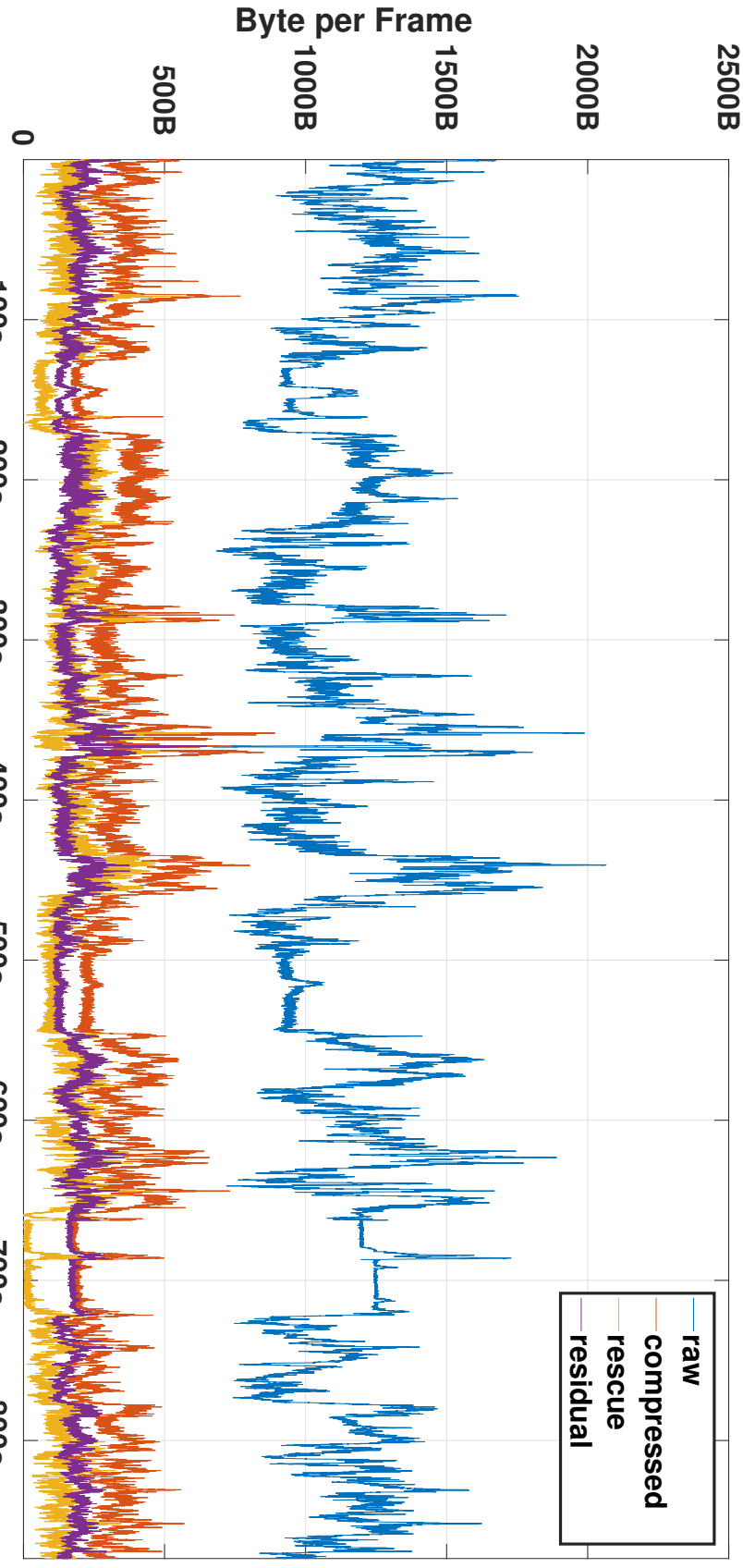


Figure 3.13: Instant data volume of Stixels.

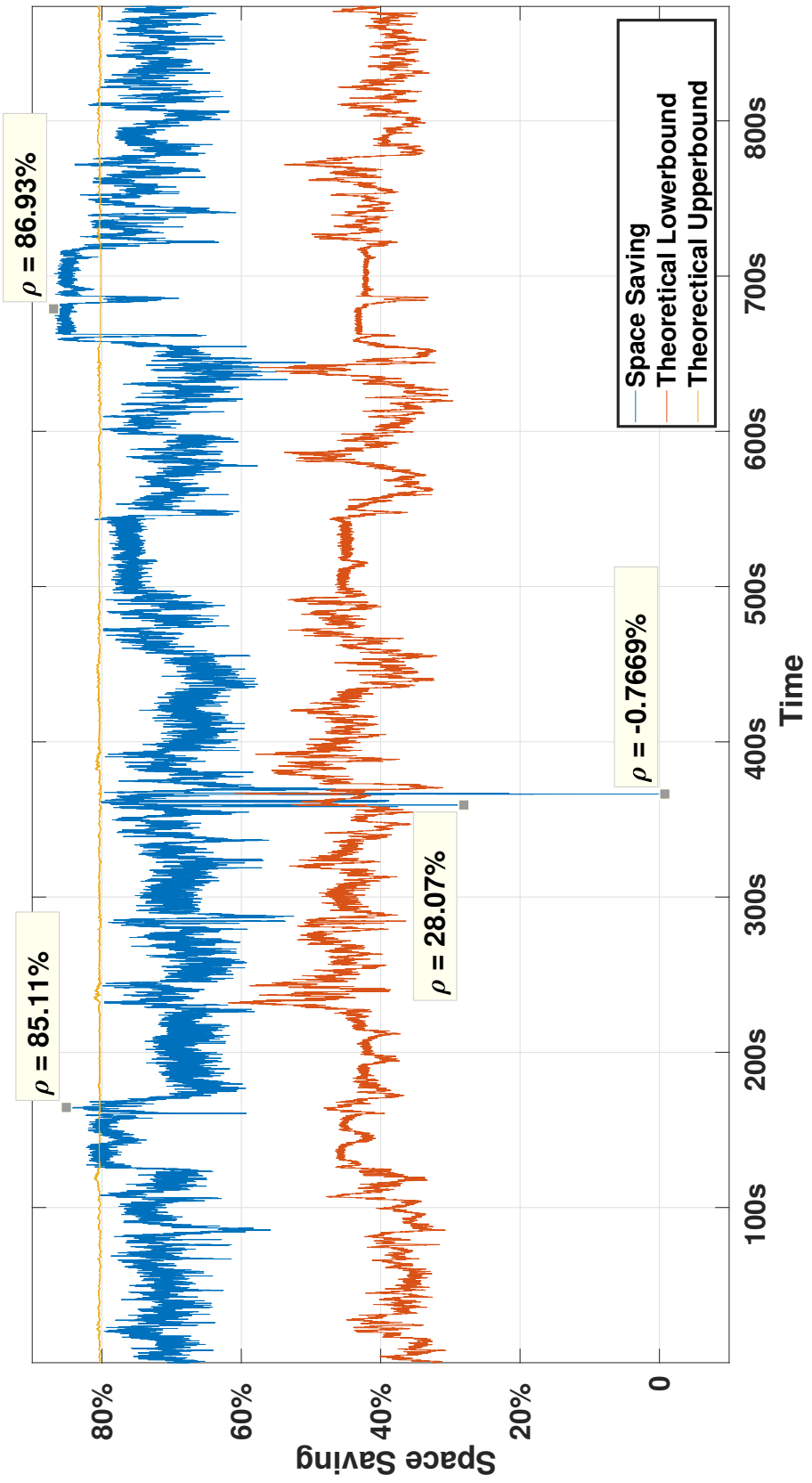


Figure 3.14: Instant ratio of space savings by Stixel compression. The theoretical upper bound and lower bound are derived based on Stixel entropies.

Table 3.5: Payload analysis for transmitting Stixels.

	w/o	Space+Time	Zlib
Instant Peak (KB)	2.02	0.87	1.93
Required Payload (KB/s)	46.37	17.68	46.17
Required Vehicle Bus	FlexRay	CAN	FlexRay
Cost	High	Medium	High

uncompressed Stixel data or compressed data using zlib, whereas compressed Stixel data through our approach can be transmitted via the CAN bus. For in-vehicle augmented reality applications that require depth information in the telematics domain, the CAN bus connecting the central switch and the telematics gateway would have to be replaced by a FlexRay bus, if our proposed compression scheme is not deployed. Also, the telematics gateway would have to be redesigned in order to be able to decode both CAN and FlexRay signals. From the point of view of automotive system architects, such a big scale of modification would be unrealistic for near future production cars. Therefore, we believe that our proposed Stixel compression algorithm is a key enabler for near future in-vehicle applications such as in-vehicle AR.

3.5 Summary and Future Work

In this chapter, we introduced a novel algorithm for compressing Stixel data, as a key solution to the problem of transmitting depth information through a reasonably priced in-vehicle communication system.

The lossless compression algorithm comprises a predictive modeling stage and an entropy coding stage. During the predictive modeling stage, a Stixel Column to be encoded is first predicted based on reference Stixels in its spatial and temporal neighborhood. The residual column is subsequently encoded through Golomb coding, an entropy coding method which is optimal for data sources following geometric distribution. Experiment results show a convincing performance of our proposed algorithm, which achieves nearly 72% of average space savings, which almost reaches the upper bound of the theoretical compression limit on our recorded dataset. The proposed compression algorithm enabled us to transmit Stixel data via a less expensive CAN bus, which has great relevance for the cost-sensitive automotive industry.

Although we already achieved convincing results, there are still improvements in our compression algorithm which could be made in the future. Different reference areas and alternative prediction directions could be experimented, in order to increase the robustness by predictive modeling. The instant peaks in the compressed data stream could be eliminated by combining lossy and lossless compression. In the near future, we will devote ourself in exploring these possible improvements. We believe that our proposed compression scheme has great potential to be adopted in series production cars.

4 Scene Understanding

One can not know the true nature of Mount Lu, for one is on that very mountain itself.

Su Shi
1037 - 1101

The Song dynasty poet Su Shi, also known as Su Dong-Po, wrote the beautiful Chinese quatrain [Ven10] quoted at the beginning of this chapter after he visited the mount Lu in today's Jiangxi Province, China. Instead of depicting a specific scenery and expressing his nature-loving, the poetry attempted to share one of his philosophical views, enlightening us how difficult it is to grasp the entire picture of things from a narrow viewpoint. Today, after almost thousand years, the idea of Su still echoes in the modern society. For example, from the point of view of a driver, we can interpret Su's poet as how difficult it is for us to get a picture of our entire surroundings through our view through the windshield. It is impossible to understand the entire traffic scene when we are part of the scene.

Scene understanding is indispensable for in-vehicle augmented reality. Here, in this chapter, we introduce a novel workflow to 3D shape reconstruction named *3D Shaping* and elaborates its necessity for in-vehicle augmented reality. We begin with a brief introduction of scene understanding and related work in Section 4.1 and Section 4.2, respectively. In Section 4.3, we present the workflow of monocular 3D Shaping that only requires a single image from a monocular camera as input. In Section 4.4 and Section 4.5, we introduce two extensions to monocular 3D Shaping by using an additional in-vehicle 3D sensors, in order to make 3D Shaping more practical for traffic scenes. In the last section, Section 4.6, we summarize this chapter and outline directions for future work.

4.1 Introduction

Scene understanding covers a series of problems from object detection, semantic classification, to 3D reconstruction, and environmental modeling. It is still one of the most challenging research topics in computer vision and robotics today. In the automotive industry, technologies based on scene understanding are already widely used. For example, an in-vehicle computer vision system is able to recognize road marks, detect pedestrians, and predict their movements to some extent, in order to support ADAS functions such as lane-keeping assist and brake assist. However, for more challenging scene understanding

tasks such as those required by an in-vehicle AR system or in a self-driving car, traditional computer vision approaches based on hand-crafted features are not sufficient anymore.

In recent years, *Deep Learning* marked a new era of computer vision by setting records in a range of scene understanding tasks. Deep-learning-based approaches use complex multilayer neural networks that process visual inputs in a similar way as the visual cortex in a human brain. The word "deep" refers to the number of layers used in a neural network. For example, the well-known neural network AlexNet [KSH12] published in 2012 has only 8 layers, whereas a more recent neural network named ResNet [HZRS16] published in 2016 contains more than 150 layers. Several research projects in the context of self-driving car already started to use deep neural networks for semantic segmentation of traffic scenes. Automotive hardware with massive processing power that supports deep learning is also available on the market and is being further developed.

For in-vehicle augmented reality, building a 3D environmental model at an object level is an essential problem, as already mentioned in Section 1.4.2. The challenge lies not only in finding an appropriate modeling algorithm but also in distributing the algorithm across the in-vehicle E/E architecture, to be more specific, across the ADAS domain and the telematics domain. On the one hand, the approach to generate a 3D environmental model should be run-time efficient enough so that it could possibly run in real-time. On the other hand, the 3D shapes of objects in the generated environmental model should be described using as few parameters as possible, since they need to be transfered from the ADAS domain to the telematics domain via a low- or medium-bandwidth in-vehicle communication system. In the rest of this chapter, we explain how our specific solution for object-level 3D reconstruction, namely 3D Shaping, copes with these challenges.

4.1.1 Key Contributions

In a nutshell, the key contribution of this chapter is the proof of concept of object-level 3D reconstruction in a car using our proposed 3D Shaping and its extensions. First, we introduce monocular 3D Shaping (Section 4.3), which requires only a single image frame taken by a monocular camera as input for 3D shape reconstruction. We adapt an existing 3D reconstruction technique [PSR12] and combine it with the state-of-the-art deep neural network, which achieves 20% performance gain in viewpoint estimation. In addition, in order to deal with real traffic scenes, we propose two extensions to monocular 3D Shaping by using additional 3D sensors, which we refer to as *Pose-RCNN* and *3D Shaping + Lidar*, respectively. The proposed Pose-RCNN (Section 4.4) for joint object detection and viewpoint estimation achieves the highest detection and orientation scores for easy scenarios of the KITTI detection benchmark [GLU12]. The proposed 3D Shaping + Lidar (Section 4.5) is able to reconstruct multiple objects in a traffic scene with self-occlusion, and it improves the accuracy of occupancy estimation by more than 80% against the monocular 3D Shaping. Using the proposed monocular 3D Shaping together with its extensions, we prove the concept of object-level 3D reconstruction in a car, which is essential to solve the key problem introduced in Section 1.4.2 for in-vehicle AR.

The specific relevance of 3D Shaping to in-vehicle AR is the use of a *Latent Shape Space* (Section 4.3.1), where various 3D geometries can be represented using only two parameters. This enables transmitting highly complex 3D shapes through a low- or medium-bandwidth communication system[1)]. Therefore, we believe that the proposed 3D Shaping approach is a key enabler for future in-vehicle AR applications to truly augment the 3D world.

4.2 Related Work

Single Frame 3D Reconstruction Early approaches to 3D reconstruction using a single image include *Shape from Shading* [Hor89], *Shape from Contour* [HB88], and *Shape from Silhouette* [Lau94]. Oswald et al. [OTNC13] gave a detailed review of the early approaches. The early approaches reconstruct 3D shapes relying on a single image cue such as light source, object boundaries, or surface discontinuities among others. Recently, 3D reconstruction approaches commonly combine multiple image cues to improve the quality of reconstruction. Guan et al. [GWBB09] combined shading, edges, and silhouette to estimate human body shape using a single photograph or painting. The human body was represented using a deformable mesh model called SCAPE [ASK+05], which is capable of recovering non-rigid deformations caused by articulation. However, user input was needed to initialize the estimation of the pose and the silhouette of a human. Cashman et al. [CF13] reconstructed the 3D geometry of a dolphin by combining occluding contours and interest points in an image. They represented the shape through a 3D morphable model which comprises a mesh of control vertices and an interpolation rule. The interest points needed to be initialized through user input as well. Kholgade et al. [KSES14] introduced a photo-editing software application which enabled 3D object manipulation in an image. The application estimated the illumination and reflectance of the environment, which enabled rendering more realistic object shadows and surface illumination during object manipulation. A user has to provide keypoint correspondences and masks for the ground and the shadow area in an image for the 2D-3D alignment.

Although the afore-mentioned recent work on single frame 3D reconstruction already achieved competitive results, user input was still required for the initialization, which made them impractical for in-vehicle applications. We need a fully autonomous step to initialize the pose (position and orientation) of the object that we want to reconstruct from a single image. This is one of the essential problems of single frame 3D reconstruction.

Latent Shape Another essential problem of 3D reconstruction is how to efficiently represent the 3D geometry. Zia et al. [ZSSS11] trained deformable wireframe models for cars and bicycles based on an Active Shape Model [CTCG95]. A wireframe model is indeed a low-dimensional representation, but we are not able to recover the complete 3D geometry of an object from a wireframe model. In order to recover the entire shape of an object,

[1)] The analysis regarding the bandwidth is presented in Chapter 5.

Sandhu et al. [SDYT09] encoded 3D geometries using an implicit function named Signed Distance Function (SDF), also known as zero-level embedding function [RR12]. They used *Principal Component Analysis* (PCA) to model the shape variance in a low-dimensional space. A 3D geometry was then recovered through a silhouette-based optimization during the recall process, where the input image is segmented into foreground and background and the silhouette of the 3D geometry is matched to the segmented foreground.

The major advantage to use SDF for describing the object geometry over explicit geometry description or other implicit methods is, that the normal vectors can be easily calculated for further uses, e.g., shading. Also, there exists a fast approach to render the 3D geometry from an SDF [LC87]. Recently, Prisacariu et al. [PSR12] made two significant improvements based on [SDYT09]. First, they transformed SDFs to the frequency domain using Discrete Cosine Transform (DCT) in order to make it feature rich. Second, they trained an extremely low-dimensional latent space through Gaussian Process Latent Variable Model (GPLVM) [Law05], which embedded all variant shapes within the same object class, e.g., *car* or *human body*. In other words, the search space for the optimal shape is constrained within a low-dimensional latent space, which enables highly efficient recall of the 3D shape.

We find the GPLVM representation of 3D object geometry in [PSR12] highly relevant to the key problem of object-level 3D reconstruction for in-vehicle AR, since it enables representing 3D object shapes using only two additional latent parameters. However, the major drawback of the approach in [PSR12] is, that the algorithm used for differentiating foreground and background in an input image is only trained based on a few frames (5 to 7 frames as described in [PSR12]). Besides, for tracking purpose, the first frame still needs to be initialized manually. We believe that using deep-learning-based methods, the performance of foreground-background segmentation can be boosted, and the initialization of object pose can be carried out using the similar neural network architecture for image segmentation. Hence, we decide to combine the approach in [PSR12] with state-of-the-art deep neural networks for in-vehicle augmented reality in our work.

Deep Learning Today, the computer vision community is experiencing a boom in deep learning. Deep-learning-based algorithms are applied in a wide range for object detection [KSH12, GDDM14, Gir15, RHGS15, RDGF16], semantic segmentation [HAGM14, LSD15, ZJRP$^+$15], viewpoint estimation [BHL15, PSG$^+$15, SQLG15, TM15], depth estimation [EF15, LSL15], and even point cloud classification [WSK$^+$15]. Early uses of Convolutional Neural Networks (CNN) for image classification [LBD$^+$89] and object recognition [LBBH98] already date back to the 1990s. However, due to the high computational cost that could hardly be covered by a CPU back then, neural networks first went through their dark decades until the concept of General Purpose computing on GPU (GPGPU) became practical. For more details about deep learning and its history, please refer to [Sch15].

Specifically for semantic segmentation, Long et al. [LSD15] introduced a novel network architecture named Fully Convolutional Network (FCN). Every pixel of the input image

is classified, yielding an output with the same size as the input. Nevertheless, FCN had an issue of non-sharp boundaries in the output which are cause by the max-pooling layers. Several studies [CPK+14, HAGM15, ZJRP+15, LSvdHR16] attempted to integrate Conditional Random Fields (CRFs) into an FCN in order to solve this issue. Among them, Zheng et al. [ZJRP+15] were able to fully integrate a CRF into a CNN-training process by breaking down the CRF minimization into a Recurrent Neural Network (RNN). For the reason that the approach in [ZJRP+15] was on the top rank of the PASCAL VOC 2012 image segmentation benchmark [EEv+15] back then in 2015, we adopted it in our monocular 3D Shaping workflow presented in Section 4.3.

Recent works [BHL15, PSG+15, SQLG15, TM15] also showed the applicability of deep learning in viewpoint estimation. Tulsiani et al. [TM15] and Su et al. [SQLG15] modeled viewpoint estimation as a multi-class classification problem. A viewpoint angle was discretized into viewpoint bins, and a cross-entropy loss function was used for training the neural network. Pepik et al. [PSG+15] and Beyer et al. [BHL15] showed however that it is a more natural way to model viewpoint estimation as a regression problem. The latter introduced Biternion Net which was capable of regressing fine-grained orientation angles. We believe both variants of modeling viewpoint estimation could be relevant, and we apply both for monocular 3D Shaping (Section 4.3) in order to investigate their influences on viewpoint estimation in traffic scenes. Based on our experience with the monocular 3D Shaping, we adapt the Biternion representation for further improvement (Section 4.4).

Object Detection So far, the introduced related work for 3D reconstruction and viewpoint estimation only works for single object, i.e., an input image only contains a single object. In a real traffic scene, however, this is rarely the case. Hence, we need to first find objects in the input image, define a Region Of Interest (ROI) around each object (typically a rectangular bounding box), and crop the image patches out of the ROI so that each image patch contains only one object. This procedure is referred to as object detection.

A typical preprocessing step for object detection is known as *bounding box proposal*, also referred to as *region proposal*, where candidate ROIs that are more likely to contain target objects are generated. Objectness [ADF12], Multi-scale Combinatorial Grouping (MCG) [APTB+14], Constrained Parametric Min-Cut (CPMC) [CS12], and Selective Search (SS) [UvdSGS13] are several well-known proposal methods. Please refer to [HBDS16] for more details of region proposal.

Recent works [LFU13, GGAM14, CKZ+15] used 3D sensors such as an RGB-D camera or a stereo camera to improve the performance of region proposal. Lin et al. [LFU13] extended the CPMC framework to 3D in order to segment and understand indoor scenes from RGB-D data. Gupta et al. [GGAM14] proposed an integrated system for scene understanding through RGB-D images by augmenting the MCG framework with depth information. Chen et al. [CKZ+15] used a stereo camera to generate 3D object proposals in common traffic scenes. Improvements in region proposal and object detection through 3D sensors are demonstrated in these publications.

Before the AlexNet [KSH12] was introduced, the most successful approaches for object detection used hand-crafted features [DT05] and Support Vector Machine (SVM)-based classifiers [FGMR10, PGSS12, DABP14, PSGS15]. Today, object detection is also dominated by deep-learning-based approaches. Compared to hand-designed features, the major advantage of a deep neural network lies above all in its learning capacity, which enables coping with highest complex models and object representations. In 2014, Girshick et al. [GDDM14] published Region-based CNN (R-CNN), which set a groundbreaking record in object detection on Pascal VOC 2012 challenge [EEv+15]. A number of improvements such as Fast R-CNN [Gir15] and Faster R-CNN [RHGS15] were proposed subsequently attempting to improve the runtime performance. The top performing algorithms in object detection on the KITTI vision benchmark [GLU12] back then in 2016 include Subcategory-aware CNN (SubCNN) [XCLS17] and the afore-mentioned 3D Object Proposals (3DOP) [CKZ+15] that made use of Lidar proposals.

In order to enable 3D Shaping in real traffic scenes, we propose a novel neural network architecture, namely Pose-RCNN, for jointly detecting objects and estimating viewpoints. The methods proposed in [DABP14, CKZ+15, HOBS15, PSGS15, XCLS17] are comprehensively compared to our proposed Pose-RCNN in Section 4.4.

4.3 Monocular 3D Shape Reconstruction

We first attempt to reconstruct the complete 3D geometry of an object only using one frame from one monocular camera. The purpose of approaching this challenging task for in-vehicle augmented reality is to prove the concept that object-level 3D reconstruction is feasible with a minimum sensor setup, e.g., a monocular camera. This proof of concept is highly important for the cost-sensitive automotive industry.

As already mentioned in Section 4.2, we find the latent variable representation of 3D geometries in [PSR12] highly relevant to the key problem of object-level 3D reconstruction we intended to solve for in-vehicle AR, and we believe the performance of this reconstruction approach could be boosted through deep neural network. This motivated us to design a workflow that exploits the strength of both deep neural networks and latent shape representation, which we refer to as monocular *3D Shaping*.

The entire workflow of monocular 3D Shaping is illustrated in Fig. 4.1. It comprises a first stage of 2D appearance detection and a second stage of viewpoint and shape optimization. The objective here is to find the correct viewpoint angle and 3D geometry of an object in an input image. In the first stage, the input image is semantically segmented into foreground and background through a segmentation network (SegNet). In addition, an initial viewpoint angle is estimated through a viewpoint network (VPNet). The foreground pixels are used as an image cue for the 3D shape optimization in the second stage, where the 3D pose and 3D shape of the object are jointly optimized.

The rest of this section is structured as follows. We introduce the latent shape space in Section 4.3.1 and discuss the 3D shape optimization in Section 4.3.2. In Section 4.3.3,

we explain the SegNet and VPNet in detail. In Section 4.3.4, we present the experiment results of monocular 3D Shaping.

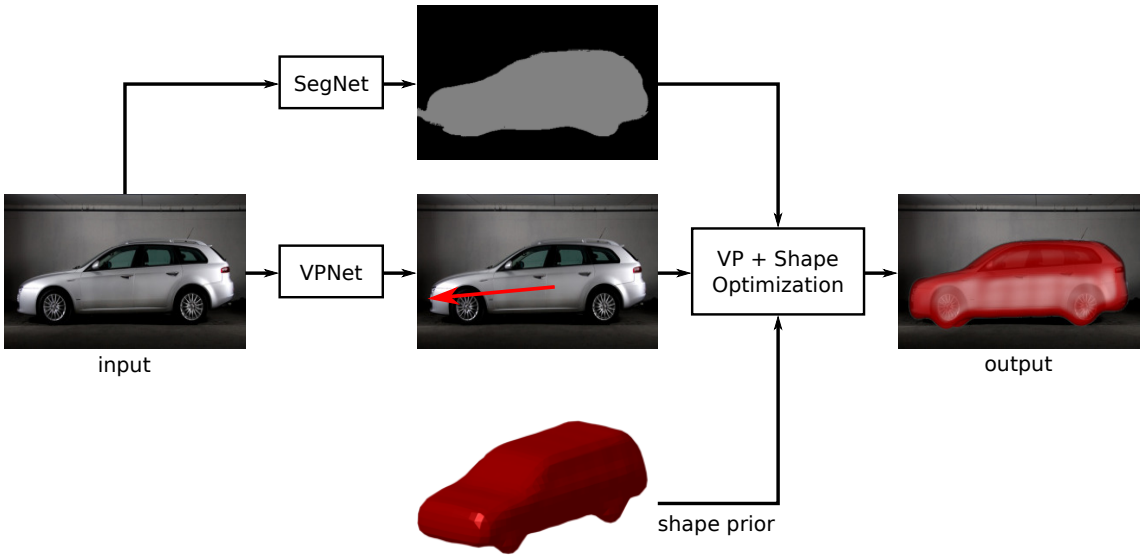


Figure 4.1: Workflow of monocular 3D Shaping.

4.3.1 Latent Shape

3D geometries are represented implicitly through Signed Distance Function (SDF). The general definition of an SDF is expressed in Eq. 4.1, where Ω denotes a subset of a metric space with $d(\cdot,\cdot)$ being the metric function. $\partial\Omega$ and Ω^c denote the boundary and the complement set of Ω, respectively. The definition of the metric function d is expressed in Eq. 4.2, which returns "the nearest distance" between x and the boundary of Ω. The SDF is positive if x is "inside" Ω and negative if outside. Theoretically, an SDF is differentiable almost everywhere if Ω is in the Euclidean space.

$$\Phi(x) = \begin{cases} d(x, \partial\Omega), & \text{if} \quad x \in \Omega \\ 0, & \text{if} \quad x \in \partial\Omega \\ -d(x, \partial\Omega), & \text{if} \quad x \in \Omega^c \end{cases} \tag{4.1}$$

$$d(x, \partial\Omega) = \inf_{x' \in \partial\Omega} d(x, x') \tag{4.2}$$

Figure 4.2 shows an example of a signed distance function in $\mathbb{R}^2$. One would immediately realize that this SDF represents the silhouette of a car. The boundary of the car is implicitly embedded in the contour line that equals zero, also known as *zero level set*.

A two-dimensional SDF is capable of representing the 2D boundary of an object in an image. By adding one dimension, the surface of a 3D geometry can be represented. In

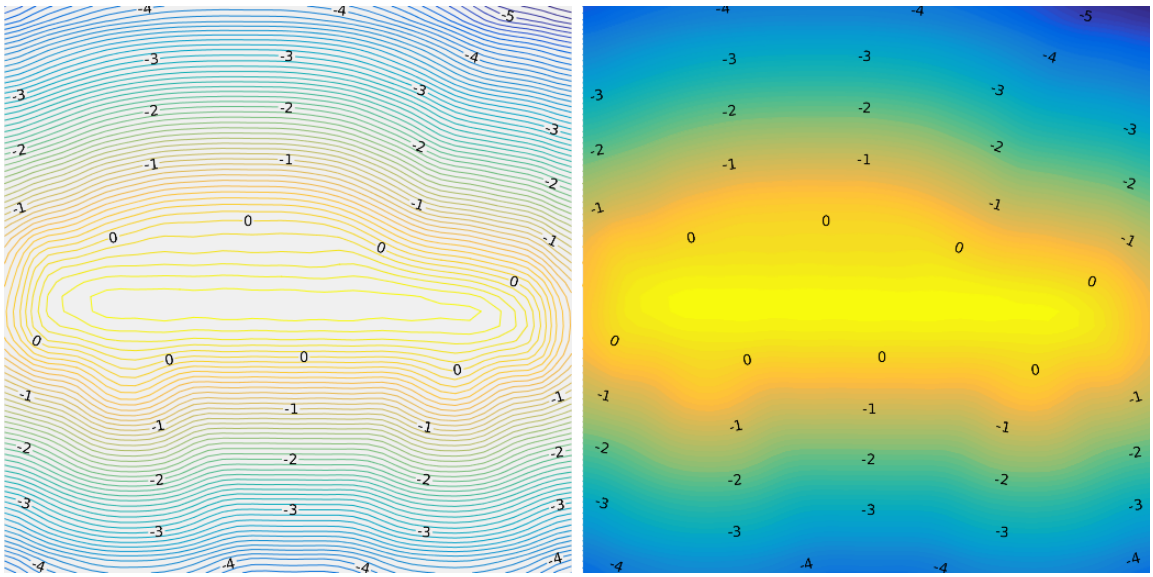


Figure 4.2: Visualization of a two-dimensional SDF.

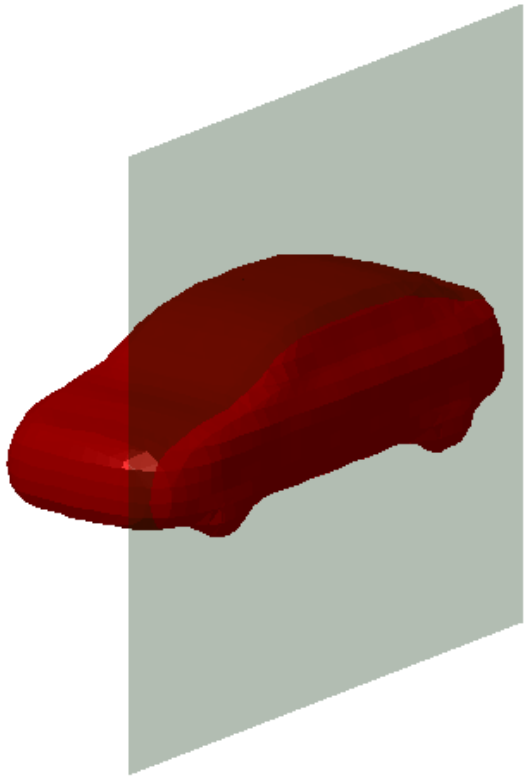
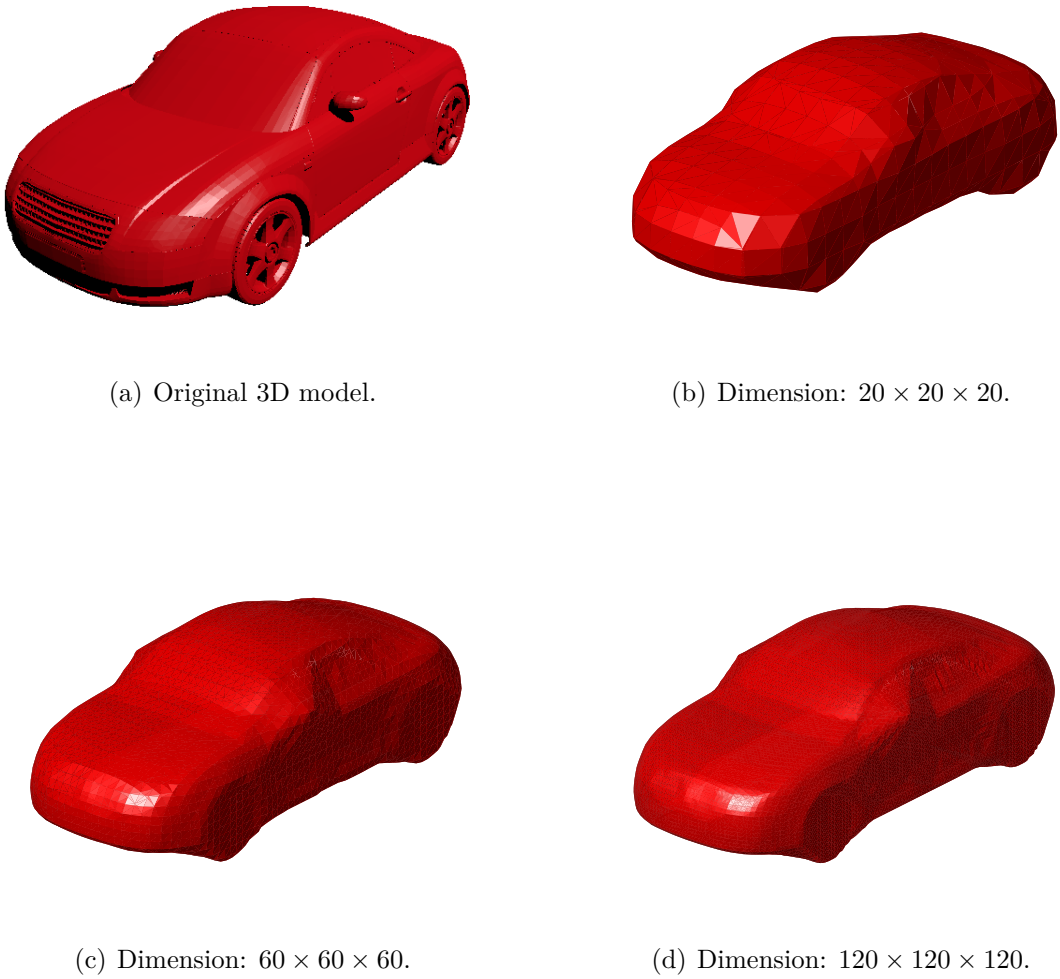


Figure 4.3: Visualization of a three-dimensional SDF.

fact, the previous example in Fig. 4.2 is just one slice from a 3D SDF that encodes the 3D geometry of a car, as shown in Fig. 4.3.

In practice, an SDF is discretized and stored in a floating point array. Each element of the array records the distance from its grid location to the zero level. The higher the dimension of the array is, the finer the original geometry is discretized. Figure 4.4 illustrates how the fineness of an SDF is affected by the dimension of the array container.



(a) Original 3D model.

(b) Dimension: $20 \times 20 \times 20$.

(c) Dimension: $60 \times 60 \times 60$.

(d) Dimension: $120 \times 120 \times 120$.

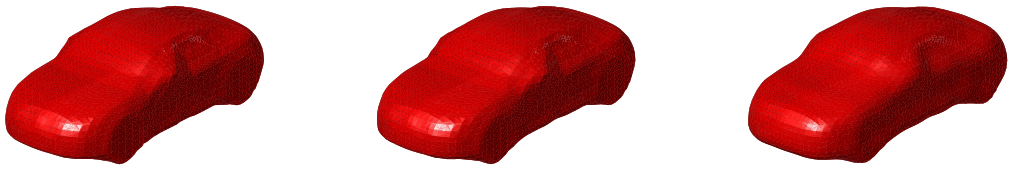
Figure 4.4: Different levels of fineness of an SDF.

Although an SDF is able to implicitly embed a 3D geometry, it cannot be directly used as a feature for classification or other machine-learning-based algorithms. The reason is that an SDF is a representation of the original 3D geometry in the spatial domain. A machine learning algorithm typically needs to operate in a so-called feature domain,

where sample points that belong to different classes are clearly separated. The same applies why we would not directly use RGB value as a feature for object recognition in an image. Therefore, a three-dimensional Discrete Cosine Transform (Eq. 4.3) is applied, which transforms an SDF into the frequency domain in order to make it feature rich.

$$X_{k_1,k_2,k_3} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \sum_{n_3=0}^{N_3-1} x_{n_1,n_2,n_3} \cos\left[\frac{\pi}{N}\left(n_1+\frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N}\left(n_2+\frac{1}{2}\right)k_2\right] \cos\left[\frac{\pi}{N}\left(n_3+\frac{1}{2}\right)k_3\right] \tag{4.3}$$

The number of DCT coefficients is the same as the number of the elements in a discretized SDF. Their sizes increase cubically with the dimension of the array. Dealing with an extremely high dimensional feature would require more computational power and slow down the entire workflow. However, if we cut the dimension of an SDF array, the quality of 3D reconstruction needs to be compromised. Here, we take advantage of the property of DCT which centralizes low-frequency components, and we simply discard small high-frequency coefficients. Figure 4.5 shows how the quality of reconstruction is affected by the number of DCT coefficients. The difference between the reconstructions using 60^3 coefficients (Fig. 4.5(a)) and 40^3 coefficients (Fig. 4.5(b)) is already barely distinguished through naked eyes.



(a) Coefficients: $60 \times 60 \times 60$. (b) Coefficients: $40 \times 40 \times 40$. (c) Coefficients: $20 \times 20 \times 20$.

Figure 4.5: Reconstruction with different numbers of DCT coefficients.

After collecting a small number of 3D geometries (training samples) and transforming them to DCT coefficients (features), a process called Gaussian Process Latent Variable Model (GPLVM) training is carried out which embeds the high-dimensional features into a low-dimensional latent space. GPLVM has the following advantages in the context of 3D shape representation. First, it only requires a small number of training samples, taking advantage of the nature of a Gaussian process. Second, it is able to significantly reduce the dimension of the feature space without any information loss. Last but not least, the low-dimensional latent space is continuous and differentiable everywhere, which is mathematically convenient by optimization.

Mathematically, GPLVM training aims at finding a Gaussian process (Eq. 4.4) that maps a low-dimensional latent space X to a high dimensional feature space Y. K denotes

a kernelized covariance matrix of the Gaussian distribution, with each element $k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

$$Y \sim \mathcal{N}(\mathbf{0}, K) \tag{4.4}$$

The kernel function $\kappa(\cdot, \cdot)$ is typically a radial basis function plus some white noise, as expressed in Eq. 4.5.

$$\kappa(\mathbf{x}, \mathbf{x}') = \theta_1 \exp\left(-\frac{\theta_2}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right) + \theta_3 + \theta_4 \delta(\mathbf{x}, \mathbf{x}') \tag{4.5}$$

The so-called hyper-parameters $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$ and the set of latent variables X are the variables to be optimized during GPLVM training. The optimization is formulated as a Maximum Likelihood Estimation (MLE) problem in Eq. 4.6. In other words, during GPLVM training, we try to find the latent variables X that are most likely to generate the given training set Y through a Gaussian process controlled by the hyper-parameters Θ.

$$(X^*; \Theta^*) = \arg\max_{X;\Theta} p(Y|X;\Theta) \tag{4.6}$$

In practice, the negative log-likelihood $L = -\ln p(Y|X;\Theta)$ is minimized through a Scaled Conjugate Gradient (SCG) method. Figure 4.6 shows a latent space that embeds 3D geometries of cars, with an extremely low latent dimension $q = 2$. The transition of 3D

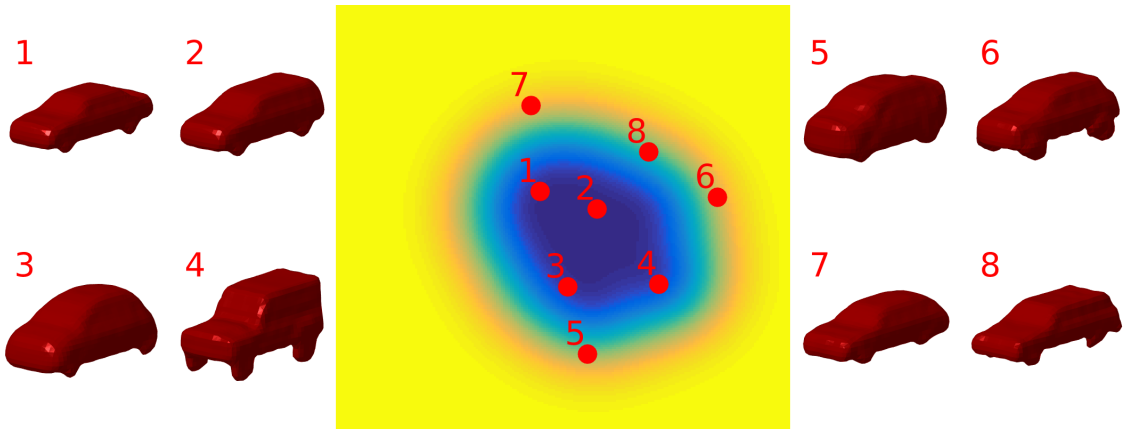


Figure 4.6: A two-dimensional latent space that embeds 3D geometries of cars. Latent points in the dark area are considered to be able to generate more car-like 3D shapes.

geometries inside the latent shape space is smooth everywhere, as shown in Fig. 4.7.

Having a trained latent space, one can infer a feature point $\mathbf{y}^*$ of an arbitrary latent variable $\mathbf{x}^*$ by the definition of Gaussian process (Eq. 4.7), where $K^* = [\kappa(\mathbf{x}^*, \mathbf{x}_1), \kappa(\mathbf{x}^*, \mathbf{x}_2), \cdots]^T$ and $K^{**} = \kappa(\mathbf{x}^*, \mathbf{x}^*)$.

$$\left[\begin{array}{c} Y \\ \hline \mathbf{y}^* \end{array}\right] \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{c|c} K & K^* \\ \hline (K^*)^T & K^{**} \end{array}\right]\right) \tag{4.7}$$

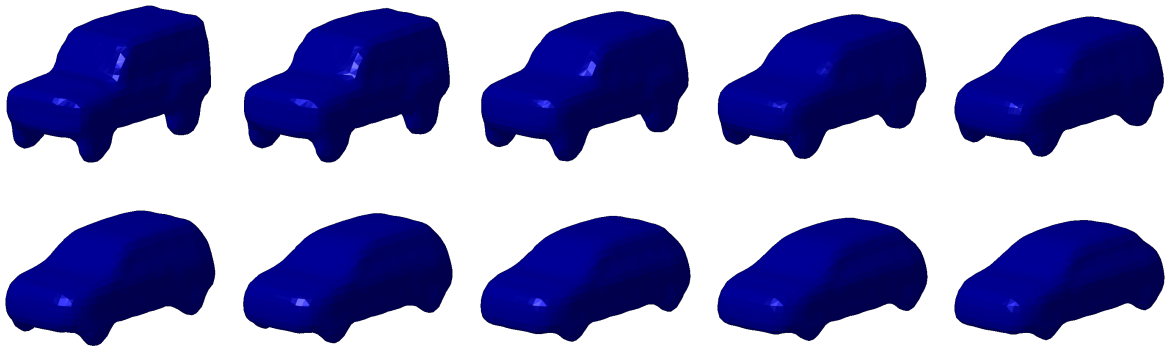


Figure 4.7: A smooth transition of latent shapes.

Therefore, the expectation and the variance of $\mathbf{y}^*$ can be estimated through Eq. 4.8.

$$\begin{aligned} E(\mathbf{y}^*) &= \kappa(\mathbf{x}^*, X)K^{-1}Y \\ Var(\mathbf{y}^*) &= \kappa(\mathbf{x}^*, \mathbf{x}^*) - \kappa(\mathbf{x}^*, X)^T K^{-1}\kappa(\mathbf{x}^*, X) \end{aligned} \tag{4.8}$$

The inference from a low-dimensional latent variable back to the original feature space is referred to as GPLVM recall.

Now, the path between a 3D geometry and a latent variable is completed, as illustrated in Fig. 4.8. 3D geometries go through distance transformation, DCT, and GPLVM training to become latent variables in the latent shape space. A latent variable generates a 3D SDF through GPLVM recall and Inverse DCT (IDCT). The triangulation of a 3D SDF is realized using the Marching Cube [LC87] algorithm.

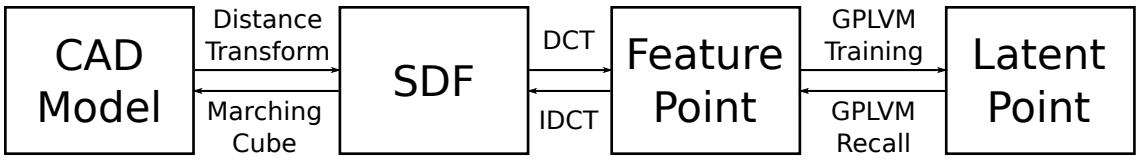


Figure 4.8: From latent variable to 3D geometry.

More details of latent shape training are given in Appendix A.

4.3.2 Optimization Problem

Figure 4.9 demonstrates how 2D-3D shape optimization works. The screen at the left side shows a semantically segmented input image, with blue pixels illustrating a car. At the right side, a 3D geometry of a car is projected onto the screen. The objective of the optimization is to find the best position, orientation, and 3D shape of the car so that the silhouette of the car on the screen matches the segmented output as close as possible.

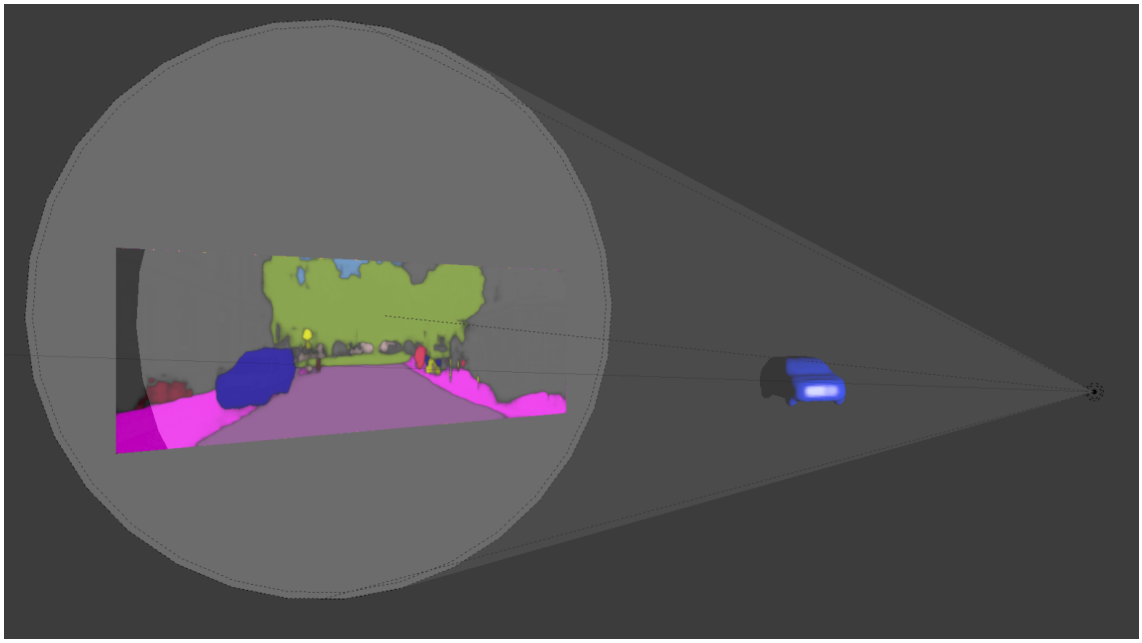


Figure 4.9: Visualization of 3D shape optimization.

Mathematically, the energy function in Eq. 4.9 is maximized during 3D shape optimization. Ω denotes a region of interest in an image and $\mathbf{x}$ an image pixel. $P_f(\cdot)$ and $P_b(\cdot)$ record how likely a pixel belongs to the foreground (the target object) or the background, respectively. They are estimated using a segmentation neural network (SegNet). $\pi(\cdot;\cdot)$ denotes a projection function that projects a 3D SDF denoted as Φ onto the image plane. The projection is controlled by the parameter set ρ. More specifically, ρ comprises three parameters for the translation, four parameters[2)] for the rotation, one parameter for the scale, and two additional latent variables for the 3D geometry. Relying on the latent shape representation, we only need to solve two additional variables in the latent space instead of, e.g., $40^3 = 64\,000$ variables in the original feature space.

$$E(\Phi;\rho) = \sum_{\mathbf{x}\in\Omega} P_f(\mathbf{x})\pi(\Phi;\rho) + P_b(\mathbf{x})[1-\pi(\Phi;\rho)] \tag{4.9}$$

The energy function is differentiable with respect to the latent variable and the viewpoint parameters. Thus, it can be optimized through standard non-linear optimization methods, such as gradient descent or Levenberg-Marquardt [Mor78]. More details of the derivatives of the energy function are given in Appendix B.

We conducted different experiments under controlled conditions in order to examine the influence of latent variables and viewpoint parameters on the energy function. First, we render a dummy image of a car from a fixed viewpoint and calculate the energy value at each point in the latent shape space. The result is visualized in Fig. 4.10, which clearly shows an area of global maxima that can be reached through gradient-based optimization

[2)] Quaternion representation.

methods. Second, we fix the latent shape and loose the orientation angle to change the viewpoint. In other words, we circle the camera a around the car and calculate the energy at each angle. Figure 4.11 illustrates the result. We observe that there is a local maximum 180°-flipped from the global maximum, i.e., there is a 50 percent chance that the optimization converged at local maximum, if we randomly initialize the viewpoint at the beginning of the optimization. Last but not least, in order to show the ambiguity between depth and scale, we loose the longitudinal distance and the scale parameter while fixing the others. A ridge-like structure can be observed in Fig. 4.12 that shows the result. This is quite straightforward to understand, since a big car far away might look like the same in an image as a small car right in front of us. The ambiguity between the depth and the scale will not disappear if only one image is used during the optimization.

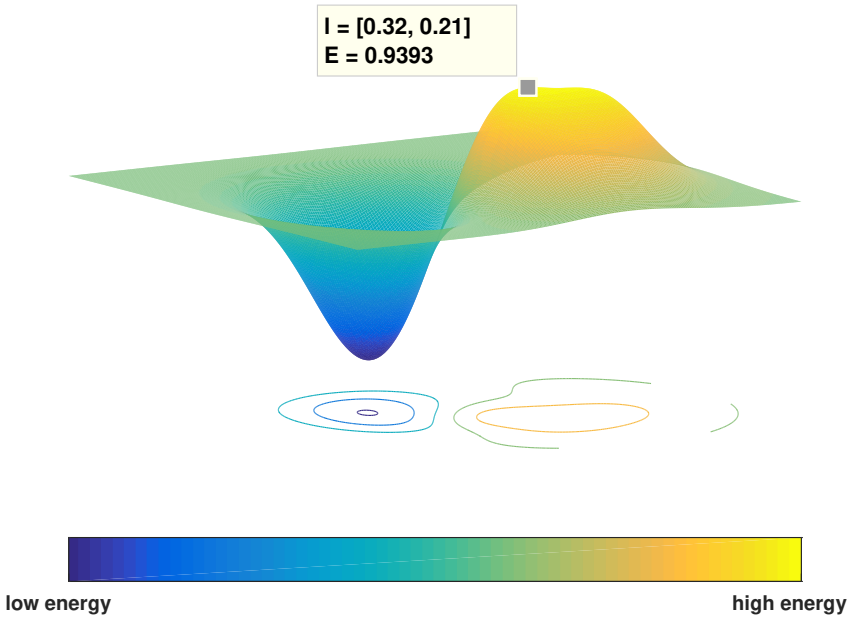


Figure 4.10: Energy by varying latent variable.

4.3.3 SegNet and VPNet

For monocular 3D Shaping, we assume the object bounding boxes are given in an earlier processing step, i.e., each input image contains only one object. For the SegNet, we use a Fully Convolutional Network (FCN) combined with a Conditional Random Field (CRF) as suggested in [ZJRP+15], for the reason that it produces sharp-edged foreground-background segmentation. The finer the segmentation results are, the better quality of 3D shape reconstruction can be achieved. In addition, we train a separate neural network specifically for viewpoint initialization. The network architecture is adapted

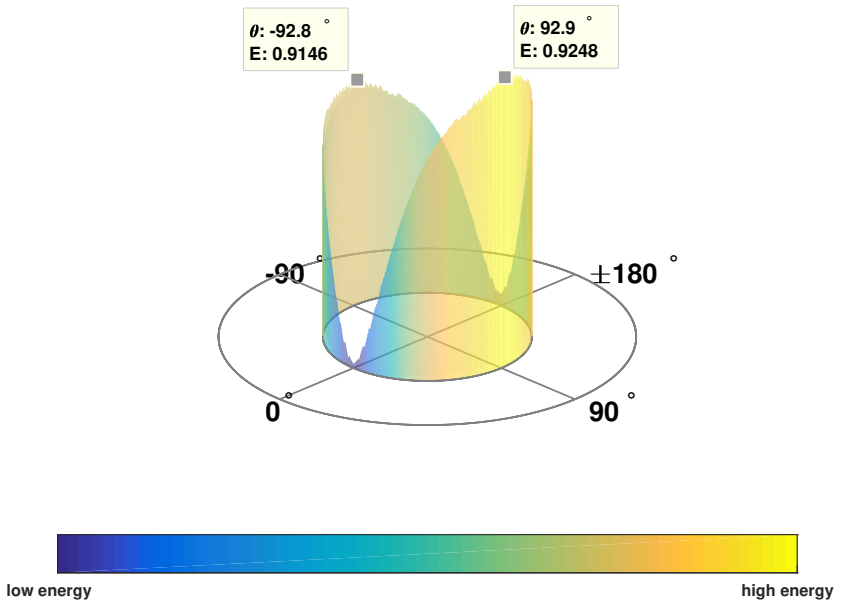


Figure 4.11: Energy by varying orientation angle.

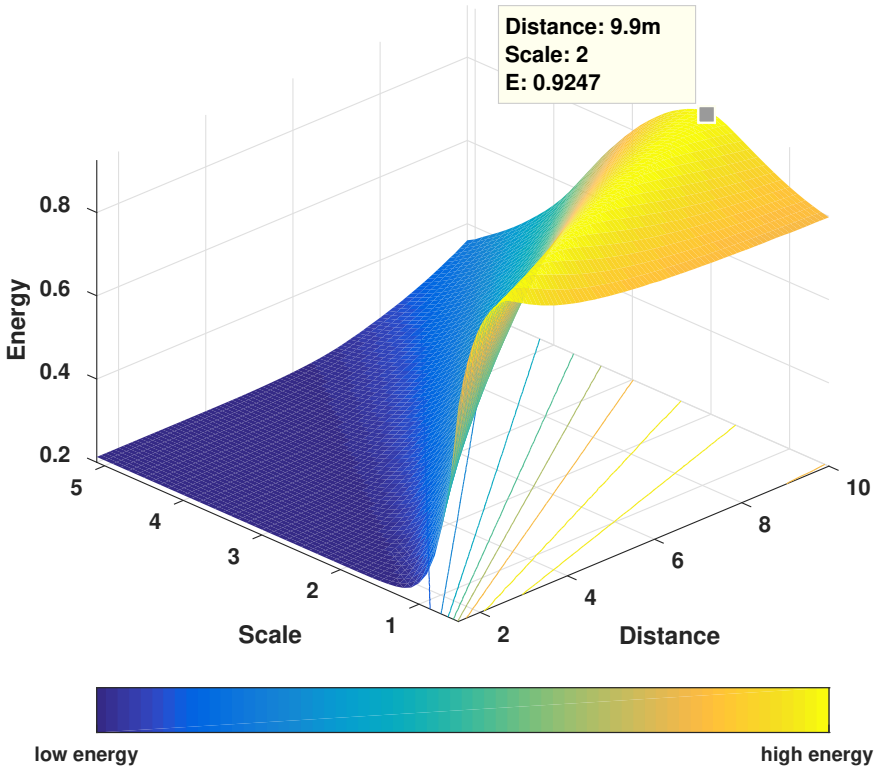


Figure 4.12: Energy by varying depth and scale.

from AlexNet [KSH12]. By modifying the last fully connected layer of the AlexNet, we are able to model the problem of viewpoint estimation differently from a mathematical point of view. Here, we use two different types of loss functions: the cross-entropy loss expressed in Eq. 4.10 and the Euclidean loss in Eq. 4.11. Using cross-entropy loss function, viewpoint estimation is formulated as a classification problem, whereas using Euclidean loss, it is defined as a regression problem.

$$L = -\frac{1}{N}\sum_{n=1}^{N} \ln\left(\frac{e^{x_{l_n}}}{\sum_{k=1}^{K} e^{x_k}}\right) \tag{4.10}$$

In Eq. 4.10, N denotes the number of training samples, K the total number of viewpoint bins, l the groundtruth viewpoint label, and x the predicted score. We tried different numbers of viewpoint bins. Generally, using a larger K increases the precision of the estimated viewpoint angle.

$$L = \frac{1}{2N}\sum_{n=1}^{N} \|\hat{x}_n - x_n\|^2 \tag{4.11}$$

The Euclidean loss in Eq. 4.11 turns the neural network into a regressor. $\hat{x}_n$ denotes the groundtruth and x_n the predicted viewpoint, respectively. The Euclidean distance between the groundtruth and the output of the neural network is minimized during the training process.

We used a subset of ImageNet with viewpoint annotations [XMS14] for training the VP-Net. A layer-separation for different classes is not considered since for monocular 3D Shaping, we specifically focus on *car* objects. All training processes are carried out using the Caffe deep learning framework [JSD+14].

4.3.4 Experimental Results

We use viewpoint accuracy as the quantitative measure to evaluate the performance of 3D shape reconstruction. The viewpoint error is measured by the geodesic distance over the rotation sphere, as expressed in Eq. 4.12, where R_{est} denotes the estimated rotation matrix and R_{gt} the groundtruth. We calculate the median viewpoint error $Med(\bullet)$ over the entire evaluation dataset. In addition, we show the percentage of accurate estimates $Acc(< \theta)$, with θ being the angular threshold. The results are presented in Tab. 4.1.

$$\delta_{vp}(R_{est}, R_{gt}) = \frac{1}{\sqrt{2}}\|\ln(R_{est}^T R_{gt})\|_F \tag{4.12}$$

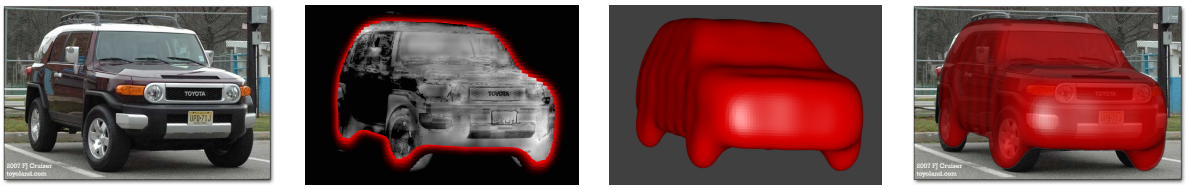
The first half of Tab. 4.1 shows results after the first stage of monocular 3D Shaping and the second half after the second stage, 2D-3D optimization. The network using cross-entropy loss for viewpoint estimation are denoted by VPNet-Kbin, with K indicating the number of viewpoint bins. The viewpoint regression network using Euclidean loss function is denoted by VPNet-Reg. VDPM-24bin stands for Viewpoint Deformable Part Model [XMS14] with 24 viewpoint bins, which was considered the state-of-the-art before deep-learning-based algorithms were introduced.

Table 4.1: Evaluation of viewpoint estimation by monocular 3D Shaping.

Approach	$Med(\delta_{vp})$	$Acc(30°)$	$Acc(15°)$	$Acc(5°)$
VDPM-24bin	19.60°	67.51%	35.57%	8.16%
VPNet-Reg	23.11°	58.74%	34.75%	12.34%
VPNet-24bin	11.82°	86.77%	64.29%	19.26%
VPNet-72bin	7.48°	**90.47%**	73.61%	35.23%
VPNet-24bin+Shape	5.47°	88.04%	**78.35%**	47.08%
VPNet-72bin+Shape	**4.29°**	89.13%	77.30%	**54.13%**

The following three points can be summarized from the results in Tab. 4.1. First, the viewpoint network with the most viewpoint bins outperforms the other approaches. Second, the estimated viewpoint angles are more close to the groundtruth after the second stage optimization. There is a nearly 20 percent performance gain in $Acc(< 5°)$ by VPNet-72bin+Shape. Last but not least, the optimization process is proven to be robust against initialization to some extent, as VPNet-24bin+Shape and VPNet-72bin+Shape yield similar results.

Figure 4.13 shows a reconstructed 3D car overlaid on the image, as an example use case of 3D Shaping for in-vehicle augmented reality. The foreground possibilities estimated through the segmentation network is illustrated in Fig. 4.13(b), ranging from zero (black) to one (white). The red frame in Fig. 4.13(b) shows the outer contour of the projected 3D geometry.

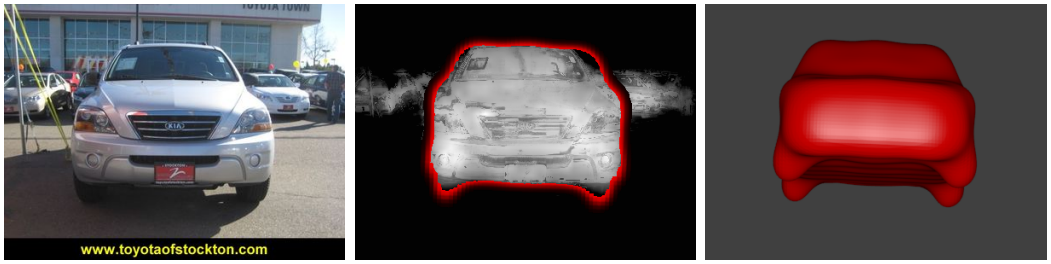


(a) Input image (b) Segmentation score (c) 3D shape recovered (d) 3D shape overlay

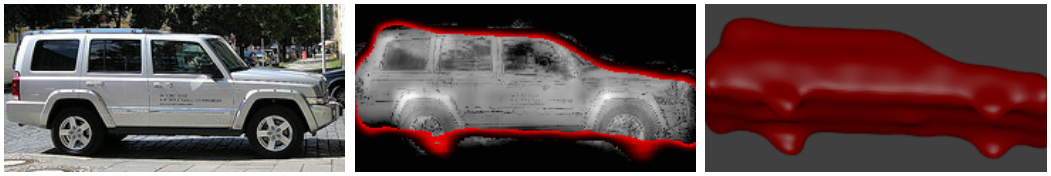
Figure 4.13: Example of satisfying 3D reconstruction.

Three failed reconstructions due to (a) ambiguous silhouette, (b) divergence in the latent shape space, and (c) bad viewpoint initialization are presented in Fig. 4.14. More reconstruction results of 3D Shaping are shown in Fig. 4.16.

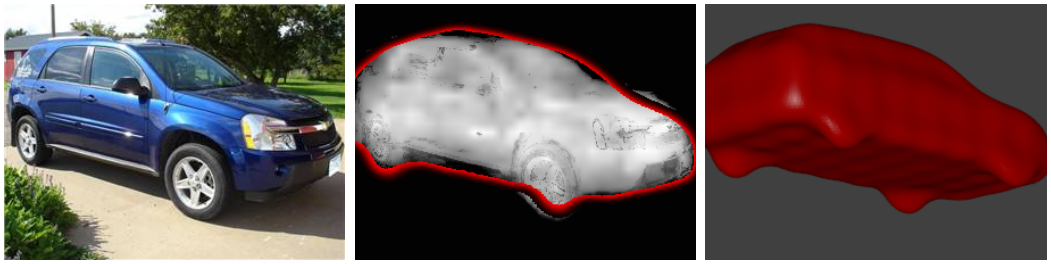
Certainly, the silhouette-based 2D-3D optimization is subject to viewpoint initialization and image segmentation. Additional constraints such as discriminant keypoints could be introduced to solve the problem of ambiguity shown in Fig. 4.14(a). For example, wheels or side mirrors could be used to identify the viewpoint. Another idea would be using shared latent space that associates latent shape variables and keypoints. However, compared to neural networks, it is not easy to parallelize these algorithms and efficiently implement them on an in-vehicle platform.



(a) Ambiguous silhouette.



(b) Divergence in the latent shape space.



(c) Bad viewpoint initialization.

Figure 4.14: Failed 3D reconstructions.

(a) Original image patch. (b) Segmented image. (c) Probability of car pixels.

Figure 4.15: Occlusion problem by monocular 3D Shaping. The optimizer would try to cover the entire white area in (c) with only one instance of the object class car.

Nevertheless, the workflow presented in this section is only designed for a single object standing in front of a clean background. Multiple instances of the same object class would cause a self-occlusion problem, as illustrated in Fig. 4.15. Without knowing the exact number of cars in the scene and boundaries between them, the optimizer is unable to deliver reasonable estimations of the viewpoint and 3D shapes.

Figure 4.16: More reconstruction results of monocular 3D Shaping. The last two rows present two failed reconstructions due to 180°-flipped viewpoint and strong reflection from the wheel, respectively.

4.4 Pose-RCNN

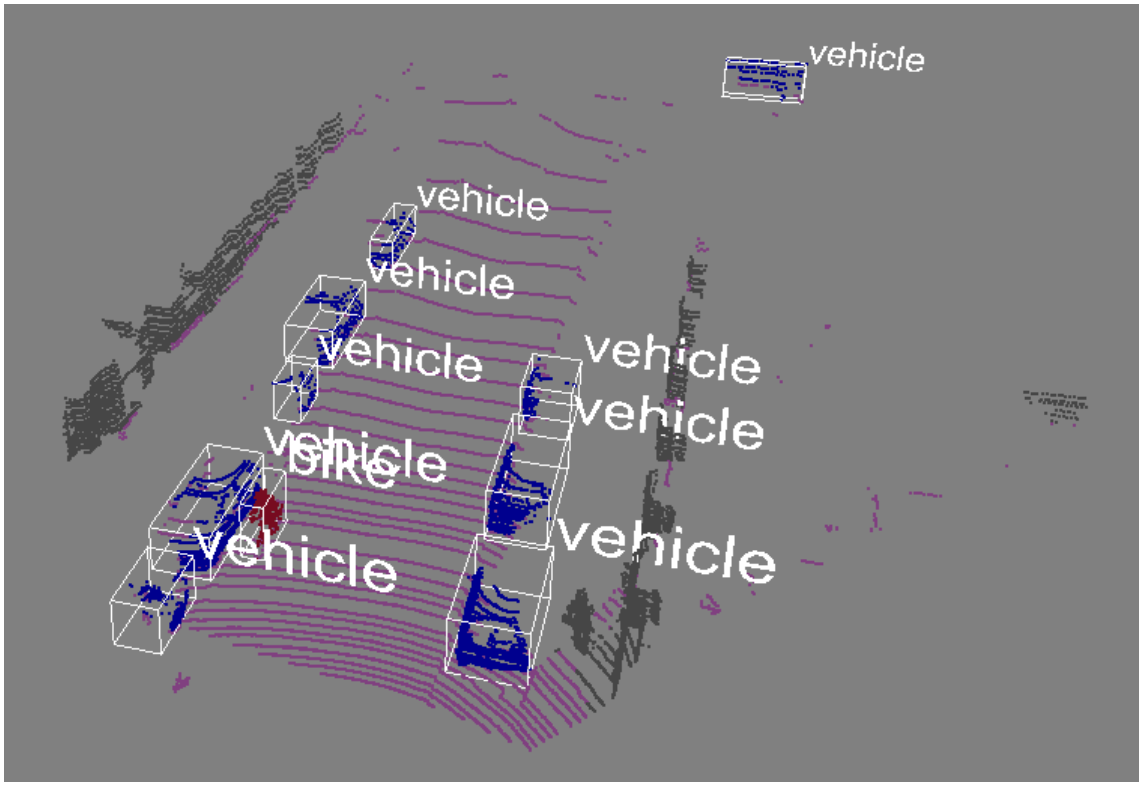
In this section, we introduce Pose-RCNN, a combined approach for object detection and viewpoint estimation. We intend to use Pose-RCNN as an extension for monocular 3D Shaping in order to make it more practical for real traffic scenes. Recall that we assume an input image to monocular 3D Shaping to only contain one object. For images containing multiple objects, we need to first detect the objects in the images. In addition, as we already shown in Section 4.3.4, the performance of the second stage optimization of 3D Shaping weakly depends on the initial guess of the viewpoint. We used a separate VPNet to estimate the viewpoint in the monocular 3D Shaping approach, and we believe the VPNet can be integrated into a detection network such as R-CNN. Therefore, we name the novel neural network architecture Pose-RCNN for it is able to detect object and estimate their pose jointly.

There are two major building blocks in our proposed Pose-RCNN workflow. The first building block is to generate ROI proposals that are more likely to contain target objects. Here, we propose two different methods for proposal generation, one of them takes Stixels as input, the other one uses point clouds captured by a Lidar. The second building block is a modified R-CNN that takes an image and the bounding box proposals generated from the first as input. It outputs an object class and a viewpoint angle *per proposal*. We cherry-pick the most trustworthy detection results of the proposed Pose-RCNN for 3D Shaping.

4.4.1 Proposal Generation

Lidar Proposal Generation In order to generate ROI proposals given an unorganized Lidar point cloud, we need to cluster the entire cloud into smaller clusters. A common approach to cluster point clouds is to remove the points on the ground in the first step and group the rest non-ground points together. Here, we use the implementation of the Point Cloud Library for clustering the point cloud. Progressive Morphological Filter (PMF) [ZCW+03] is used to estimate ground points, and k-dimensional tree is used for clustering non-ground points based on their Euclidean distances. At last, for each point cloud cluster, a 3D bounding box is generated and projected onto the input image, yielding 2D ROI proposals (Fig. 4.17). The projected 2D proposals are enriched again through spatial translation and scaling in order to increase the number of proposals.

The parameters of PMF and k-d-tree-based nearest neighbor clustering have considerable influence on the recall rate of Lidar proposals. Here, we pick up two different parameter sets for evaluation. Through the first set, denoted as $Li1$, we attempt to rigorously keep the false negative rate as low as possible, whereas we intend to increase the recall rate through the second set $Li2$ where the constraint of minimum size of a cluster is lifted. The detailed parameters of $Li1$ and $Li2$ are given in Tab. 4.2. Note that ground estimation using the second parameter set $Li2$ only takes short range Lidar points as input as a Lidar can hardly reach the ground beyond a certain distance. In other words, using $Li2$, points



(a) Proposals in 3D space.

(b) Proposals projected onto the image.

Figure 4.17: Lidar proposal generation.

Table 4.2: Detailed parameter settings of Lidar proposal.

Step	Parameter	*Li1*	*Li2*
Ground estimation	initial ground distance	0.2m	0.15m
	maximal ground distance	0.5m	0.15m
Euclidean clustering	cluster distance	0.3m	0.45m
	minimal number of points	50	10

in the far range are not removed through PMF in the first step. This allows us to catch potential objects in the far range and thus to increase the recall rate. The range threshold between "near" and "far" is set to be 20 meters.

Stereo Proposal Generation We also use Stixel (Fig. 4.18) to generate ROI proposals. As already introduced in Section 3.2, Stixel is a compact mid-level representation of depth information, which are generated based on the assumption that objects in a traffic scene roughly have a vertical "facade", such as pedestrians and cars. We use our *a priori* knowledge about the size of common traffic participants to turn Stixels into bounding box proposals. First, Stixels that are taller than 2.4m, shorter than 1.2m, farther away than 100m, or higher than 0.5m above the ground are removed, since the possibility of an existing pedestrian or car in this range is low. Then, the width of each remaining Stixel is adjusted by multiplying three different aspect ratios, including 0.5, 1, and 2. Here, we experiment two parameter sets as well, namely Stixel Proposal *SP* with 7-pixel width and Stixel Proposal Less-wide and Jittered *SPLJ* with 3-pixel width. For *SPLJ*, proposals are "jittered" by 10 percent to the left, right, top, and bottom, in order to increase the number of proposals.

4.4.2 Network Architecture of Pose-RCNN

The network architecture of the proposed Pose-RCNN is shown in Fig. 4.19. We attach a small viewpoint regression network on top of the ROI pooling layer of an R-CNN. The feature vectors at the output of the ROI pooling layer go through three parallel fully-connected layers, yielding object classes, bounding box offsets, and viewpoint angles. We use the von Mises loss function for regressing viewpoint angles. Against the cross-entropy loss and Euclidean loss function used in the VPNet (Section 4.3.3) for monocular 3D Shaping, the von Mises loss function has the following advantages. First, the von Mises distribution resembles a normal distribution around a circle, i.e., there is no discontinuity between $-180°$ and $180°$ in the output anymore. Second, the von Mises loss function is differentiable everywhere and thus mathematically convenient to be integrated into a neural network. These allow the network to interpret the input feature more correctly to the corresponding groundtruth angle, and to converge faster during a training process.

(a) Stixels generated by a stereo camera.

(b) Filtered Stixels.

(c) Generated proposals after adjusting the Stixel width.

Figure 4.18: Stereo proposal generation.

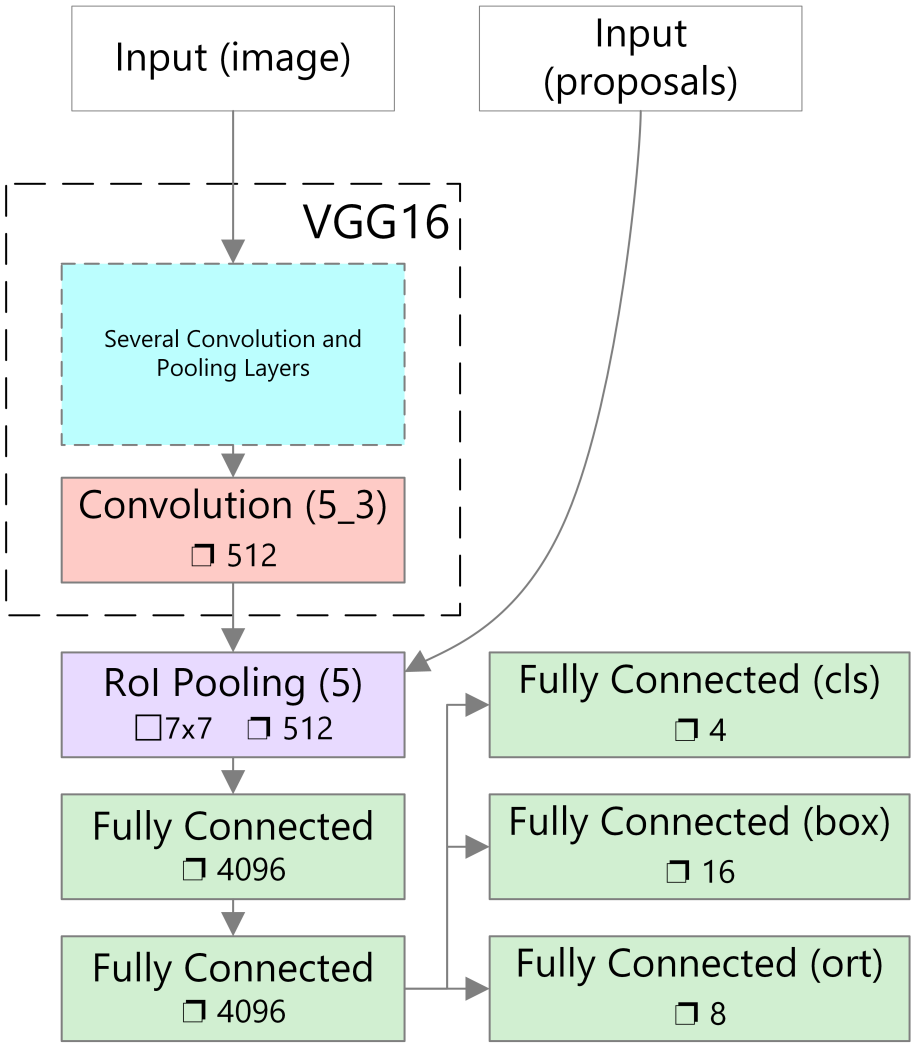


Figure 4.19: Network architecture of the proposed Pose-RCNN. ❒ denotes the dimension of a layer output and ☐ the size of the ROI pooling window. The outputs of the Pose-RCNN include (from top to bottom) an object class (cls), a bounding box (box), and an orientation angle (ort).

4.4.3 Experimental Results

We evaluate Pose-RCNN using the KITTI vision benchmark [GLU12] for object detection. A total of 80 256 labeled objects including cars, pedestrians, and cyclists in common traffic scenes are available in the training dataset of KITTI. We evaluate our proposed methods for proposal generation, namely *SP*, *SPLJ*, *Li1*, and *Li2*, as well as their combinations *SP-Li1*, *SPLJ-Li1*, and *SPLJ-Li2*. The methods are combined by taking the union of the results without any filtering of duplicates.

Table 4.3 and Tab. 4.4 present the Average Precision (AP) in object detection and the Average Orientation Similarity (AOS) in viewpoint estimation, respectively. We achieve the highest detection and orientation scores for easy scenarios. For other moderate and hard scenarios, our results are also competitive with the state-of-the-art object detection methods introduced in Section 4.2, including ACF [DABP14], R-CNN [HOBS15], DPM-VOC+VP [PSGS15], 3DOP [CKZ+15], and SubCNN [XCLS17]. The biggest advantage of our proposed Pose-RCNN is, even if its detection performance is a few percent less than the state-of-the-art approaches for certain scenarios, that it has a potential to be integrated into a real-time in-vehicle framework relying on its single feed-forward neural network architecture.

Table 4.3: Evaluation of Pose-RCNN in average precision (%).

Object	Approach	Difficulty		
		Easy	Moderate	Hard
Car	ACF [DABP14]	55.89	54.74	42.98
	R-CNN [HOBS15]	-	-	-
	DPM-VOC+VP [PSGS15]	74.95	64.71	48.76
	3DOP [CKZ+15]	**93.04**	88.64	79.10
	SubCNN [XCLS17]	90.81	**89.04**	**79.27**
	Proposed	88.43	75.80	66.57
Pedestrian	ACF [DABP14]	44.49	39.81	37.21
	R-CNN [HOBS15]	61.61	50.13	44.79
	DPM-VOC+VP [PSGS15]	59.48	44.86	40.37
	3DOP [CKZ+15]	81.78	67.47	64.70
	SubCNN [XCLS17]	**83.28**	**71.33**	**66.36**
	Proposed	77.53	63.40	57.49
Cyclist	ACF [DABP14]	-	-	-
	R-CNN [HOBS15]	-	-	-
	DPM-VOC+VP [PSGS15]	42.43	31.08	28.23
	3DOP [CKZ+15]	78.39	68.94	61.37
	SubCNN [XCLS17]	79.48	**71.06**	**62.68**
	Proposed	**80.79**	68.79	60.40

In fact, average orientation similarity as defined in [GLU12] is strongly correlated with the average precision. The average orientation similarity is the upper bound of the average

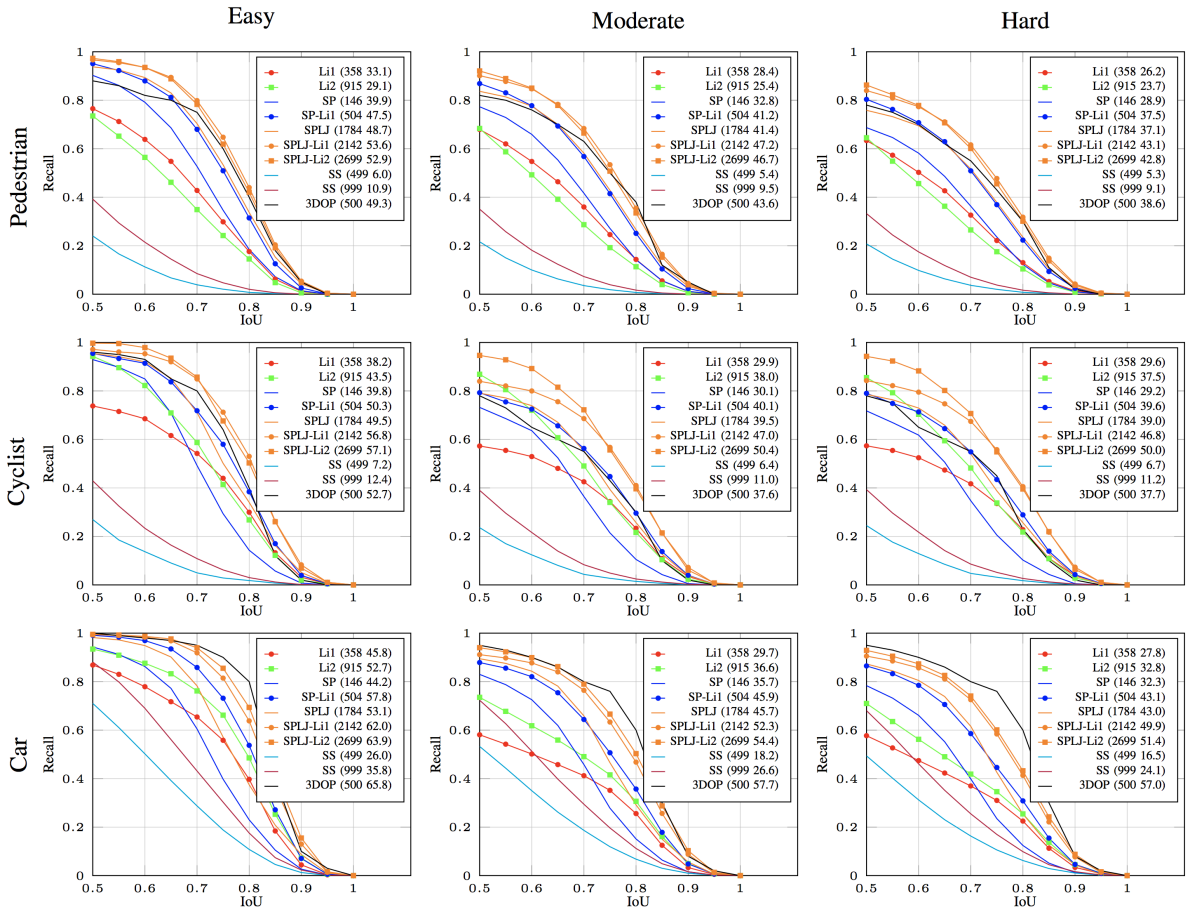


Figure 4.20: Recall – IoU curves of different proposal methods. The numbers inside the brackets show the average number of proposals and the average recall (%), respectively. The *3DOP* curve is sampled from the original paper [CKZ+15]. The curves of Selective Search (*SS*) are calculated based on an implementation of [UvdSGS13].

Table 4.4: Evaluation of Pose-RCNN in average orientation similarity (%).

Object	Approach	Difficulty		
		Easy	Moderate	Hard
Car	DPM-VOC+VP [PSGS15]	72.28	61.84	46.54
	3DOP [CKZ+15]	**91.44**	86.10	76.52
	SubCNN [XCLS17]	90.67	**88.62**	**78.68**
	Ours	88.34	75.41	66.07
Pedestrian	DPM-VOC+VP [PSGS15]	53.55	39.83	35.73
	3DOP [CKZ+15]	72.94	59.80	57.03
	SubCNN [XCLS17]	**78.45**	**66.28**	**61.36**
	Ours	73.95	59.90	54.27
Cyclist	DPM-VOC+VP [PSGS15]	30.52	23.17	21.58
	3DOP [CKZ+15]	70.13	58.68	52.35
	SubCNN [XCLS17]	72.00	**63.65**	**56.32**
	Ours	**75.49**	62.87	55.47

precision, i.e., the ratio $\beta = AOS/AP$ is always smaller than 1. For cars, our approach already achieves a similar β as 3DOP [CKZ+15] and SubCNN [XCLS17]. For cyclists and pedestrians, our β ratio is even higher than those by 3DOP and SubCNN. These evince a great potential of our proposed Pose-RCNN. Improving the average recall of the proposals would automatically boost the average orientation similarity score.

4.5 3D Shaping + Lidar

In this section, we introduce another extension of monocular 3D Shaping in order to make it more practical for real traffic scenes. Here, we take advantage of another in-vehicle 3D sensor, a Lidar, to reconstruct the 3D shape of multiple object instances in the same class. Hence, we name the extension *3D Shaping + Lidar*.

Recall that the monocular 3D Shaping proposed in Section 4.3 requires the input image only to contain one object. For multiple object in the same class, a problem of self-occlusion will occur, as already shown in Fig. 4.15. We aim at solving this problem by taking advantage of the additional Lidar.

The workflow of 3D Shaping + Lidar is similar to that of monocular 3D Shaping. It comprises the first stage of appearance detection and the second stage of shape optimization. In the first stage, we can take advantage of the proposed Pose-RCNN (Section 4.4) to cluster the scene in 3D and generate object proposals together with an estimated orientation angle for each detected object. Furthermore, we can use the clustered Lidar points as an additional cue for shape optimization. By directly using 3D measurements, the ambiguity between distance and scale as observed by monocular 3D Shaping (Fig. 4.12) can be eliminated.

In the rest of this section, we first explain the updated energy function for 3D Shaping + Lidar in Section 4.5.1 and present the evaluation results in Section 4.5.2.

4.5.1 Energy and Optimization

The new energy function is defined in Eq. 4.13, where a point cloud energy E_{cloud} is combined with the image-based energy E_{img} defined in Eq. 4.9. λ is the weight combining the two energies. Φ denotes an SDF that encodes 3D geometries, and ρ denotes a set of pose parameters. This energy function is first used by Dame et al. [DPRR13] for similar purposes.

$$E(\Phi;\rho) = E_{img}(\Phi;\rho) + \lambda E_{cloud}(\Phi;\rho) \tag{4.13}$$

The cloud energy is formulated in Eq. 4.14, where $\mathbf{X}^L$ denotes a 3D point in a Lidar cluster $\mathcal{L}$, and $g \in SE(3)$ the Lie algebra that transforms a point from Lidar coordinates to the object geometry coordinates. The object geometry coordinates system is attached at the geometrical centroid of an object. The exponential term of the Geman-McClure function [GM85] reaches its minimum if $\mathbf{X}^O$ exactly lies on the zero-level of the SDF Φ. It increases monotonically with the distance between $\mathbf{X}^O$ and the object surface, and the increasing rate is controlled by σ.

$$\begin{aligned} E_{cloud}(\Phi;\rho) &= \sum_{\mathbf{X}^L \in \mathcal{L}} \exp\left\{\frac{\Phi^2(g(\mathbf{X}^L;\rho))}{\Phi^2(g(\mathbf{X}^L;\rho)) + \sigma}\right\} \\ &= \sum_{\mathbf{X}^O} \exp\left\{\frac{\Phi^2(\mathbf{X}^O)}{\Phi^2(\mathbf{X}^O) + \sigma}\right\} \end{aligned} \tag{4.14}$$

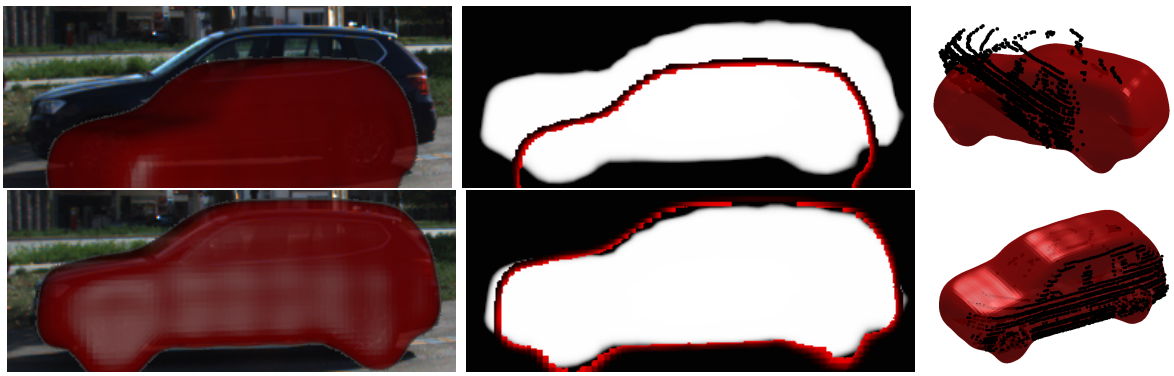


Figure 4.21: 3D Shaping optimization using image and Lidar measurements. The first row shows the initial state, and the second row the final state. Image statistics are illustrated in the middle column, with white pixels being more likely to be foreground pixels. The right column shows SDFs and Lidar measurements.

In order to examine the properties of the point cloud energy, we conduct similar experiments under controlled conditions as in Section 4.3.2. We generate a dummy point cloud cluster by random sampling the surface of an SDF of a car, and we use the dummy cluster

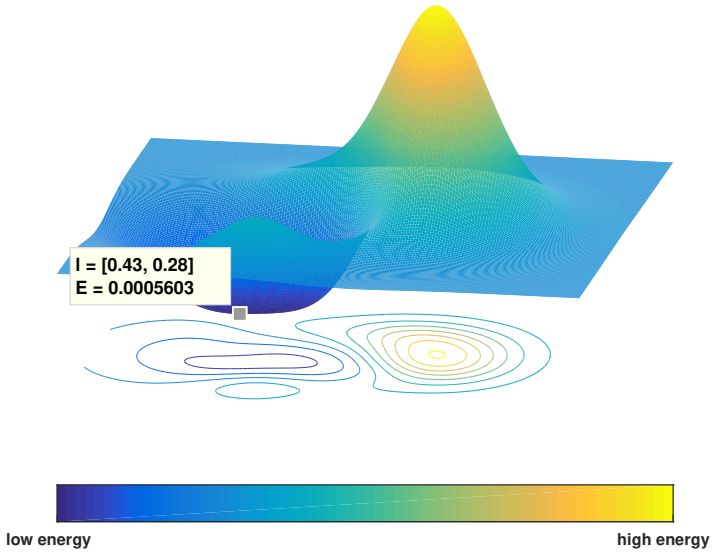


Figure 4.22: Cloud energy by varying latent variable.

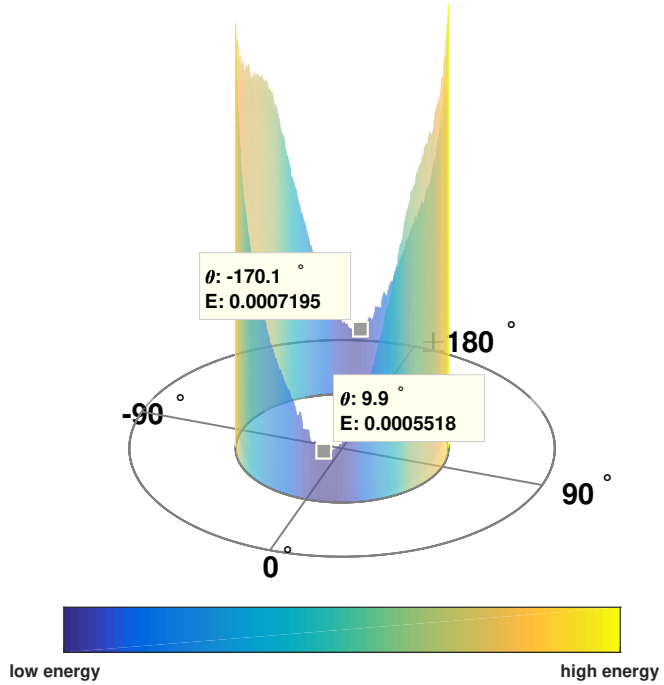


Figure 4.23: Cloud energy by varying orientation angle.

to simulate Lidar measurements. First, we fix the viewpoint and vary the latent variables within a trained latent shape space. Second, we fix the latent shape and alter the orientation angle. The results are visualized in Fig. 4.22 and Fig. 4.23, respectively. We observe similarities between the point cloud energy and the image-based energy. There is a unique global minimum by varying latent variables, while a 180°-flipped local minimum by varying orientation angle occurs.

In practice, the image energy and the point cloud energy can be jointly optimized through gradient-based optimization methods. Figure 4.21 shows an example to help understand the optimization process. Lidar points would "stick" to the surface if the optimization is ideally converged.

4.5.2 Experimental Results

We evaluate the combined energy function on a subset of the KITTI vision benchmark. Thirty-eight raw recording sequences with groundtruth 3D object labels are included in our evaluation dataset. In order to evaluate the performance of 3D reconstruction, we use the accuracy of orientation angle and the accuracy of occupancy bounding box as two measures.

We manually add a bias with 15° mean and 15° standard deviation to the groundtruth orientation, in order uncouple the evaluation of 3D reconstruction from viewpoint initialization. This bias is set to be larger than the average estimation error of the top ranking algorithms of the KITTI vision benchmark. The results are presented in Tab. 4.5 and Fig. 4.26. In Tab. 4.5, $Med(\delta)$ shows the median of the absolute estimation errors δ in degree, and $Acc(< \theta)$ indicates the percentage of accurate estimates that are smaller than the threshold angle θ. According to the results, both 3D Shaping approaches using or without using a Lidar are able to correct viewpoint angle within a certain initialization error. At short range, the approach using a Lidar is able to correct an error more than 30° and at full range up to 20°. This clearly shows the improvement by using a Lidar.

Table 4.5: Evaluation of 3D Shaping in viewpoint estimation.

Range	Approach	$Med(\delta)$	$Acc(< 20°)$	$Acc(< 10°)$	$Acc(< 5°)$
short ($\leq$ 20m)	Initialization	16.07°	61.23%	31.80%	15.15%
	Orientation Net	12.77°	63.00%	42.12%	23.61%
	Shaping Cam	13.96°	65.67%	37.95%	21.64%
	Shaping Cam+Lidar	**7.02°**	85.45%	67.53%	37.99%
full	Initialization	16.01°	62.08%	31.87%	15.49%
	Orientation Net	20.44°	49.42%	30.61%	16.57%
	Shaping Cam	16.08°	60.24%	32.59%	17.91%
	Shaping Cam+Lidar	14.14°	61.50%	38.44%	20.22%

The occupancy bounding box is defined as the rectangular bounding box around an object from a bird's-eye view. The bounding box overlap ratio is given by Eq. 4.15. We present the evaluation results of bounding box accuracy in Tab. 4.6 and Fig. 4.26. In Tab. 4.6, $Mean(Ovl)$ denotes the average of the overlap ratios. $Acc(> \bullet)$ indicates the percentage of accurate estimates that are larger than a certain threshold of overlap ratio. An estimated occupancy bounding box is considered correct if the overlap ratio is larger than 0.5. According to the results, the approach using a Lidar significantly improves the bounding box estimation performance compared to the monocular approach, due to the fact that the ambiguity between distance and scale is eliminated by directly using 3D measurements.

$$Ovl(A, B) = \frac{A \cap B}{A \cup B} \tag{4.15}$$

Table 4.6: Evaluation of 3D Shaping in bounding box estimation.

Range	Approach	$Mean(Ovl)$	$Acc(> 0.5)$	$Acc(> 0.7)$
short ($\leq$ 20m)	Shaping Cam	0.17	8.34%	1.97%
	Shaping Cam+Lidar	0.69	**90.57**%	62.77%
full	Shaping Cam	0.11	5.67%	1.21%
	Shaping Cam+Lidar	0.53	58.36%	28.87%

We also evaluate the absolute translation error and present the results in Tab. 4.7 and Fig. 4.26. In Tab. 4.7, Med and Std denote the median and the standard deviation of absolute translation errors, respectively. We observe a significant improvement in estimating longitudinal distance for 3D Shaping using a Lidar compared to monocular 3D Shaping.

Table 4.7: Evaluation of 3D Shaping in translation estimation.

Range	Approach	Med (m)		Std (m)	
		longitudinal	lateral	longitudinal	lateral
short ($\leq$ 20m)	Shaping Cam	2.724	0.522	1.751	0.805
	Shaping Cam+Lidar	**0.157**	**0.085**	0.376	0.525
full	Shaping Cam	3.856	0.494	2.540	0.982
	Shaping Cam+Lidar	0.359	0.106	0.702	0.655

Figure 4.24 gives an example of how occupancy estimation is improved by using a Lidar. It is quite common in a traffic scene that only a part of the object surface is captured by Lidar, yielding an L-shape or I-shape measurements in the resulting point clouds when looking from a bird's-eye view. The reasons for that include reflective surface material, occlusion, and clustering errors. Relying on our prior knowledge about 3D geometries of

a specific object class, e.g., a car, the estimated occupancy bounding box are more correct even if the point cloud cluster is sparse and incomplete.

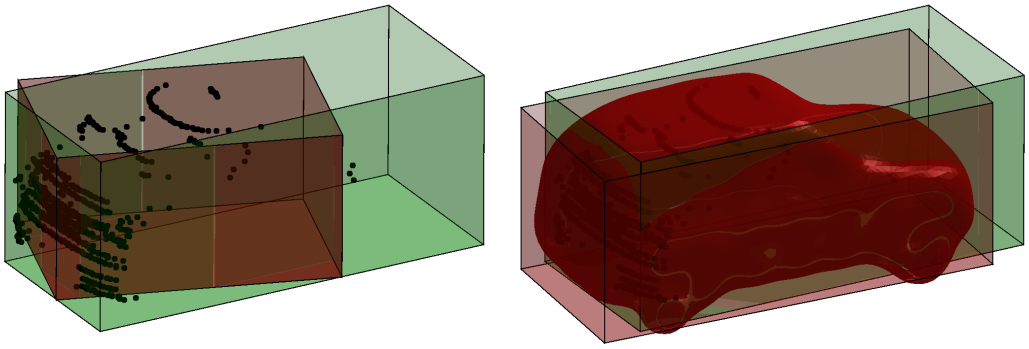


Figure 4.24: Improvement in occupancy estimation for 3D Shaping. The left image shows the bounding box (dark red) that is directly obtained from a Lidar cluster. The right image shows the estimated bounding box (red) through 3D Shaping. The green bounding box illustrates the groundtruth.

Figure 4.25 shows a rendered result when heavy self-occlusion occurred. Using a Lidar allows us to cluster the scene directly in 3D and thus to reconstruct multiple instances within the same class that are occluded by each other. This enables us to build a 3D environmental model at an object level, which is essential for in-vehicle augmented reality.

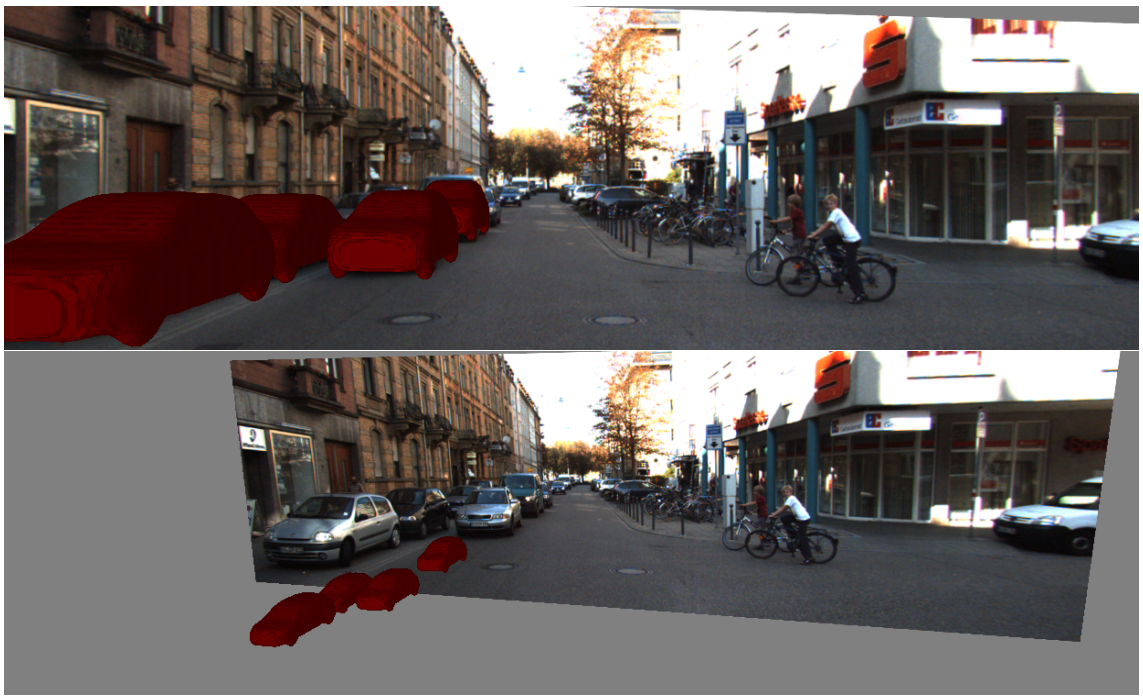


Figure 4.25: AR visualization of 3D Shaping by heavy occlusion.

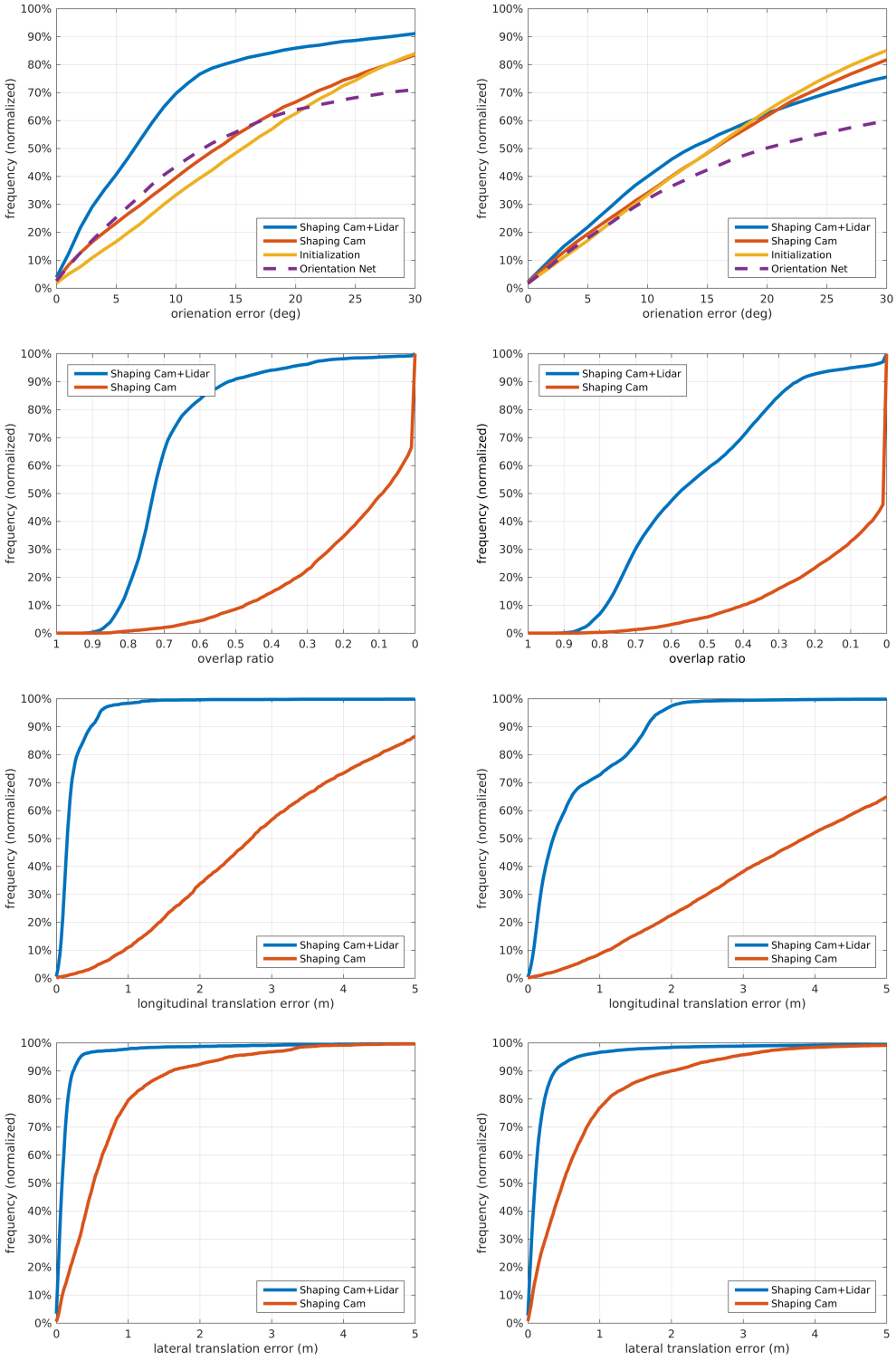


Figure 4.26: Evaluation of 3D Shaping + Lidar. Left: short range objects. Right: all objects. From top to bottom: absolute orientation error, overlap ratio of occupancy bounding box, absolute longitudinal error, and absolute lateral error. The vertical axis indicates the percentage of estimates that are smaller or larger than the corresponding threshold on the horizontal axis.

4.6 Summary and Future Work

In this chapter, we presented a novel workflow named 3D Shaping for the reconstruction of 3D environment at an object level. We first proposed monocular 3D Shaping that only requires a single image from a monocular camera as input. Then, we presented two extensions of monocular 3D Shaping by taking advantage of additional in-vehicle 3D sensor in order to make it more practical for real traffic scenes.

The monocular 3D Shaping workflow combines an existing silhouette-based reconstruction technique with deep neural networks. The performance of 3D reconstruction is boosted by nearly 20 percent in viewpoint accuracy. Through the use of an extremely low-dimensional latent shape space, we are able to describe the complete 3D geometry of an object using only two additional parameters. In other words, the transmission of complex 3D object geometries through a low- or medium-bandwidth in-vehicle bus becomes possible. This has great relevance for in-vehicle augmented reality.

The first extension we proposed for monocular 3D Shaping is named Pose-RCNN, a novel neural network architecture that is able to jointly detect objects and estimate viewpoint angles. We showed competitive results of our proposed Pose-RCNN on the KITTI vision benchmark against other state-of-the-art approaches. The second extension we introduced for monocular 3D Shaping is referred to as 3D Shaping + Lidar, where we directly use 3D point clouds measured by a Lidar as an additional cue for 3D shape optimization. The evaluation showed a significant improvement in pose and occupancy bounding box estimation against the monocular 3D Shaping approach. Relying on the proposed extensions of monocular 3D Shaping, object-level 3D reconstruction in a traffic scene becomes possible.

Although the presented 3D Shaping approach and its extensions already achieved satisfying reconstruction results, there are still possible improvements which could be taken into consideration in the future. Instance-ware segmentation could be applied in order to further increase the quality of 3D reconstruction for scenes with heavy occlusion. A neural network that directly reconstructs a 3D geometry from an image would also be of great interest. These are the scope of further research on the topic of 3D reconstruction.

5 System Design and Integration

Rome wasn't built in a day.

John Heywood
c. 1497 - c. 1580

Rome was not built in a day [Bed00]; In-vehicle augmented reality is not built in one step as well. Today, compared to the popularity of AR applications in consumer electronic devices such as smartphones or glasses, AR in a car is still not a mature technology. Why is it so difficult to incorporate augmented reality in a car, especially when the sensors installed in a modern vehicle are much more powerful than those in a smartphone? What are the specific challenges in realizing in-vehicle augmented reality? How do we cope with these challenges? These are the questions that we answer in this chapter.

We begin this chapter with a brief introduction of system design in Section 5.1, followed by the related work in Section 5.2. In Section 5.3, we present a purely GPS-based AR system which we design for the current generation of production cars. We address the challenges which we faced during the development, and we explain in detail how we made design decisions in order to overcome the challenges. In Section 5.4, we present a redesigned AR system for the next generation of production cars that will be equipped with more advanced sensor technology. We take advantage of in-vehicle depth sensors and fully exploit depth understanding for the next generation AR system. Also, we show the benefit of Stixel compression proposed in Section 3.4 for the transmission of depth information from one vehicle domain to another. In Section 5.5, we propose our design for future in-vehicle AR system aimed at enabling AR for series production. We present the functional modules, software components, and the E/E architecture of our designed future AR system in detail. Here, we fully exploit scene understanding, more specifically, the 3D Shaping pipeline proposed Section 4.3, in order to boost the performance of AR functions. In the last section, Section 5.6, we summarize this chapter and outline directions for furture work.

5.1 Introduction

The objective of system design is to determine the overall system blueprint that satisfies the system's essential requirements [RWD12]. It involves a series of processes defining functions, software components, hardware components, architectures, and physical processing components of a system. In general, the output of system design is a deliverable

that contains all design documents of the afore-mentioned system elements, from the functional layer down to the physical layer. This document is referred to as *system specification*.

In the automotive industry, a system specification plays an important role as the interface between OEMs and suppliers. In a traditional development process that follows the V-model (Fig. 5.1), OEMs generally do not implement software or hardware components themselves. The actual implementation is usually outsourced to suppliers, who implement the designed system according to the system specification made by the OEMs. It is thus essential for OEMs to genuinely "translate" the system's requirements into the system specification during system design. Any late phase requirement change would result in massive development cost by both OEMs and suppliers and eventually seriously delay the entire product pipeline.

At an early stage of development, OEMs use to build prototype systems themselves before making design decisions and deliver specifications to suppliers. The advantage of system prototyping lies above all in the fast cycle of requirements analysis, system design, implementation, and testing. Relying on system prototyping, developers are able to get evaluation and feedback from test users, which helps them update and refine the system step by step. This approach is typically helpful if system designers find themselves lacking know-how on new types of systems that have never been built before.

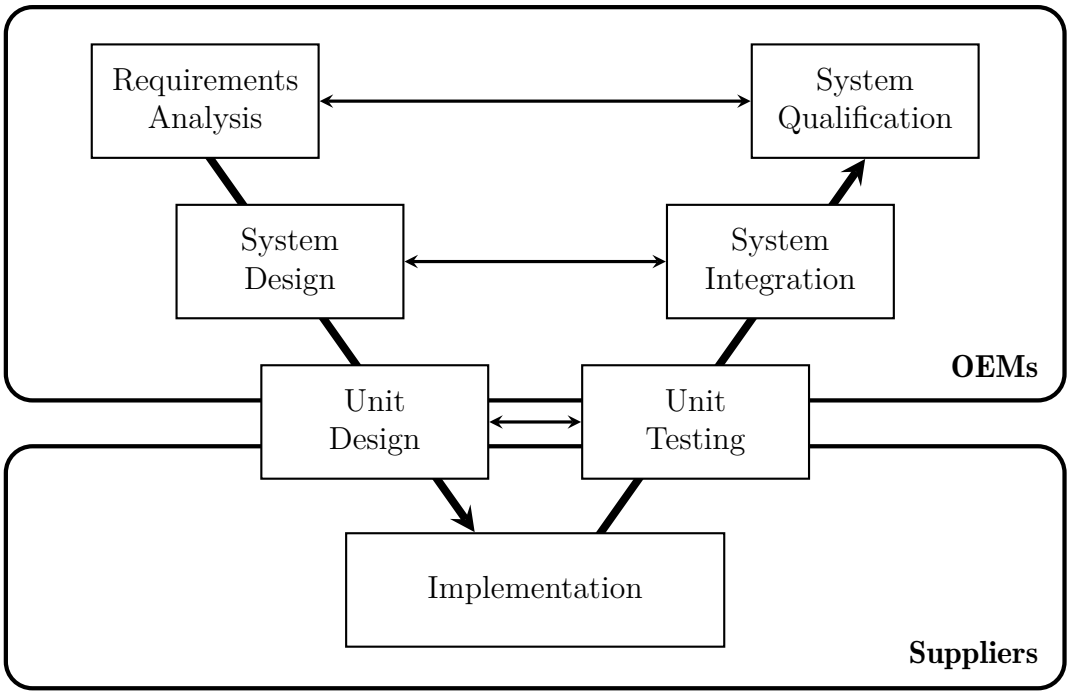


Figure 5.1: Development model in the automotive industry that resembles the letter 'V'.

5.1.1 Key Contributions

We exactly found ourselves with only a blank sheet of paper at the very beginning of the AR project. So we decided to start with a modest design which uses the available in-vehicle technologies as far as possible. This resulted in our first AR prototype vehicle which was the first of its kind in the automotive industry that successfully proved the concept of in-vehicle augmented reality. Based on the lessons learned from our first AR prototype, we continuously integrated our new findings during the research in *depth understanding* and *scene understanding* into our new designed AR systems. We fully exploited depth understanding in our second design for the next generation of in-vehicle AR systems, and we integrated it into a modified upper-class production car equipped with more advanced sensors compared to our first AR prototype vehicle. Beyond that, we propose a third design for future AR system aimed at future series production, where we deploy scene understanding in order to boost AR functions including localization, tracking, rendering.

From a technical view point, the contributions of this chapter are described as follows. In Section 5.3, we introduce a low-cost GPS-smoothing algorithm in substitution for Kalman-filter-based tracking or visual tracking. The proposed algorithm requires much less computation resource than a Kalman filter. In exchange, the robustness by tracking is compromised in some corner cases. We use a "clever" HMI design and a sophisticated data retrieval mechanism to visually compensate the unstable tracking effect. In Section 5.4, we demonstrate how AR functions including road surface estimation, ego-motion estimation, and depth culling can be supported through depth understanding. We show the advantage of our proposed Stixel compression scheme which enables the transmission of depth information through an economically reasonable in-vehicle transmission system, namely a CAN bus, as already discussed in Section 3.4. In Section 5.5, we present in detail the functional modules, software components, and the E/E architecture of our designed future generation of in-vehicle AR systems. We explain how road surface estimation, ego-motion estimation, and depth culling can be further boosted through scene understanding. Furthermore, we show how 3D Shaping proposed in Section 4.3 can be integrated into the future generation AR system, which enables object-level 3D reconstruction of the surroundings.

In a nutshell, the key contributions of our work in this chapter lie above all in the proof of concept of in-vehicle AR and the continuous integration of research results into new designed AR systems, which solves the last and the most important key problem (Section 1.4.3) for in-vehicle AR. We expect our work to be seriously considered by industrial decision makers and eventually adopted in series production cars in the future.

5.2 Related Work

Related AR projects in the automotive industry are already introduced in Section 1.1 and Section 1.2. In the academic research, AR prototypes have also been built for years. In 2004, Hu et al. [HU04] developed the VIsion-based Car NAvigation System (VICNAS)

with an AR navigation mode. The system augmented the road using virtual navigation arrows through a hybrid tracking approach based on visual, GPS, and inertial measurements. In 2012, George et al. [GTFC12] presented the Driver Assistance by Augmented Reality for Intelligent Automobile (DAARIA) system which was able to detect obstacles in the surroundings of the prototype vehicle and track the eyes of the driver. Two commercially available camera systems were adopted for detection and tracking. A wide angle scene camera was additionally mounted behind the rear-view mirror, serving as the system input. Proper augmentation of virtual objects was achieved through a precise extrinsic calibration between different cameras of the system.

We note that none of the previous works, both AR projects in the industry and AR prototypes built by research institutes, aimed at integrating their developed AR systems into a production car. As already mentioned in Section 1.2, this work is, to our best knowledge, the only one so far that comprehensively discusses the design and integration of an in-vehicle AR system aimed for possible series production in the future.

5.3 Current Generation AR System

5.3.1 System Design

We started with a rather modest design of the in-vehicle AR system in 2013, aiming at only using technologies that are available in a middle-class commercially available production car with least possible modifications. Regarding the limit of available sensors and software components back then, we made the following design decisions.

- The AR system shall be purely GPS-based.
- The AR system shall be able to show contact-analog information to some extent.
- The AR system shall operate in two different modes, namely AR driver assistance and AR passenger infotainment, respectively.

Figure 5.2 shows the layered architecture of our designed AR system. The input layer of the system comprises cameras, vehicle sensors, navigation routes, and a database of Point of Interests (POIs). The cameras are only responsible for taking images of the real world and streaming them into the AR Engine without any further processing. The vehicle sensors, including a GPS, a steering wheel sensor, a wheel odometer, and an inertial measurement unit (IMU), together with the navigation routes, provide measurements to the AR engine in order to determine the global position of the ego-car. All relevant measurement data are fed into a sensor fusion interface in order to make more precise estimations of the position and orientation of the cameras. The AR engine, functioning as the core of the AR system, is responsible for retrieving nearby POIs (virtual objects) from the POI database, projecting them onto the image according to their relative positions to the ego-car, and compositing AR blended images. The blended images are then sent out to the output layer of the system which comprises two displays: a Liquid Crystal Display

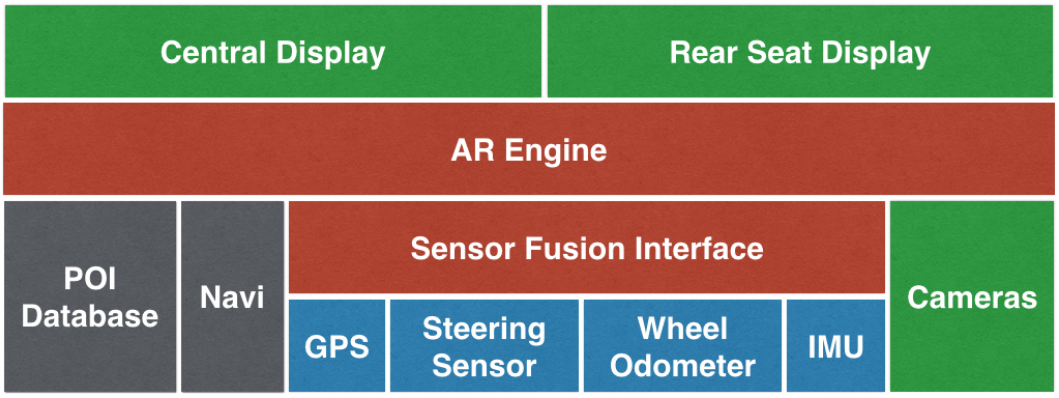


Figure 5.2: Overview of the AR system architecture. Hardware components that are originally available in the vehicle are marked in blue, while additionally installed components are highlighted in green. Red blocks show fundamental software components of the AR system. Gray blocks draw offline or online databases.

(LCD) mounted on top of the central console and a touch screen mounted at the back of the driver seat. AR blends for driver assistance and passenger infotainment are separately displayed on these two output screens.

In fact, we built four displays (central, rear seat, side window, and HUD) and two cameras (front, side) into the car at the very beginning in order to study which combinations are most satisfactory for which AR use cases. Certainly, the HUD is the preferred display for in-vehicle augmented reality. However, the display area of a current generation of HUD is only ca. 21cm $\times$ 7cm, which is not enough for showing contact-analog navigation arrows and driver assistance information. Therefore, we decided not to use the HUD for the current generation AR system. Also, AR features on the side window screen turned out to be rarely satisfying. Quite the opposite of the Window to the World concept [CT11], most of the time, the side camera in our prototype vehicle only captured fields and forests on German highways and country roads. Even in the best cases, augmented POIs appeared less than a second on the side screen. This is too short for passengers to perceive the virtual objects, let alone to interact with them. Therefore, we also ceased the development of side window features, which was expected to be the least challenging task among other design options.

The final system components of our AR system are illustrated in Fig. 5.3. We integrated our designed system into a Mercedes-Benz R-Class which had enough trunk space for modifications. The front camera with an 83° diagonal FOV was mounted behind the rear-view mirror. It took video of the real world at 25 frames per second and streamed it to the AR engine. The AR engine ran on a mobile workstation with a 2.2GHz Intel Core i7-2720QM CPU and a NVIDIA Quadro 2000M GPU. An accelerometer was mounted above the middle point of the rear axle in order to capture roll and pitch motions of the ego-car. We additionally installed an UMTS router which enabled the AR engine to access databases of POIs. Several pictures of the modified prototype vehicle are presented in Fig. 5.4.

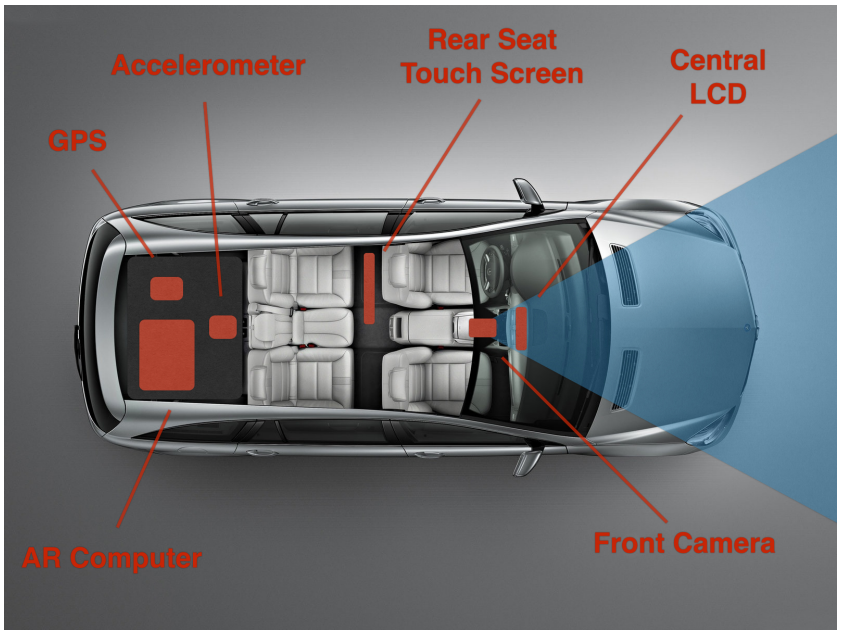
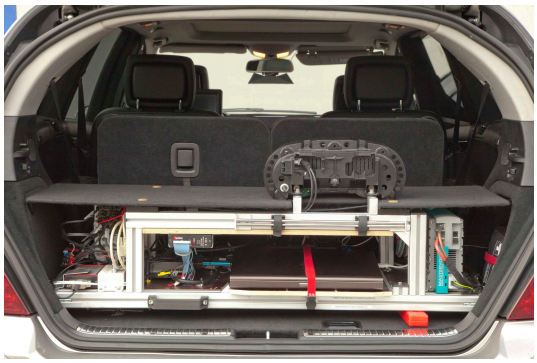


Figure 5.3: Different system components in the R-Class prototype vehicle. Image source: Daimler AG. Adapted by the author.



(a) Vehicle trunk.

(b) Central LCD.

(c) Rear-seat touch screen.

(d) Vehicle interior.

Figure 5.4: The R-Class AR prototype vehicle.

5.3.2 System Implementation

In the following subsection, we address several challenges we faced during the implementation of the AR system and explain how we overcame them. These include, but are not limited to, how to estimate the camera pose using position-based techniques, how to retrieve and organize POIs from diverse content providers, and how to compensate inaccuracy of pose estimation using a sophisticated HMI design.

Camera Pose Estimation As already explained in Section 2.4, a realistic AR blend requires precise alignment between the real and the virtual camera. Since our virtual POIs are defined in the global coordinate system, this requirement can be equally formulated as to estimate the pose of the ego-car in the real world, where the AR camera is mounted. We simplify this problem using Constant Turn Rate and Velocity (CTRV) model [SRW08], which tracks the longitudinal and lateral position (x, y) of the ego-car, as well as the orientation (yaw) angle θ. The CTRV model works well if there is no sudden acceleration expected during the drive. Besides, the pitch and the roll angle are compensated through the additional accelerometer. The state equation of CTRV is given in Eq. 5.1, where v and ω indicate the linear and the angular velocity of the ego-car, respectively. k is the index of the system state and Δt the time interval between two states.

$$\begin{aligned} x_{k+1} &= x_k + \frac{v_k}{\omega_k}\sin(\theta_k + \omega_k \Delta t) - \frac{v_k}{\omega_k}\sin\theta_k \\ y_{k+1} &= y_k - \frac{v_k}{\omega_k}\cos(\theta_k + \omega_k \Delta t) + \frac{v_k}{\omega_k}\cos\theta_k \\ \theta_{k+1} &= \theta_k + \omega_k \Delta t \end{aligned} \tag{5.1}$$

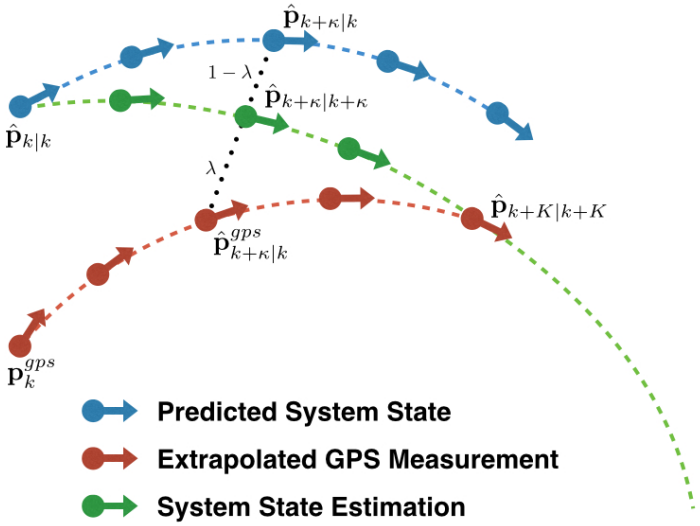


Figure 5.5: Illustration of the smoothing step by GPS-based pose estimation.

GPS measurements (latitude, longitude, and heading) are used to update the predicted system state. The particular problem regarding GPS measurements is their slow update frequency (only one update per second), whereas the velocity and the yaw rate are updated 20 times per second. As a consequence, the measured GPS pose $\mathbf{p}_k^{gps}(x_k^{gps}, y_k^{gps}, \theta_k^{gps})$ differs most of the time from the pose $\mathbf{p}_{k|k-1}$ predicted by the system state equations. To overcome this problem, we added a low-cost smoothing step where GPS measurements are extrapolated relying on vehicle dynamics (Fig. 5.5). A new GPS measurement is predicted K-steps into the future according to the current velocity and yaw rate of the ego-car. K indicates the number of system states between two GPS measurements. A linear combination of the extrapolated GPS pose and the predicted pose, as formulated in Eq. 5.2, is then used as the actual estimate for system update. The weight λ is given by κ/K in order to increase the smoothness of the trajectory.

$$\hat{\mathbf{p}}_{k+\kappa|k+\kappa} = (1-\lambda)\hat{\mathbf{p}}_{k+\kappa|k} + \lambda\hat{\mathbf{p}}^{gps}_{k+\kappa|k}, \kappa \in \{1, 2, \ldots, K\} \tag{5.2}$$

Compared to a more complicated update step, e.g., the update equation used by Extended Kalman Filter (EKF) [Sor85], our proposed low-cost smoothing step avoids the calculation of the Jacobian and thus allows us to save computation time and spare critical resources of the AR system.

POI Data Retrieval While retrieving POIs from online content providers, we faced a problem caused by the density of POIs and the diversity of POI data format. In some specific areas such as tourist hot spot or urban downtowns, POIs usually overlap with each other on the displays. This would result in a confusing and irritating AR output (Fig. 5.6). Also, similar POIs retrieved from different providers could appear on the screen at the same time, showing duplicated information to the users.

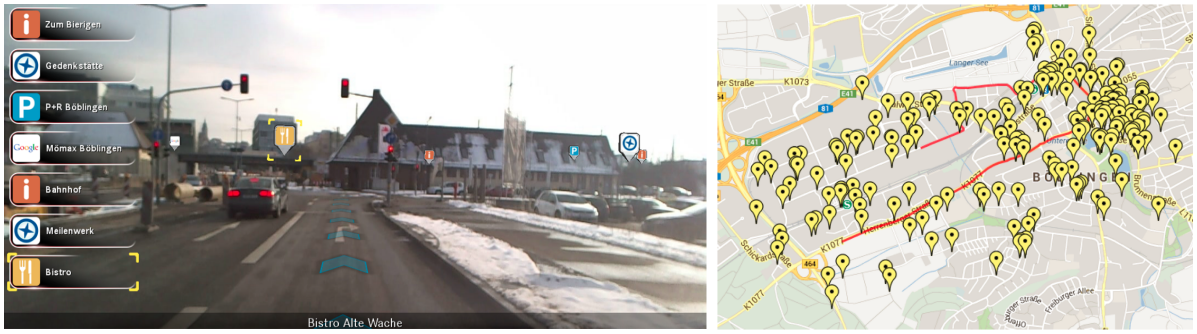
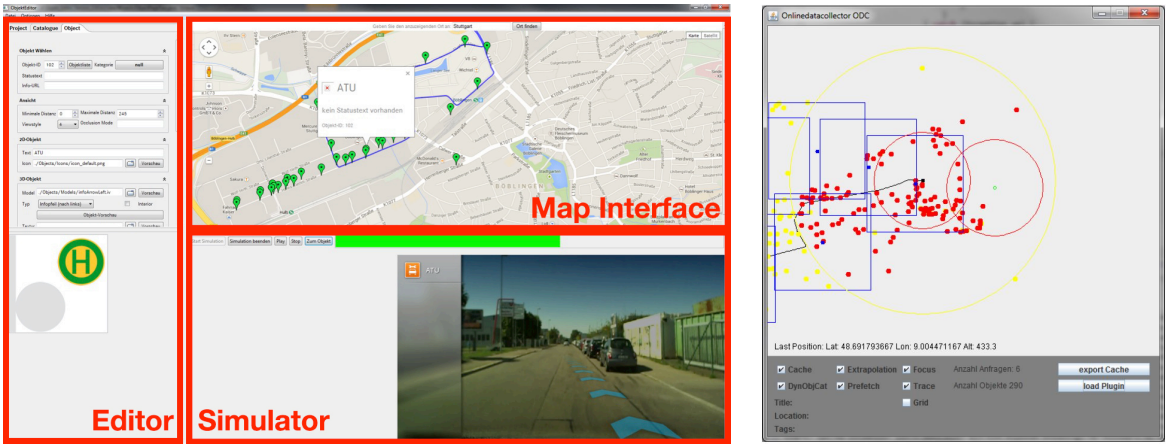


Figure 5.6: Early stage experiment with online data retrieval. Image source of the map: Google Maps.

We solved this issue by filtering and preprocessing raw POI data in a semi-automatic manner using a self-implemented software component named Object Editor (Fig. 5.7(a)). Its graphical interface is divided into three functional panels. The editor panel provides basic editing functions for POIs and object catalogs. The map interface allows selecting test tracks and generating new POIs directly on the map. The simulator window shows

how virtual POIs would look like in the AR blended output. The object editor provided functionalities to finetune various attributes of a POI, such as its geographical location, icon texture, and 3D shape among others. Using the object editor, we were able to select POIs manually out of the POI dataset that we want to augmented.



(a) Object editor. (b) Online data collector.

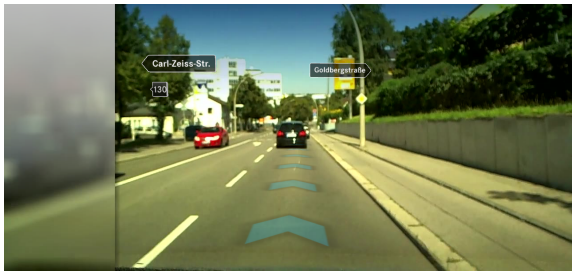
Figure 5.7: Self-developed tools for POI data retrieval.

Internet connection was another issue throughout online POI retrieval. The quality of the Internet connection is subject to a variety of factors, such as weather condition, the location of the ego-car, and the infrastructure in the surroundings. The time of response to a retrieval query varies from few seconds to infinity. For example, queries sent out from a tunnel or a forest would get lost most of the time. Also, with respect to the speed of a moving vehicle, the average response time from an online content provider (a few seconds) is still too long.

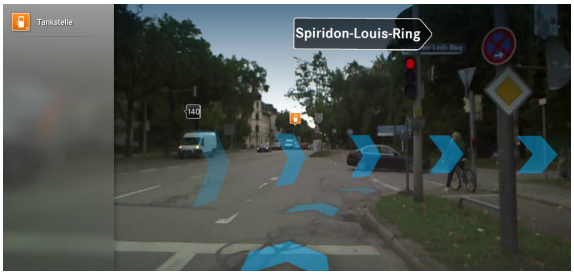
In order resolve the connection issue, we developed a software module called Online Data Collector (Fig. 5.7(b)) in order to fetch and cache POIs in advance. We use navigation information to predict an area where the ego-car will show up. In case no navigation routes are given, the upcoming route is predicted using extrapolation based on the motion dynamics of the ego-car. POIs in the upcoming area are then fetched from content providers and cached into the AR engine.

HMI Design Driving distraction has always been a challenging issue for the design of the Human Machine Interface (HMI) for in-vehicle augmented reality. A not properly designed AR user interface would increase the driving stress through confusing augmented information. Since it is hard to keep a low-level distraction while providing interactivity at the same time, we decided to design two HMIs separately for AR driver assistance and AR passenger infotainment, respectively.

For drivers, we squeezed out the most significant information to help them by driving tasks. We replaced traditional navigation arrows through a *navi-carpet* (Fig. 5.8(a))



(a) Street names and how numbers.



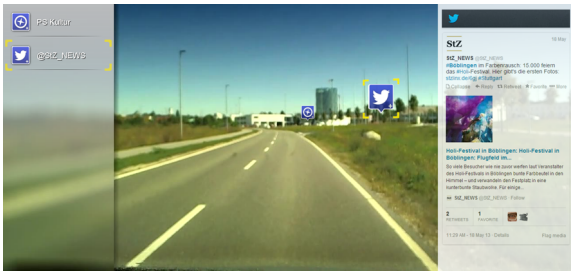
(b) Navigation carpet and barriers.

(c) Speed limit warning.

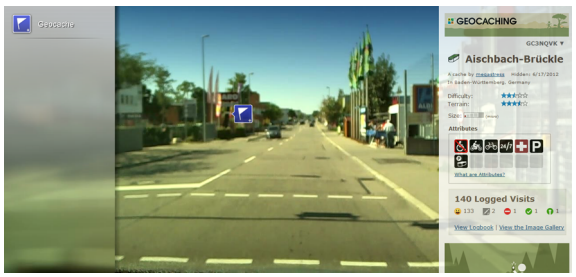
(d) Traffic jam head.

Figure 5.8: Example of AR driver assistance.

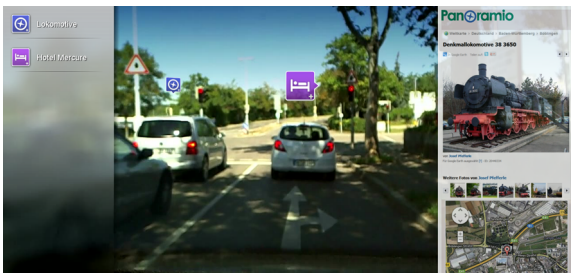
(a) Standard view.



(b) Social media.



(c) Gaming.



(d) Scenic information.

Figure 5.9: Example of AR infotainment.

which is blended onto the road surface in the real world as if it was real road marking. In curves and roundabouts, a part of the navi-carpet turned into a barrier "standing" on the road (Fig. 5.8(b)). This would catch more attention of the driver for the upcoming turning action. In a dangerous situation such as speeding (Fig. 5.8(c)) or traffic jam (Fig. 5.8(d)), the color of the navi-carpet turned red in order to warn the driver. Emphasized virtual traffic signs and road banners would furthermore remind the driver of the dangerous situation.

For rear-seat passengers, we integrated social networking (Fig. 5.9(a)), gaming (Fig. 5.9(a)), and tour guiding (Fig. 5.9(b)) features into the AR infotainment system. These features were presented to the passengers through a standardized POI interface. If a passenger had a special interest in a specific POI, he or she could touch it on the screen in order to gain more detailed information. In some cases, e.g., by sudden acceleration in curves, POIs could be hardly hit on the screen through finger touch. Therefore, we reserved an area on the left screen where we listed all currently visible POIs. Users can always touch their interested POIs in this area and have an overview of all of them.

5.3.3 System Testing and Demonstration

We tested our AR prototype vehicle in real world traffics. Figure 5.10 shows one of our prepared test tracks in Stuttgart region, which covered an urban area, an industrial area, and a country area in a small forest, providing a large variety of traffic scenes.

Figure 5.10: Test tracks for the R-Class AR prototype vehicle in Stuttgart region. Image source of the map: Google Maps.

During the road tests, we added unusual driving maneuvers such as circling roundabouts or making a sudden turn, in order to explore the limit of the GPS-based positioning system. The results showed certain robustness of the system to augmented POIs reasonably, as long as we drove the test vehicle in an "appropriate" manner in curves and roundabouts. In case of sudden acceleration at a turn, we could observe a "drift effect", as shown in

Fig. 5.11. The drift effect is caused by an unrealistic approximation of the mathematical model used by the pose estimation step, since the CTRV model only assumes constant velocity and is thus not able to properly track acceleration. We modified several HMI designs to make this effect less noticeable to the users, such as lifting the navi-carpet in curves and adding the static area of POI list. Also, the self-developed online data collector is able to filter out POIs near sharp curves upfront, so that they are not shown to the users at all. However, we believe that in order to fundamentally solve this problem, a sensor fusion approach combining visual cues is indispensable.

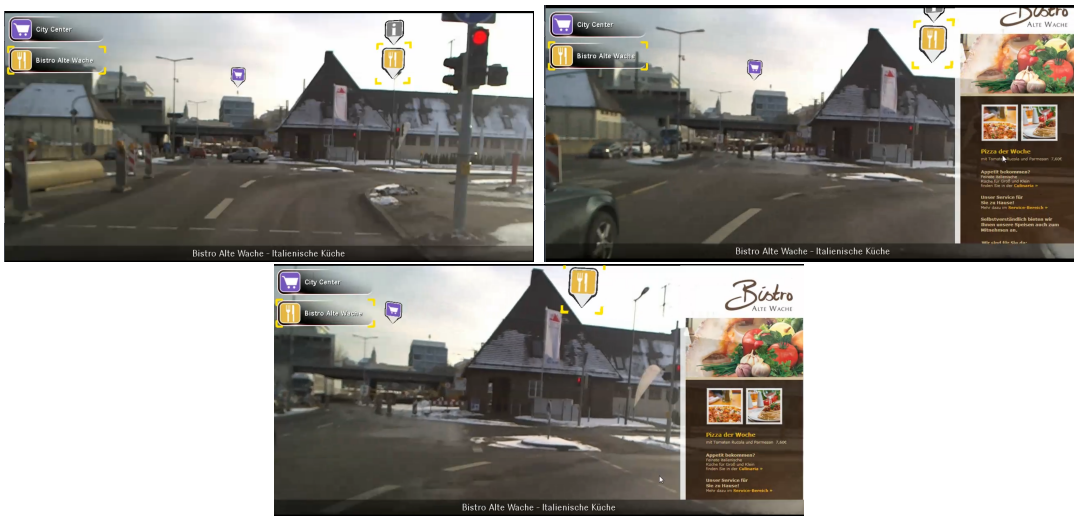


Figure 5.11: Drift effect of virtual objects by sudden acceleration in a curve.

We also conducted usability testing in order to evaluate the HMI of our AR system. We invited over 200 beta testers[1)] to the Mercedes-Benz Driving Simulation Center [Zee10] (Fig. 5.12) and placed them into the simulator where the same interface of our AR system was simulated. The only difference between the simulator and our prototype vehicle was that the AR blended output was directly shown on the 360° screen in the simulator instead of on the displays in the car.

During the usability testing, we asked the participants a number of open questions such as *"What do you particularly like with respect to the augmented information?"* and *"What do you not like, what is most disturbing?"*, and we recorded their answers. A summarized answer to the question *"What is your general opinion about in-vehicle augmented reality?"* is presented in Tab. 5.1. It shows that the majority of the participants were enthusiastic about in-vehicle AR, yet there were still concerns about information overflow and driver distraction. The feedback of the usability testing helped us make design decisions for further HMI development.

In October 2013, we demonstrated our R-Class AR prototype vehicle on the augmented reality conference insideAR [Met13], one of the largest AR events worldwide back then. We prepared a short demonstration track near the conference hall (Fig. 5.13) and invited

[1)] All test users are employees at Daimler AG working in different areas.

Figure 5.12: Mercedes-Benz driving simulator [Zee10]. Image source: Daimler AG.

other conference attendees for a 5-minute demo ride. On the occasion of this event, we have received numerous constructive feedback from our industrial peers, which positively helped us in further development towards a close-to-production in-vehicle AR system.

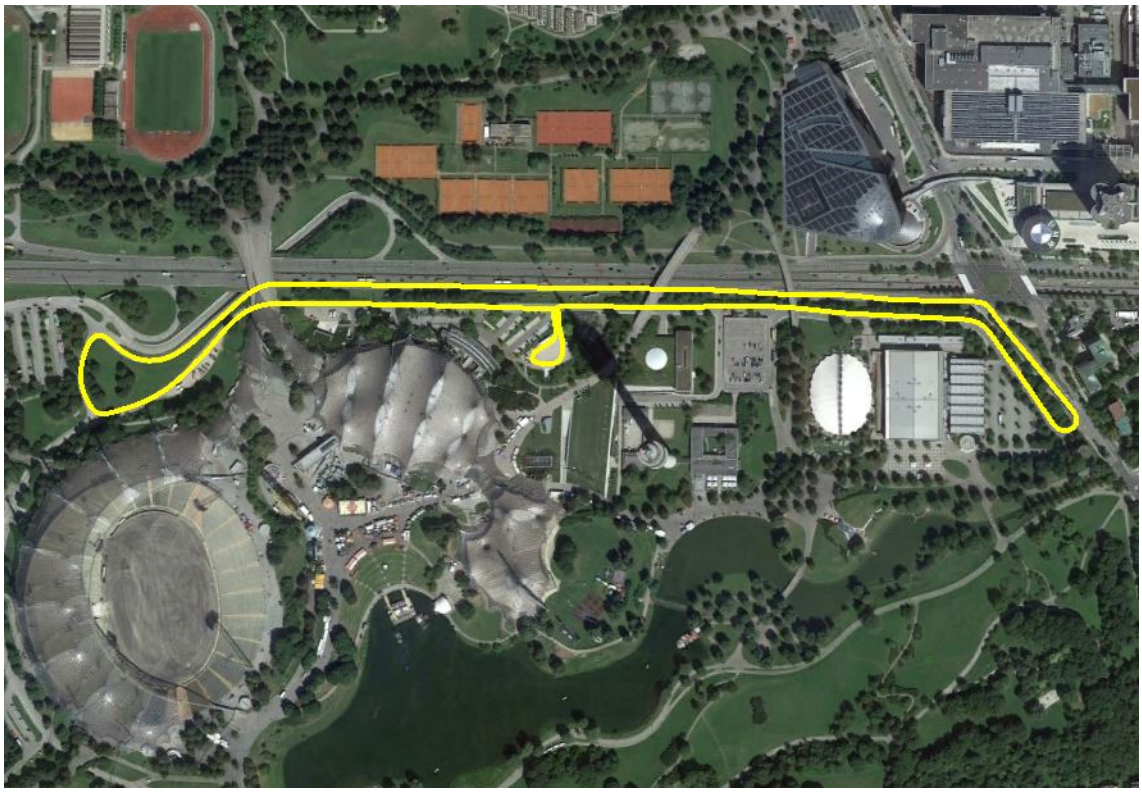


Figure 5.13: Demonstration track on insideAR at Olympiapark in Munich. Image source of the map: Google Maps.

Table 5.1: "What is your general opinion about in-vehicle augmented reality?"

Category	Participants	Percentage	Keywords	Quotes
Positive	154	72%	good, informative, helpful, interesting, etc.	"Basically good. I feel like in a jet fighter." "Good as long as the system remains under control." "Very useful in holiday, or if one is not familiar with the area."
Negative	38	18%	unnecessary, information overload, etc.	"Only sometimes interesting." "Too much detailed information. Rather for co-driver." "Unimportant. I am not interested in the augmented information at all."
Neutral	22	10%	only combined with autonomous driving, information filtering needed, etc.	"Depends on preferences of the user." "Very interesting but nothing special." "Function already available in smartphones."

5.4 Next Generation AR System

5.4.1 System Design

We proved the concept of in-vehicle AR through our R-Class prototype vehicle while keeping the car as unmodified as possible. However, the AR system in the R-Class was still considered far away from being integrated. It was a centralized system running on a mobile workstation. The input (camera) and the output (displays) of the system were additionally installed into the car. Even the tracking algorithm itself was not robust enough in several cases, which had to be compensated using a "clever" HMI design. Therefore, we decided to build a new AR prototype vehicle using a Mercedes-Benz S-Class, the flagship vehicle of Mercedes which was equipped with the most advanced automotive sensors back then. Based on the abundant resources we had in the S-Class, we adjusted our design requirements as follows.

- The AR system shall be distributed.
- Vision-based tracking shall be deployed.
- The interior of the prototype vehicle shall be unmodified, i.e., extra cameras and displays must not be installed.

The new AR system design is illustrated in Fig. 5.14. We built two desktop workstations into the trunk in order to separately simulate the ADAS domain and the telematics domain in the car. They were connected through a Gigabit Ethernet (GE) cable. The ADAS workstation was responsible for processing input image pairs taken by the in-vehicle stereo camera. It forwarded both raw data and processed data to the telematics workstation, where the AR engine was hosted. The AR engine rendered output images for different in-vehicle displays. Figure 5.15 illustrates several possible display options for in-vehicle augmented reality, among which we chose the central LCD and the instrument cluster LCD as the output of our new designed AR system.

The final hardware components of the AR system in the S-Class is illustrated in Fig. 5.16. Compared to the previous system in the R-Class, we installed more hardware components in the vehicle trunk while keeping the vehicle interior as unmodified as possible.

Figure 5.17 shows several pictures of the modified S-Class. We kept the interior of the car "clean" while filling the trunk with workstations and other hardware components. All additional cables for video transmission were laid inside the body of the car and thus not visible to the driver or the passengers. There were only two potentially detectable interior changes which might distinguish the prototype vehicle from normal production cars. One of them was the control panel hiding in the central console storage (Fig. 5.17(c)) which provides power switch functionalities and diverse peripheral interfaces. This panel could be perfectly hidden by the cover of the storage. The other one was the extra installed IMU in the middle of the rear-seat (Fig. 5.17(d)). If the armrest of the rear-seat was re-attached, the IMU would not be exposed to the passengers anymore.

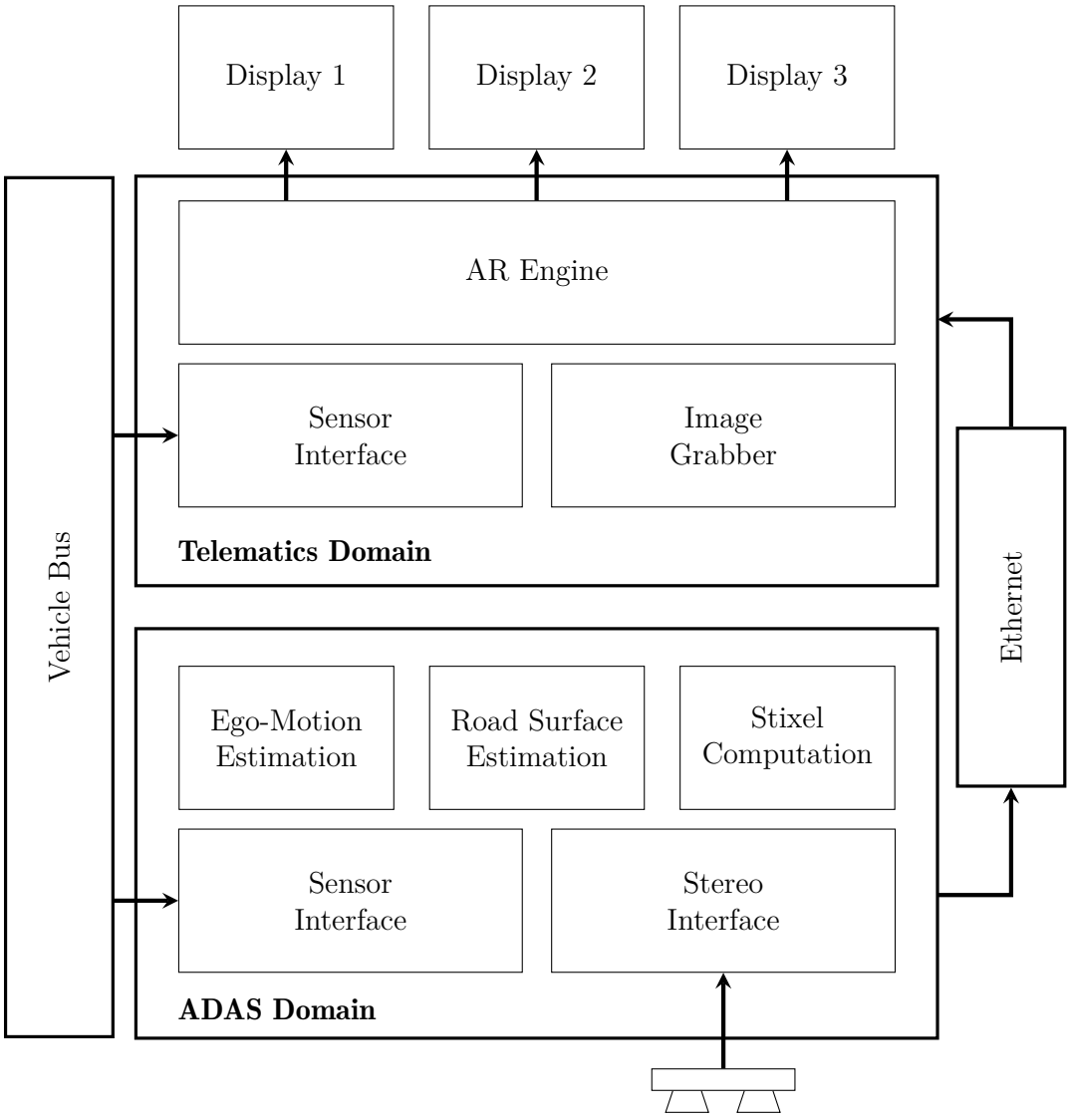


Figure 5.14: AR system in the S-Class prototype vehicle.

Figure 5.15: Possible display areas in the S-Class AR prototype vehicle. Image source: Daimler AG. Adapted by the author.

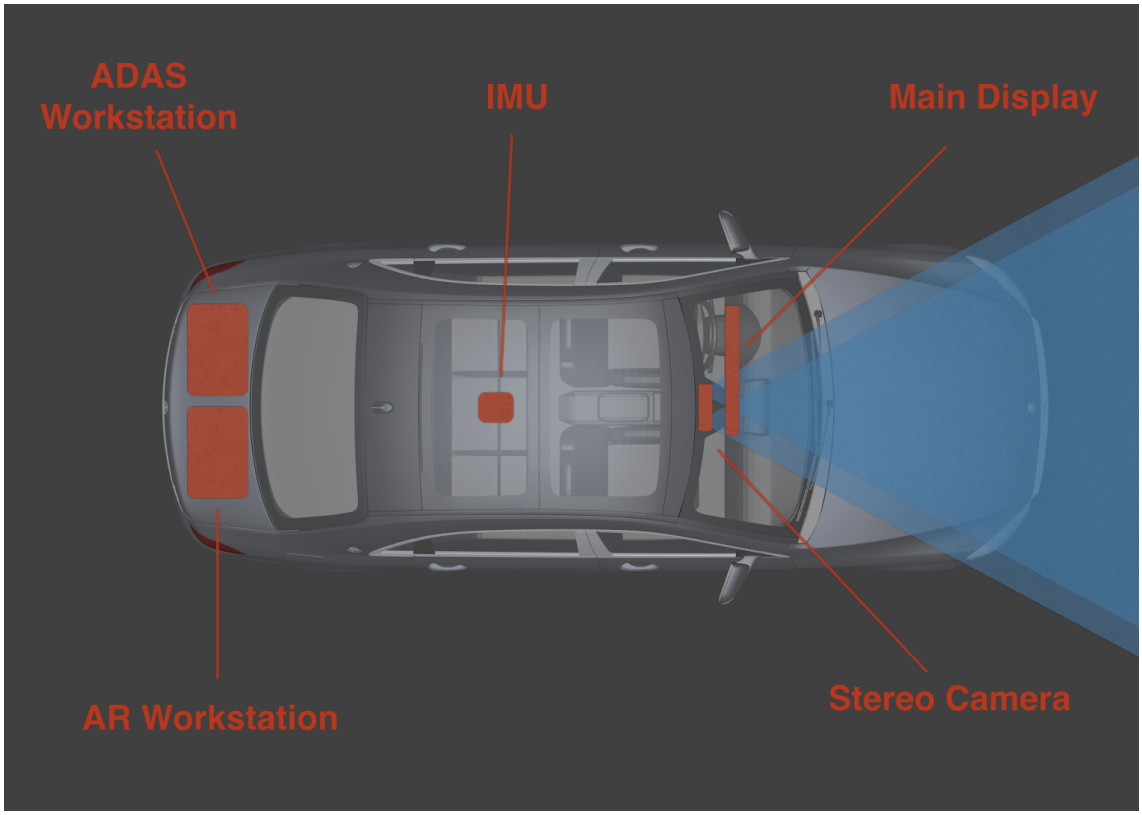
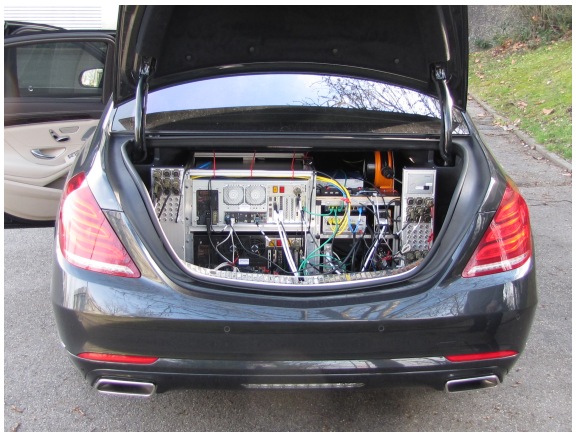


Figure 5.16: Hardware components of the S-Class AR prototype vehicle. Image source: Daimler AG. Adapted by the author.



(a) Vehicle trunk.

(b) Main display.

(c) System control panel.

(d) Additional IMU.

Figure 5.17: The S-Class AR prototype vehicle.

5.4.2 Exploiting Depth Understanding

We fully exploited depth understanding in our new design, which supported a number of AR functions that would improve the robustness and the overall performance of the AR system. We used a depth image to estimate the road surface and the tilt angle of the camera[2)], which enabled a seamless overlay of the navi-carpet. In order to deal with the drifting problem in the R-Class, we integrated a stereo vision based tracking algorithm [FRBG05] to estimate and compensate the motion of the ego-car. We also used Stixel frames as depth masks by rendering. Issues occurred in the previous design could be successfully resolved through the use of depth information.

Road Surface Estimation By assuming the road surface to be planar, the disparities of ground points would approximate a line in the v-disparity space, as formulated in Eq. 5.3. In the camera coordinate system (Fig. 5.18), a point on the road surface satisfies Eq. 5.4, where h and α indicate the height and the tilt angle of the camera, respectively. Additionally, the vertical coordinate v follows the pinhole projection as expressed in Eq. 5.5, where f_y and c_y are the vertical focus and image center of the camera.

$$v = p_1 d + p_2 \tag{5.3}$$

$$\frac{h - Y}{Z} = \tan\alpha \tag{5.4}$$

$$v = f_y \frac{Y}{Z} + c_y \tag{5.5}$$

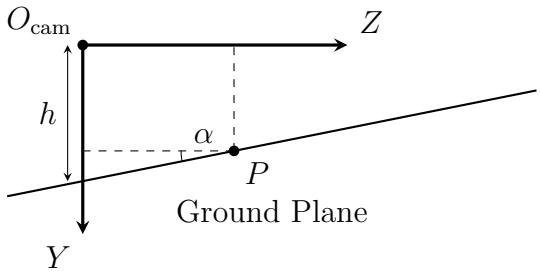


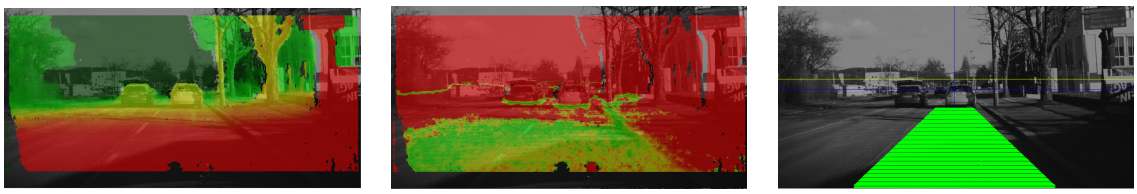
Figure 5.18: Ground plane in the camera coordinate system.

By substituting the disparity d through $d = f_x \cdot b/Z$ (Eq. 2.8) and associating Eq. 5.3, Eq. 5.4, and Eq. 5.5, we are able to derive the dependencies of the parameters as expressed in Eq. 5.6.

$$\begin{aligned} p_1 &= \frac{f_y}{f_x} \cdot \frac{h}{b} \\ p_2 &= c_y - f_y \tan\alpha \end{aligned} \tag{5.6}$$

[2)] The software components of stereo camera-based road surface estimation and ego-motion estimation are implemented by the Image Understanding Group of Daimler AG.

In practice, the line variables p_1 and p_2 are estimated through Least SQuares (LSQ) using a number of (v, d) pairs. Then, the camera height h and the tilt angle α can be determined. With this information, the augmented driving corridor (Fig. 5.19) can be aligned with the road surface even if the slope of the road is steep.



(a) (b) (c)

Figure 5.19: Road surface estimation for AR using depth information. (a) Disparity image. (b) Estimated road area. The likelihood increases from red to green. (c) Augmented driving corridor (technical visualization for engineers).

Ego-Motion Estimation We track the 6 DoF ego-motion of the stereo camera using an Extended Kalman filter. Recall the coordinate system for in-vehicle AR introduced in Section 2.3, the state of the system is the ego-motion matrix M, which transforms a point ${}^cP$ from the previous camera coordinate system to the current one, as the ego-car moves itself in the real world (Eq. 5.7).

$$ {}^cP_k = M_{k|k-1}\, {}^cP_{k-1} \tag{5.7} $$

The ego-motion is predicted using the measurements from the IMU and updated through point correspondences (pairs of (u, v, d) measurements). The Kalman innovation and the measurement matrix can be derived by the optical flow formulated in Eq. 5.8, where X, Y, and Z are the three coordinates of the point ${}^cP$.

$$ \begin{aligned} \Delta u &= f_x \frac{X_k}{Z_k} - f_x \frac{X_{k-1}}{Z_{k-1}} \\ \Delta v &= f_y \frac{Y_k}{Z_k} - f_y \frac{Y_{k-1}}{Z_{k-1}} \\ \Delta d &= f_x \frac{b}{Z_k} - f_x \frac{b}{Z_{k-1}} \end{aligned} \tag{5.8} $$

A more detailed derivation is given in [FRBG05].

We demonstrated the robustness of the ego-motion estimation through a small AR application, as shown in Fig. 5.20. In this example, we randomly threw a virtual cube into the scene and observed whether it remained stable in the world. According to our observation, the virtual cubes were anchored in their environment throughout our experiments, which proved a significant improvement against the tracking algorithm used in the previous system in the R-Class.

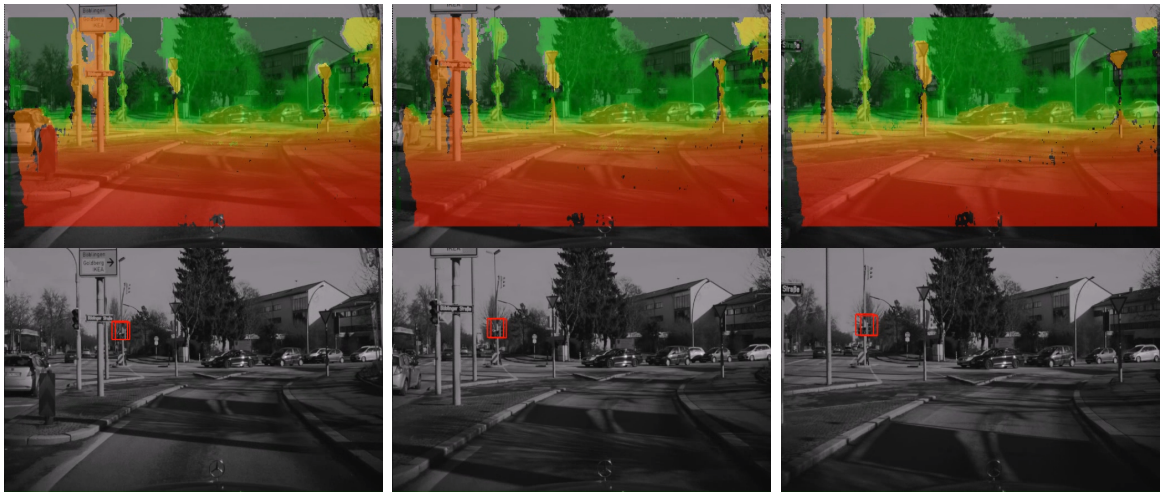


Figure 5.20: Ego-motion estimation for AR using depth information.

Depth Culling We used Stixel frames as a depth buffer throughout the AR rendering. The part of a virtual object which is occluded by closer Stixels was not rendered, as shown in Fig. 5.21. This enabled a more realistic blending between the virtual and the real and thus improved user experience with the in-vehicle AR system.

(a) AR rendering without depth culling.

(b) AR rendering with depth culling.

Figure 5.21: Improvement by AR rendering using depth information.

5.4.3 Key Enabler: Stixel Compression

As introduced in Section 1.4.1, the in-vehicle transmission of depth information is considered a key problem for incorporating in-vehicle augmented reality. As already presented in Section 3.4.4, our proposed Stixel compression algorithm enables transmitting compressed Stixel data through a CAN bus, whereas uncompressed Stixel or compressed data through a general compression algorithm require FlexRay. The latter is more expensive from both point of view of developers as well as customers. In order to enable in-vehicle AR which requires depth information to be transmitted to the telematics domain, the CAN bus connecting the central switch and the telematics gateway would have to be replaced by

FlexRay without applying our proposed compression scheme. Also, the telematics gateway would have to be redesigned in order to be able to decode both CAN and FlexRay signals. Therefore, we believe that our proposed Stixel compression algorithm is a key enabler for in-vehicle AR in the next generation of production cars.

Table 5.2: Payload analysis for transmitting Stixels.

	w/o	Space+Time	Zlib
Instant Peak (KB)	2.02	0.87	1.93
Required Payload (KB/s)	46.37	17.68	46.17
Required Vehicle Bus	FlexRay	CAN	FlexRay
Cost	High	Medium	High

5.4.4 System Demonstration

In May 2014, we invited the advisor of this thesis to a demonstration ride around the TUM main campus in Munich (Fig. 5.22). We demonstrated the S-Class AR prototype vehicle with the major focus on the use of depth information. Accordingly, we demonstrated AR functions in the daily traffic including road surface estimation, ego-motion estimation, and depth culling using Stixels. The advisor gave us positive feedback regarding the AR prototype vehicle and the development of the AR system, which further motivated us to design a future in-vehicle AR system aimed for future series production.

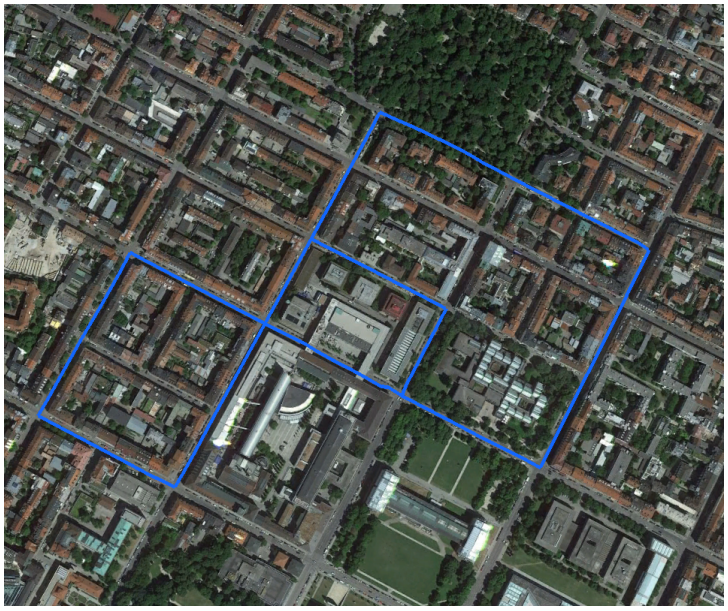


Figure 5.22: Demonstration track around the TUM campus in Munich. Image source of the map: Google Maps.

5.5 Future Generation

5.5.1 System Design

We laid the groundwork for the system design of an in-vehicle AR system through our R-Class and S-Class prototype vehicle. Now, based on the lessons we learned during our previous development, we share our own vision[3)] of how a future in-vehicle AR system would like to be. The future system would take advantage of additional vehicle sensors such as Radar or Lidar, which would enable direct tracking in the three-dimensional world. It would be able to communicate with other traffic users or infrastructure, e.g., traffic light, and understand the current traffic situation and choose the most appropriate virtual content for the users. It would also support various types of user interaction, such as voice control and gesture control. In the rest of this subsection, we explain our designed future system with respect to the functional modules, the software components, and the E/E architecture.

Functional Modules Figure 5.23 shows our vision of typical functional modules of a future in-vehicle AR system. Functions inside the colored box are supported by both depth understanding and scene understanding. Outputs of individual functional modules are fused through a sensor fusion interface, yielding more robust detection and tracking results. We plan additional driver/passenger monitoring functions including gaze estimation, gesture recognition, and voice control, in order to enrich the user experience throughout AR interaction. Furthermore, we believe that Car2X communication will be a new fundamental input source to the AR system, which will enable features that cannot be easily implemented based on depth or scene understanding, e.g., traffic light detection.

Software Components Our vision of typical software components of a future in-vehicle AR system are presented in Fig. 5.24. Different software components are distributed in the ADAS domain and the telematics domain. As more sensors would be installed into a future vehicle, the local surroundings would be comprehensively perceived and understood, enabling robust and realistic blending between the virtual and the real. Images taken by different input cameras would be streamed to the AR engine where they are merged together through image stitching techniques, which would provide users with a different view of the reality. Instant traffic information such as traffic jams or accidents would be sent and received through Car2X communication, which would keep users updated about their journey at any time. AR outputs would also be shown on different screens in the car, including the HUD whose field of view would be significantly enlarged in the future.

[3)] All system designs, diagrams, and features presented in this section do not necessarily reflect the official decisions of Daimler AG.

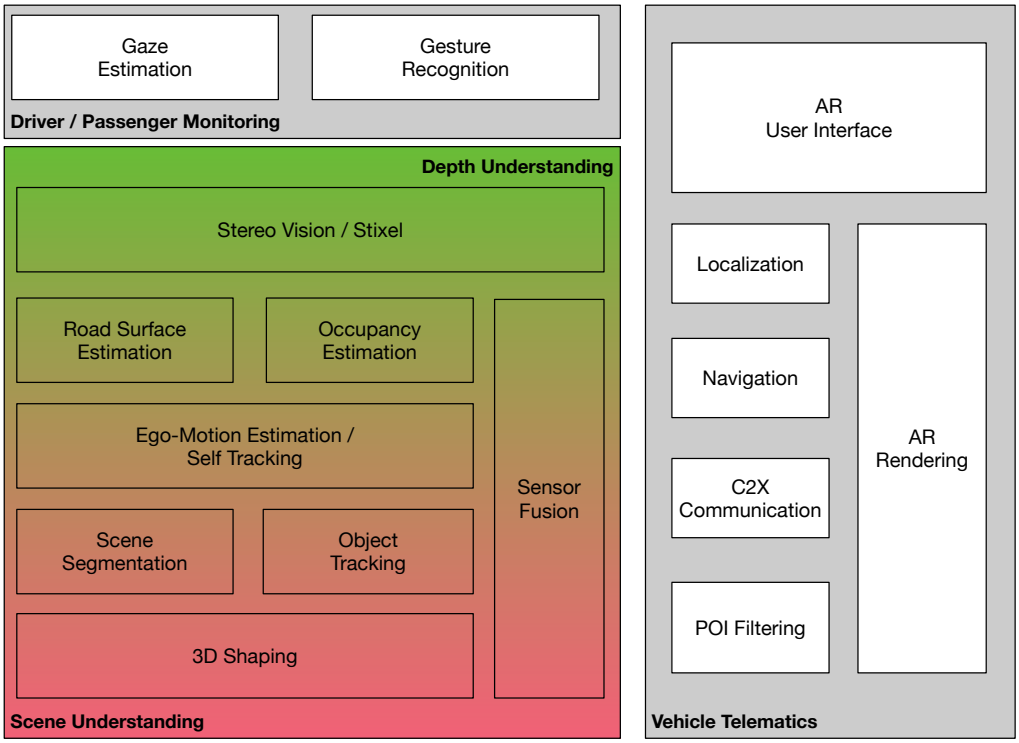


Figure 5.23: Typical functional modules of a future in-vehicle AR system.

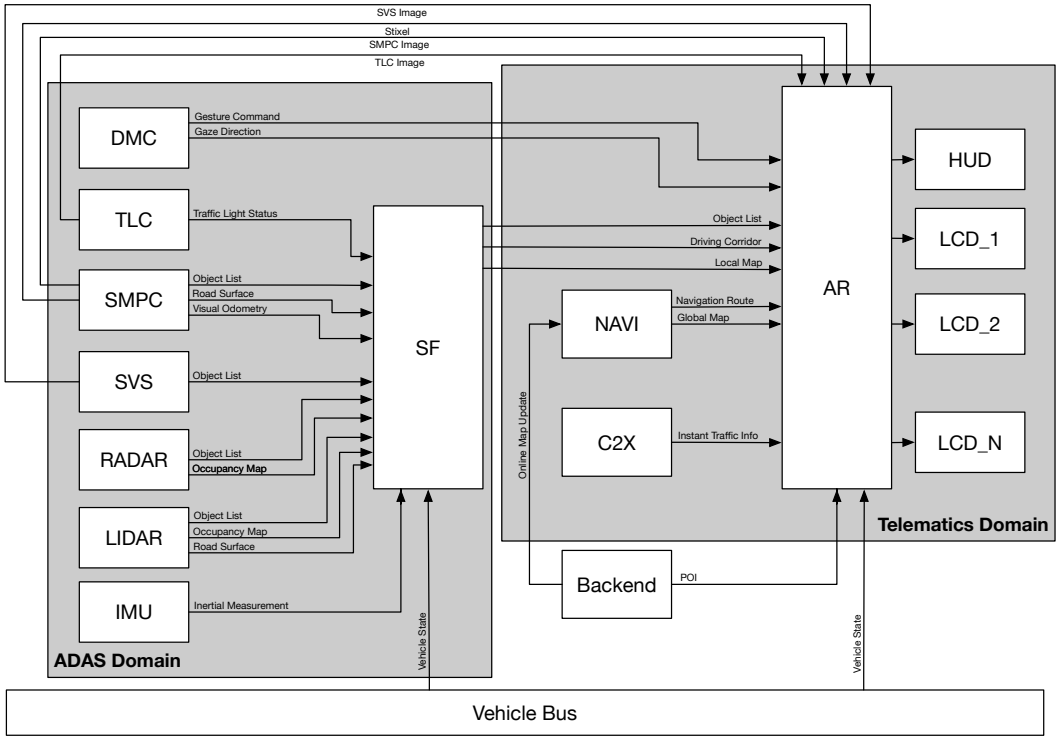


Figure 5.24: Typical software components of a future in-vehicle AR system.

E/E Architecture Domains in a future in-vehicle E/E architecture would be more clearly separated. The overall performance of the AR system would be then boosted through a high-speed inter-domain communication mechanism, e.g., a high-speed backbone bus (Fig. 5.25).

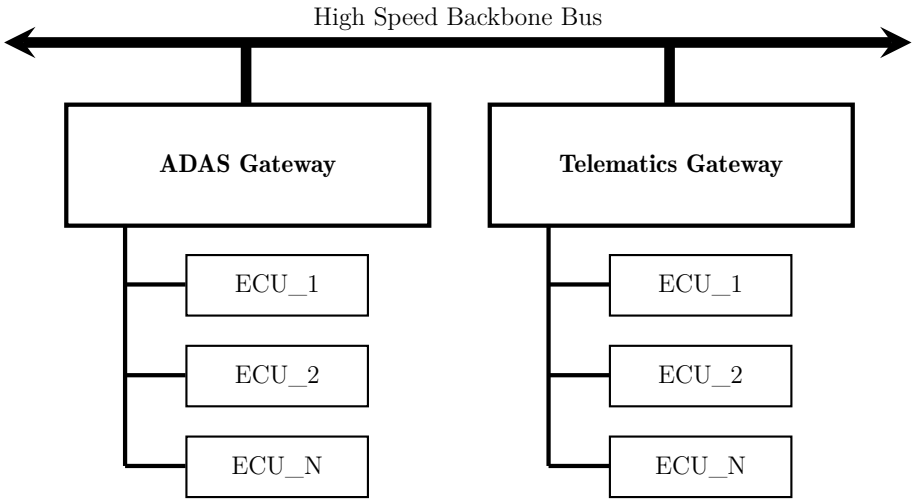


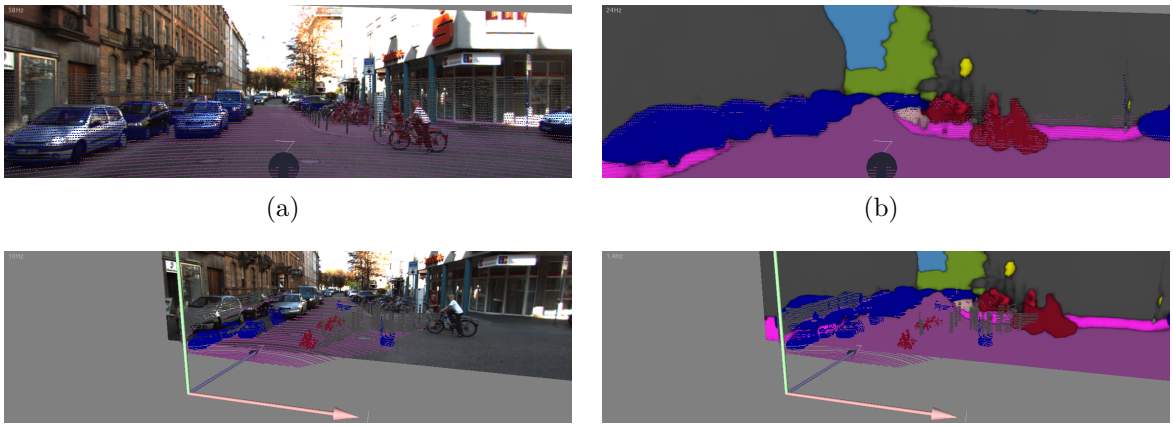
Figure 5.25: Typical future E/E architecture for in-vehicle AR.

5.5.2 Exploiting Scene Understanding

A future AR system would also fully exploit scene understanding. Here, we present examples of how the AR functions including road surface estimation, ego-motion estimation, and depth culling could be further improved through the use of semantic information.

Road Surface Estimation The algorithm introduced in Section 5.4.2 assumes the road surface to be planar and approximates the v-disparity space using linear functions. This assumption does not always reflect the truth. By applying deep-learning-based semantic segmentation techniques, road pixels can be directly identified from an RGB-image and associated with 3D measurements, as shown in Fig. 5.26. Using semantic information will enable estimation of non-planar road surfaces.

Ego-Motion Estimation The performance of ego-motion estimation introduced in Section 5.4.2 largely depends on the quality of the collection of features points named *feature pool*. Feature points extracted from dynamics objects are considered outliers and should be eliminated before going into the Kalman filter. Relying on a pixel-level semantic segmentation, we are able to identify static areas in an image even before extracting feature points (Fig. 5.27). This would increase the robustness of ego-motion estimation, which is critical to the user experience of an in-vehicle AR system.



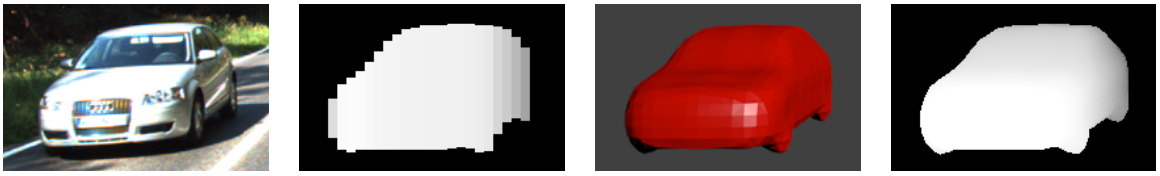
(a) (b)

(c) (d)

Figure 5.26: Road surface estimation using semantic information. (a) Point cloud overlaid on input image. (b) Point cloud overlaid on segmented image. (c)(d) are the corresponding 3D view of (a) and (b), respectively.

Figure 5.27: Elimination of outliers by ego-motion estimation using semantic information. Feature points inside the red eclipses are identified car pixels and thus excluded from the feature pool.

Depth Culling The quality of a depth buffer generated from Stixel data is limited, as Stixel is only an approximation of a fine-grained disparity map. The ridges at the top and bottom of Stixels would cause disturbing rendering effect. In order to generate a smooth depth map, we could use fine geometries reconstructed through 3D Shaping introduced in Section 4.3. The comparison is shown in Fig. 5.28, demonstrating how the use of semantic information and prior knowledge about object geometries would improve AR rendering.



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

Figure 5.28: Depth culling using semantic information. (a) Color image. (b) Depth buffer generated from Stixel. (c) 3D Shaping reconstruction. (d) Depth buffer generated from 3D Shaping reconstruction.

5.5.3 Key Enabler: 3D Shaping

As already stated in Section 1.4.2, object-level 3D reconstruction is essential to solving the 3D registration problem by augmented reality. Our proposed 3D Shaping workflow in Section 4.3 serves as the first step towards generating 3D object map of the entire surroundings. Using 3D Shaping, 3D geometries are embedded into an extremely low-dimensional latent shape space. Theoretically, we only need two additional variables for reconstructing the complete 3D shape of an object. This property is ideal for us to apply 3D Shaping for in-vehicle augmented reality.

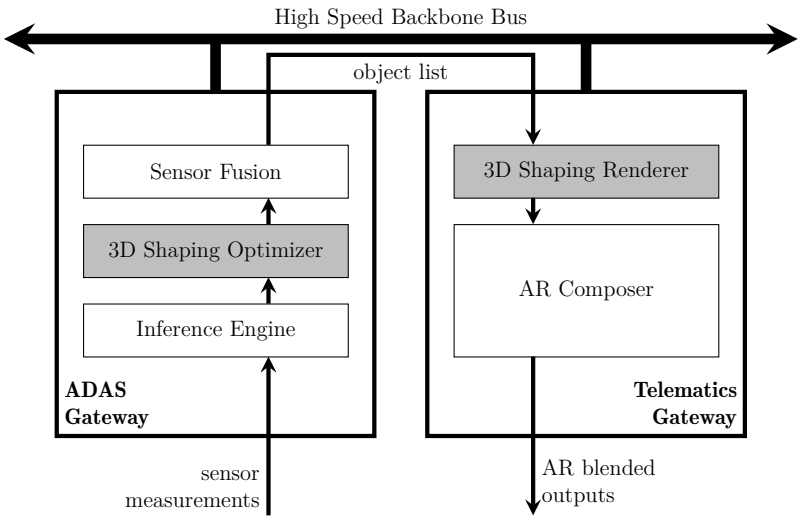


Figure 5.29: Integration of 3D Shaping into a future in-vehicle AR system.

Figure 5.29 demonstrates how 3D Shaping could be integrated into a future in-vehicle AR system. The optimizer and the renderer in the 3D Shaping workflow are separated and distributed into the ADAS domain and the telematics domain, respectively. For each new object, the optimizer would first minimize the energy function in Eq. 4.13 and then send out the optimized latent shape variables to the sensor fusion interface, where they would be tracked according to the shape consistency. The sensor fusion interface would pack the two additional latent variables into an object list and send the list to the telematics gateway through the vehicle backbone bus. The payload on the backbone would only increase by the size of two floating points multiplied by the number of objects being tracked.

We analyzed the payload requirements for transmitting 3D geometries via in-vehicle buses. Assuming the 3D shaping optimizer operates at 25 frames per second and at most 32 objects can be simultaneously tracked, we calculate the required payload using different representations of object geometry. The size of the SDF is set to be $40 \times 40 \times 40$ and single-precision floating point is used. The result is presented in Tab. 5.3, where FE indicates Fast Ethernet with a transfer rate of 100 Mbps, and N-GE indicates Gigabit Ethernet with N Gbps transfer rate. The result clearly shows the advantage of using latent shape representation. In fact, a vehicle backbone will be filled with all different signals for inter-domain communication. Every free byte on the backbone is considered a critical resource. 3D Shaping is one of the most practical solutions to fully exploit 3D object-level reconstruction for in-vehicle augmented reality, with the minimum sensor requirement of only one monocular camera. Therefore, we are firmly convinced that 3D Shaping is another key enabler for incorporating AR in a future production car.

Table 5.3: Payload analysis for transmitting 3D geometries.

	SDF	Face + Vertex	Latent Shape
Single Object	250 KB	$\sim$ 73 KB	8 B
Required Payload	$\sim$ 195 MB/s	$\sim$ 57 MB/s	6.25 KB/s
Required Backbone	10-GE	1-GE	FE
Cost	High	Medium	Low

5.5.4 Future Challenges and Opportunities

As the number of sensors integrated into a future in-vehicle AR system grows, it becomes harder to keep the entire system running at low latency, even if each component of the system operates in real-time. A simple example of the problem in shown in Fig. 5.30, where a camera, a Radar and an IMU have sensed a real world event at the same time. The processing of camera and Radar measurements would cause significant delay compared to the processing of IMU measurements. As a result, the sensor interface would receive a camera or a Radar measurement "from the past", yielding non-optimal fusion results.

In order to appropriately use this so-called Out Of Sequence Measurement (OOSM), all system states in the past have to be stored, from the sensing time point to the arrival time point of the OOSM. This requires a large memory space in the ECU where sensor fusion is carried out. Also, due to frequent matrix inversions by OOSM update, additional computational power and memory bandwidth are indispensable. These will increase the difficulty of incorporating augmented reality in a production car.

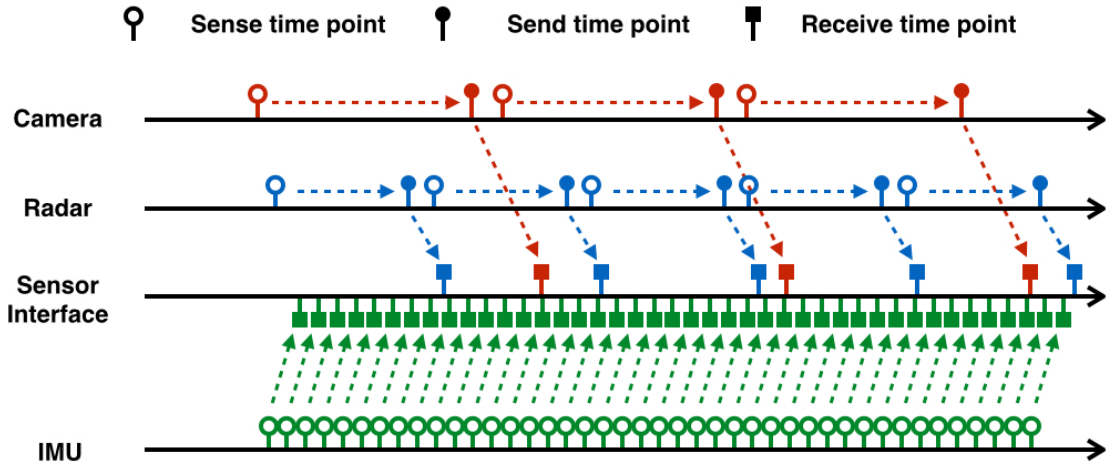


Figure 5.30: Example of OOSM problem.

Artificial Intelligence, particularly deep-learning-based techniques for depth and scene understanding have been boosted in the past years and will continue developing. The intelligence of a car would be in another dimension in the future, allowing it comprehensively perceive the surroundings, communicating with infrastructures and other road users, and even driving itself. Being freed from the otherwise tedious driving tasks, a driver would enjoy the mixture of the virtual and the real world throughout the journey. Also, new display technologies such as Organic LED (OLED) and 3D holograph would enable a new series of HMIs for augmented reality. One day in the future, we could interact with the AR system in a way far beyond touch, voice, or gesture control. All these together lead in-vehicle augmented reality towards a bright future.

5.6 Summary and Future Work

In this chapter, we proposed three designs of in-vehicle augmented reality systems, respectively for the current generation, next generation, and future generation of production cars.

We integrated the current generation AR system into a Mercedes-Benz R-Class production car, with the requirement of only using GPS-based positioning techniques. The system was centralized, used one front-looking monocular camera as input, and showed AR blended

images on a central LCD and a rear-seat touch screen. We demonstrated AR driver assistance and AR passenger infotainment in the R-Class prototype vehicle.

We built another AR prototype vehicle by modifying a Mercedes-Benz S-Class production car for the next generation in-vehicle AR system. The system was distributed into two desktop workstations simulating the ADAS domain and the telematics domain, respectively. We built them into the trunk of the vehicle while keeping the interior as unmodified as possible. In the next generation AR system, we made full use of depth understanding through stereo vision which supported AR functions by localization, tracking, and rendering. We showed that our proposed Stixel compression algorithm is a key enabler for further integrating AR into the E/E architecture of the next generation of production cars.

Based on the lessons learned during our development of the R-Class and the S-Class prototype vehicle, we proposed an AR system designed for future series production. We envisioned the future in-vehicle AR system to fully exploit semantic scene understanding for a various number of AR features. We also show that our proposed 3D Shaping algorithm is another key enabler which could enable truly 3D augmentation of the real world. With fast booming deep learning and display technologies, we firmly believe that in-vehicle AR will have a bright future.

6 Concluding Remarks

6.1 Summary of the Thesis

In this thesis, we presented a roadmap toward practical implementations of in-vehicle augmented reality. We proposed two key enablers, namely Stixel Compression and 3D Shaping which solve the key technical problems for incorporating augmented reality in a car. In addition, we successfully tackled the key engineering problem to prove the concept of AR in test vehicles. Based on our findings and lessons learned during the development, we designed three AR systems, respectively for the current, the next, and the future generation of production cars. The current and the next generation of our designed AR system were integrated and demonstrated in an R-Class and an S-Class prototype vehicle, respectively. We expect our proposed future generation AR system to be seriously considered by industrial decision makers and eventually be adopted in series productions.

In Chapter 3, we proposed a lossless compression algorithm for Stixel as a key solution to the problem of transmitting depth information via a low- or medium-bandwidth in-vehicle communication system. The lossless compression algorithm comprises a predictive modeling stage and an entropy coding stage. During predictive modeling, a Stixel Column to be encoded is first predicted by references in its spatial and temporal neighborhood. Subsequently, the resulted residual column is encoded through Golomb coding, an entropy coding method which is optimal for data sources following geometric distribution. Experiment results show that our proposed algorithm achieved nearly 72% of average space savings, which almost reached the upper bound of the theoretical compression limit on our recorded dataset. Our analysis indicated that without Stixel compression, the more expensive FlexRay bus is required for transmitting raw Stixel data from the ADAS domain to the telematics domain in order to support AR functions based on depth understanding. Relying on our proposed Stixel compression algorithm, the depth information can be transmitted through a CAN bus without modifying the current generation of in-vehicle E/E architecture.

In Chapter 4, we proposed a novel workflow named 3D Shaping as the first step toward object-level 3D reconstruction, which is key to solving the 3D registration problem for augmented reality. The proposed monocular 3D Shaping only requires a single image from a monocular camera as input, and it is able to jointly estimate the pose, the scale, and the 3D geometry of an object in the image. The workflow comprises the first stage of 2D appearance detection and the second stage of 3D shape optimization. The first

stage exploits a state-of-the-art deep neural network to semantically segment an input image at a pixel level. The second stage adapts an existing silhouette-based reconstruction technique which takes advantage of an extremely low-dimensional latent space to represent 3D geometries. Theoretically, only two additional parameters are needed to reconstruct a complex 3D geometry without any information loss. Our evaluation shows a nearly 20 percent performance gain of 3D reconstruction in viewpoint accuracy. Relying on the extremely low-dimensional latent representation of 3D geometries, 3D Shaping can be distributed into the ADAS domain and the telematics domain in the future in-vehicle E/E architecture, which enables truly 3D augmentation of the real world.

Furthermore, we proposed two extensions to monocular 3D Shaping in Chapter 4, which makes it more practical for traffice scenes. The proposed improvements made use of additional 3D sensors in a car, e.g., a stereo camera or a Lidar. The first extension we proposed is a novel neural network architecture named Pose-RCNN which is able to jointly detect objects in a scene and estimate a viewpoint angle for each detected object. Evaluation of our proposed Pose-RCNN showed competitive results against other state-of-the-art approaches. The second extension we proposed is called 3D Shaping + Lidar, where we used an additional point cloud energy to optimize 3D geometries directly in the 3D space. This enabled us to reconstruct multiple object instances within the same class by heavy self-occlusion. Evaluation results show a significant improvement in pose and occupancy bounding box estimation against the monocular approach.

Figure 6.1: Vision of in-vehicle augmented reality. Image source: Daimler AG.

In Chapter 5, we tackled the final key problem of this thesis by proving the concept of in-vehicle augmented reality through three designed AR systems. we first designed a modest AR system only using GPS-based positioning techniques, and we integrated the system into a Mercedes-Benz R-Class prototype vehicle. The AR system was a centralized system, and it used a single front-looking monocular camera as the system input plus a central LCD and a rear-seat touch screen as the system output. We demonstrated AR driver assistance and AR passenger infotainment in the R-Class prototype vehicle. During our

road test, we observed that the GPS-based positioning algorithm was not robust enough in curves and roundabouts. We compensated the "drifting" effect of POIs through a "clever" HMI design and a sophisticated POI retrieval mechanism in the R-Class.

In order to resolve the issues we had in the first AR prototype system, we built another AR prototype vehicle by modifying a Mercedes-Benz S-Class production car which was equipped with powerful in-vehicle sensors and provided more options of display area for in-vehicle AR. The new AR system was distributed into two desktop workstations simulating the ADAS domain and the telematics domain, respectively. Almost all additional hardware components were built into the trunk of the car so that the vehicle interior remained as unmodified as possible. We fully exploited depth understanding through stereo vision, and we integrated AR functions that made use of depth information to enable precise localization, robust tracking, and realistic rendering. In addition, we showed the advantage of our proposed Stixel compression scheme for the transmission of depth information through a low- or medium-bandwidth in-vehicle bus.

Based on the lessons learned during our development of the R-Class and the S-Class prototype vehicle, we proposed a future AR system designed for series production in the future. We presented the functional modules, software components, and the E/E architecture of our designed future AR system in detail. We also presented the scheme for the integration of our proposed 3D Shaping workflow into the AR system, which will enable 3D reconstruction of the surroundings at an object level and boost the performance of AR functions such as road estimation, ego-motion estimation, and depth culling.

6.2 Future Outlook

Today, Artificial Intelligence (AI) is revolutionizing the modern society. AI-boosted products and applications start entering our daily life, influencing a range of activities including learning, trading, and moving from A to B. As AI becomes more and more powerful in the future, cars would eventually be able to drive themselves, and drivers would be emancipated from the tedious task of driving and enjoy the ride in the "third place" between their homes and their offices. This sets the stage for in-vehicle augmented reality to really demonstrate its value.

In the future, in-vehicle AR would turn the ride of the no-more-driving drivers into quality time from three different perspectives. First, as the OLED display technology becomes mature, every interior surface of the car could become a display for augmented reality. People could see augmented information through the windshield, the dashboard, and the doors. The entire interior could even become an AR room while the car is driving itself. Second, the interaction between the passengers and the car would be revolutionized relying on advanced sensor technologies. Gesture control, gaze control, natural language control could be enabled, bringing the user experience with the in-vehicle AR system to another level. Last but not least, AR contents would become more personalized relying on artificial intelligence. Content providers could push the most interested AR contents

to a driver based on his or her driving habits and daily routines. All of these would make in-vehicle augmented reality more connected, more sophisticated, and more "real".

From a technical viewpoint, an in-vehicle AR system in the future would fully exploit depth and semantic scene understanding in order to further boost localization, tracking, and rendering. The proposed Stixel compression algorithm in Chapter 3 could be further improved by using different reference areas or combining lossy and lossless compression. If the instant peak of the compressed Stixel stream could be further cut down, more stereo cameras (backward-looking, side-looking) could be connected to the AR system without necessarily modifying the in-vehicle communication system. This would truly enable 360° depth understanding for in-vehicle AR. Also, the proposed 3D Shape reconstruction workflow in Chapter 4 could be further developed by incorporating instance-aware segmentation techniques or 3D neural networks. Semantic Stixels could be an alternative input to the 3D Shaping reconstruction pipeline. Once the complete scene of the 360° surroundings is reconstructed at an object level, any kind of AR applications such as see-through navigation carpet and real-time AR shooting game could be realized. Moreover, Car2X communication could be a new fundamental input source to the future AR system, enabling features beyond the scope of depth and scene understanding. Ethernet backbone would become the main carrier medium for transmitting massive amount of data that are required by the AR system from one vehicle domain to the other.

Nevertheless, these visions would not become reality if we stop pushing forward the frontier of our research in augmented reality. We hope this thesis could catch the attention of our peers in both academic research and industry so that more researchers and developers would devote themselves to augmented reality. We firmly believe in a bright future of in-vehicle AR.

A Latent Shape Training Software

In Section 4.3.1, we introduced the latent shape space where 3D geometries are embedded. In order to train a latent shape space, we gather a small number of training samples (CAD models of cars), calculate the feature points out of them, and embed the feature points into a lower-dimensional latent space using GPLVM. Figure A.1 illustrates again how a 3D CAD model is "translated" into a latent variable and vice versa.

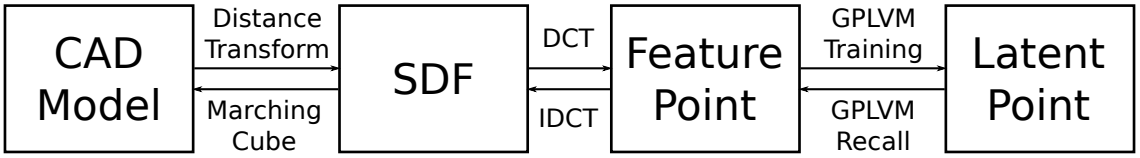


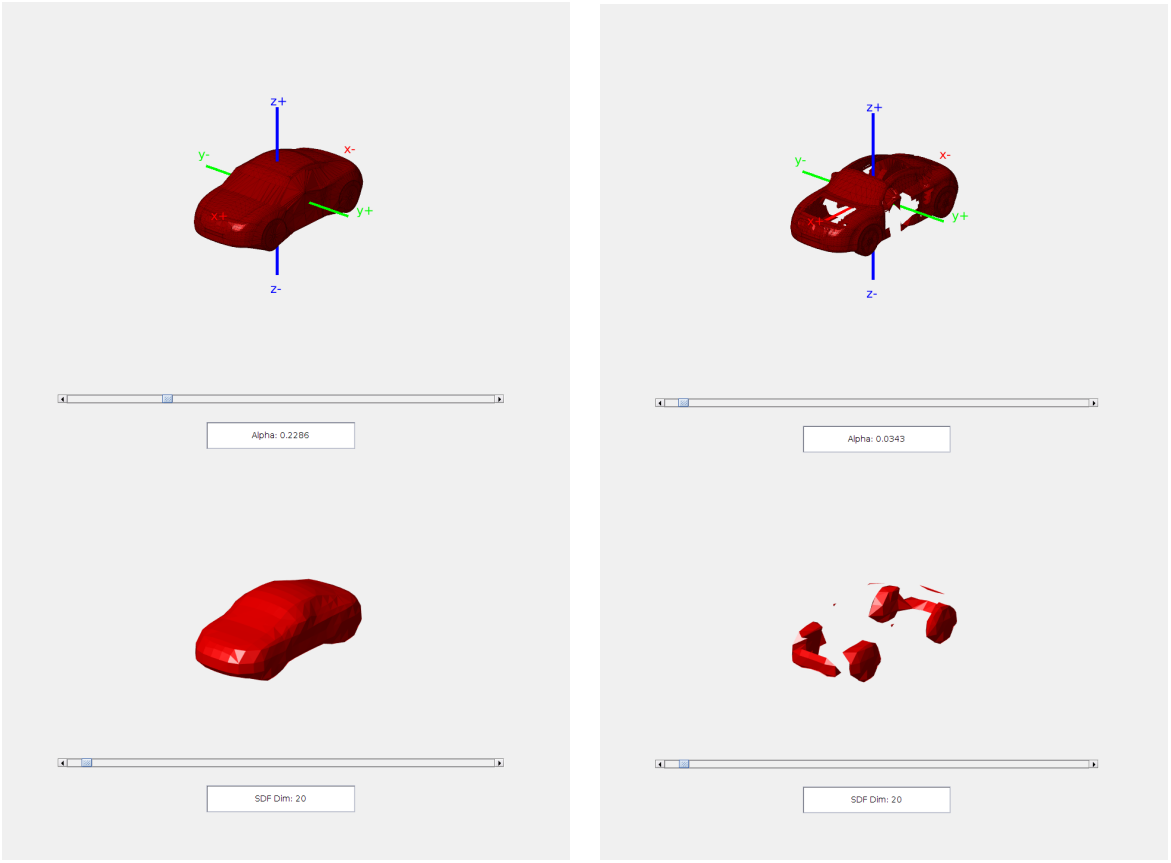
Figure A.1: From 3D CAD model to latent variable.

In order to facilitate the training process and reach a certain level of automation, we developed a Graphical User Interface (GUI) using Matlab. It comprises a panel for data preparation (Fig. A.3(a)) and a panel for GPLVM training (Fig. A.3(b)). As the names suggest, the data panel provides functionalities to compute feature points out of CAD models, whereas the GPLVM panel mainly focuses on tuning training parameters and visualizing the trained latent space.

We start a training process by preparing training samples. Each training sample is a vectorized SDF that encodes a 3D geometry of a specific class, e.g., *car*. First, we pick up a small number of public available CAD models of cars and load them into Matlab one after another. Each loaded CAD model is approximated through its alpha shape in order to reduce the complexity of the geometry and remove irrelevant interior faces. We implemented a slider for alpha radius, a non-negative scalar that controls the convexity of the approximated geometry. A larger alpha radius generates a more convex 3D surface, whereas a smaller radius results in a closer approximation to the original geometry, though leaving holes and discontinuities inside the model (Fig. A.2). Subsequently, an SDF is calculated from an approximated alpha shape using the implementation of [Bat15].

We only need a small number of training samples for GPLVM training. The more diverse the training samples are, the better coverage of the latent shape space can be achieved. For example, we are not able to reconstruct a truck out of a latent shape space that is trained with only passenger cars. After a set of training samples are prepared, we use the GPLVM panel to adjust training parameters and initialize a training process. The GPLVM panel visualizes the variation of the latent shape space during the training

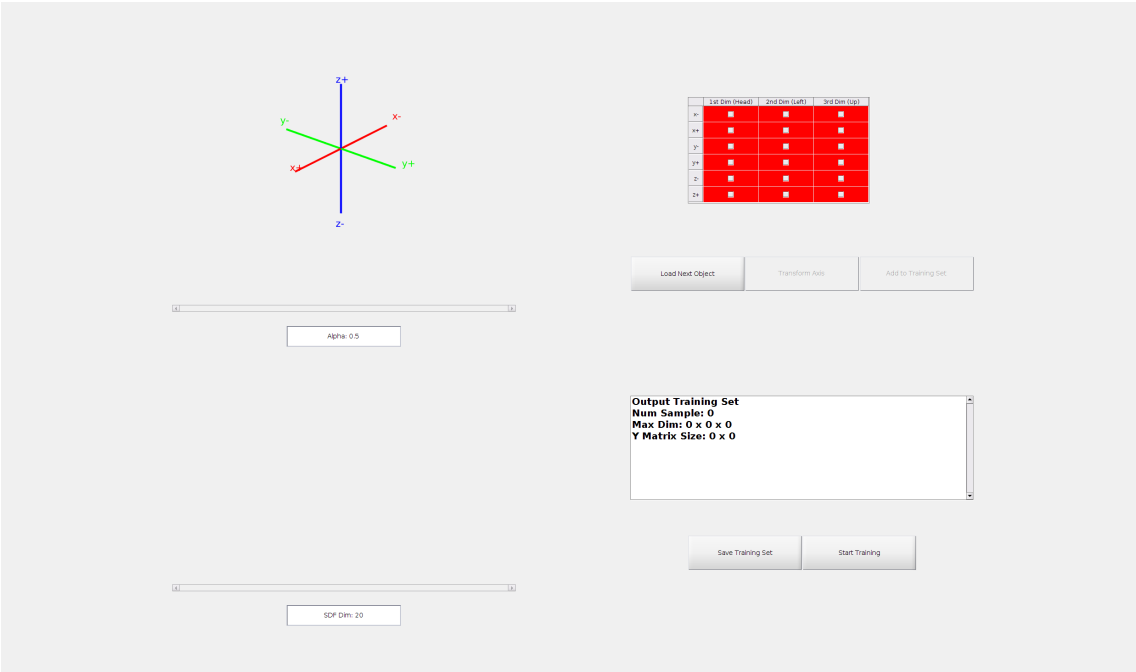
process. The actual optimization during GPLVM training is implemented by Sheffield Machine Learning Group [She13]. When the training process is finished, we can generate and export new 3D geometries by clicking at an arbitrary spot in the latent space inside the panel. We can also change the color and the dimension of a generated 3D geometry using the GPLVM panel.



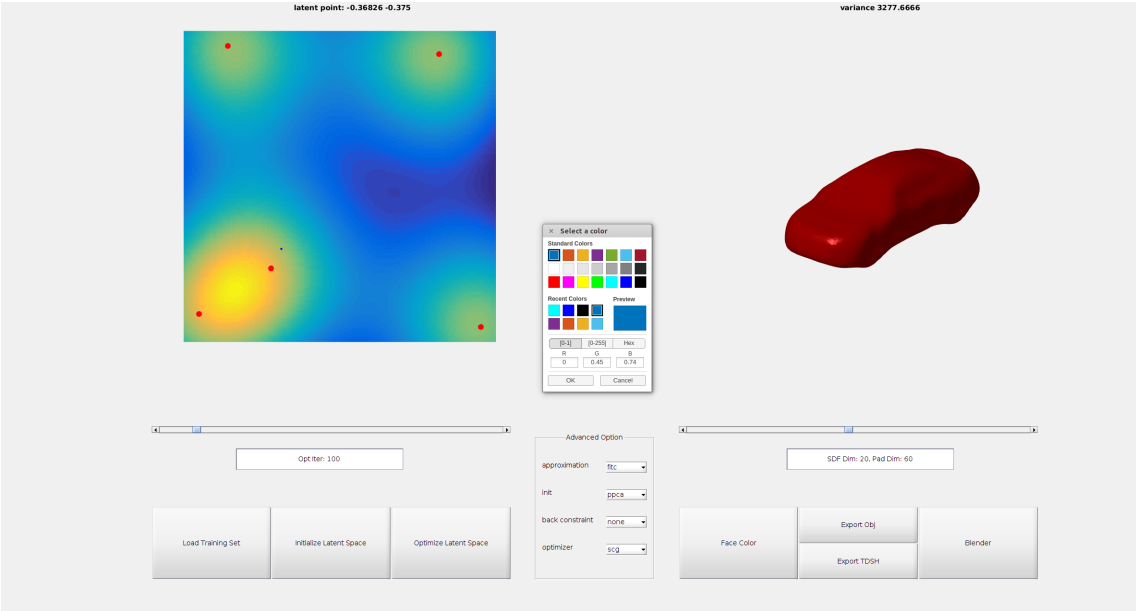
(a) Larger alpha radius. (b) Smaller alpha radius.

Figure A.2: Alpha shape slider.

The output of a training process is a GPLVM model containing a set of latent variables and the hyper-parameters of the Gaussian Process that maps the latent variables to their corresponding training samples. During 3D Shaping optimization, we search the optimal shape within the trained latent space instead of the high-dimensional feature space. This makes 3D Shaping optimization highly efficient.



(a) Panel for data preparation.



(b) Panel for GPLVM training.

Figure A.3: GUI of the latent shape training software.

B Derivatives of 3D Shaping Energy

In Section 4.3.2, we introduced the energy function (Eq. 4.9) for monocular 3D shape reconstruction. The energy function is differentiable with respect to the latent variable and the viewpoint parameters, and it can be minimized using gradient-based non-linear optimization methods. Here, we provide the mathematical details how the partial derivatives of the energy function are derived. The notations used in this appendix are given in Tab. B.1. The partial derivatives with respect to the optimization targets (3 variables for translation, 4 variables for rotation, 1 variable for scale, 2 additional latent variables for 3D geometry) are decomposed by using the chain rule, as expressed in Eq. B.1. The transformation of coordinate systems (Section 2.3) is given in Eq. B.2.

$$\frac{\partial E}{\partial \rho} = \frac{\partial E}{\partial \Phi} \cdot \frac{\partial \Phi}{\partial {}^{sdf}\mathbf{X}} \cdot \frac{\partial {}^{sdf}\mathbf{X}}{\partial {}^{ndc}\mathbf{X}} \cdot \frac{\partial {}^{ndc}\mathbf{X}}{\partial \rho} \tag{B.1}$$

$${}^{o}\mathbf{X} \xrightarrow{M} {}^{c}\mathbf{X} \xrightarrow{P} {}^{ndc}\mathbf{X} \xrightarrow{\text{trivial}} {}^{sdf}\mathbf{X} \tag{B.2}$$

Table B.1: Notations used for 3D Shaping derivatives.

Symbol	Definition
E	Energy (negative log-likelihood).
Ω	Region of interest on image.
π	Projection function.
Φ	Signed distance function.
P_f	Foreground probability.
P_b	Background probability.
L	Ray.
ρ	Optimization target.
${}^{o}\mathbf{X}$	Point in object geometry coordinate.
${}^{c}\mathbf{X}$	Point in camera coordinate.
${}^{ndc}\mathbf{X}$	Point in normalized device coordinate.
${}^{sdf}\mathbf{X}$	Point in SDF cube coordinate.
d	Dimension of a SDF cube.
P	Projection matrix.
M	Modelview matrix.
J_M	Modelview Jacobian matrix.

Step 1: $\partial E/\partial \Phi$ The energy E is a function of a projection π (Eq. B.3), which is again a function of an SDF cube Φ (Eq. B.4).

$$E(\pi) = -\sum_{\Omega} \ln\left[\pi P_f + (1-\pi)P_b\right] \tag{B.3}$$

$$\pi(\Phi) = 1 - \exp\left\{-\sum_{L} \ln\left(1 + e^{\Phi\zeta}\right)\right\} \tag{B.4}$$

By applying the chain rule, we have

$$\frac{\partial E}{\partial \Phi} = \frac{\partial E}{\partial \pi} \cdot \frac{\partial \pi}{\partial \Phi} \tag{B.5}$$

where

$$\frac{\partial E}{\partial \pi} = -\sum_{\Omega} \frac{P_f - P_b}{\pi P_f + (1-\pi)P_b} \tag{B.6}$$

and

$$\frac{\partial \pi}{\partial \Phi} = \exp\left\{-\sum_{L} \ln\left(1 + e^{\Phi\zeta}\right)\right\} \sum_{L} \frac{\zeta e^{\Phi\zeta}}{1 + e^{\Phi\zeta}} \tag{B.7}$$

$$= (1-\pi) \sum_{L} \frac{\zeta e^{\Phi\zeta}}{1 + e^{\Phi\zeta}} \tag{B.8}$$

Therefore,

$$\frac{\partial E}{\partial \Phi} = -\sum_{\Omega} \frac{(P_f - P_b)(1-\pi)}{\pi P_f + (1-\pi)P_b} \sum_{L} \frac{\zeta e^{\Phi\zeta}}{1 + e^{\Phi\zeta}} \tag{B.9}$$

Note that P_f and P_b are estimated using a segmentation neural network.

Step 2: $\partial \Phi / \partial\, {}^{sdf}\mathbf{X}$ This derivative is numerically computed using Eq. B.10.

$$\frac{\partial \Phi}{\partial\, {}^{sdf}\mathbf{X}} = \frac{\Phi(\mathbf{X} + \Delta\mathbf{X}) - \Phi(\mathbf{X} - \Delta\mathbf{X})}{2\Delta\mathbf{X}} \tag{B.10}$$

Step 3: $\partial\, {}^{sdf}\mathbf{X} / \partial\, {}^{ndc}\mathbf{X}$ This derivative is constant and thus trivial, for the reason that

$${}^{sdf}\mathbf{X} = \left({}^{ndc}\mathbf{X} + \frac{1}{2}\right) \cdot d \tag{B.11}$$

Step 4: $\partial\, {}^{ndc}\mathbf{X} / \partial \rho$ First, we write an NDC point ${}^{ndc}\mathbf{X}$ as the projection of a camera point ${}^{c}\mathbf{X}$:

$${}^{ndc}\mathbf{X} = P\, {}^{c}\mathbf{X} \tag{B.12}$$

$$\begin{bmatrix} {}^{ndc}X \\ {}^{ndc}Y \\ {}^{ndc}Z \\ 1 \end{bmatrix} \sim \begin{bmatrix} p_0 & 0 & p_8 & 0 \\ 0 & p_5 & p_9 & 0 \\ 0 & 0 & p_{10} & p_{14} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{bmatrix} \tag{B.13}$$

$$
\begin{aligned}
{}^{ndc}X &= p_0 \frac{{}^{c}X}{{}^{c}Z} + p_8 && \text{(B.14)}\\
{}^{ndc}Y &= p_5 \frac{{}^{c}Y}{{}^{c}Z} + p_9 && \text{(B.15)}\\
{}^{ndc}Z &= p_{14} \frac{1}{{}^{c}Z} + p_{10} && \text{(B.16)}
\end{aligned}
$$

Their partial derivatives are given by the quotient rule as the coordinates of a camera point are functions of the optimization targets as well.

$$
\begin{aligned}
\frac{\partial\, {}^{ndc}X}{\partial \rho} &= p_0 \cdot \frac{1}{{}^{c}Z^2} \left({}^{c}Z \frac{\partial\, {}^{c}X}{\partial \rho} - {}^{c}X \frac{\partial\, {}^{c}Z}{\partial \rho} \right) && \text{(B.17)}\\
\frac{\partial\, {}^{ndc}Y}{\partial \rho} &= p_5 \cdot \frac{1}{{}^{c}Z^2} \left({}^{c}Z \frac{\partial\, {}^{c}Y}{\partial \rho} - {}^{c}Y \frac{\partial\, {}^{c}Z}{\partial \rho} \right) && \text{(B.18)}\\
\frac{\partial\, {}^{ndc}Z}{\partial \rho} &= p_{14} \cdot \frac{1}{{}^{c}Z^2} \cdot \frac{\partial\, {}^{c}Z}{\partial \rho} && \text{(B.19)}
\end{aligned}
$$

The partial derivatives of a camera point are given by the multiplication of the modelview Jacobian matrix and the corresponding object point. For more details of the Jacobian of the modelview matrix, please refer to [PR12].

$$
{}^{c}\mathbf{X} = M(\rho)\, {}^{o}\mathbf{X} \qquad \text{(B.20)}
$$

$$
\frac{\partial\, {}^{c}\mathbf{X}}{\partial \rho} = J_M(\rho)\, {}^{o}\mathbf{X} \qquad \text{(B.21)}
$$

Bibliography

[ABB+01] Azuma, R., Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre: *Recent Advances in Augmented Reality*. IEEE Computer Graphics and Applications, 21(6):34–47, 2001. 9, 11, 16

[ADF12] Alexe, B., T. Deselaers and V. Ferrari: *Measuring the Objectness of Image Windows*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2189–2202, 2012. 63

[APTB+14] Arbeláez, P., J. Pont-Tuset, J. T. Barron, F. Marques and J. Malik: *Multiscale Combinatorial Grouping*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 328–335, 2014. 63

[AS16] Alakuijala, J. and Z. Szabadka: *Brotli Compressed Data Format*. `https://tools.ietf.org/html/rfc7932`, 2016. [Online; accessed 31-08-2017]. 39

[ASK+05] Anguelov, D., P. Srinivasan, D. Koller, S. Thrun, J. Rodgers and J. Davis: *SCAPE: Shape Completion and Animation of People*. ACM Transactions on Graphics, 24(3):408–416, 2005. 61

[Bad04] Badino, H.: *A Robust Approach for Ego-Motion Estimation Using a Mobile Stereo Platform*. In *Proceedings of the 1st International Conference on Complex Motion (IWCM)*, pages 198–208, 2004. 5, 15

[Bat15] Batty, C.: *SDFGen*. `https://github.com/christopherbatty/SDFGen`, 2015. [Online; accessed 31-08-2017]. 127

[BB95] Beauchemin, S. S. and J. L. Barron: *The Computation of Optical Flow*. ACM Computing Surveys, 27(3):433–466, 1995. 15

[Bed00] Bednar, J. C.: *Li Proberbe Au Vilain: A Critical Edition*. University Press of the South, New Orleans, USA, 2000. 93

[Ber08] Bergmeier, U.: *Augmented Reality in Vehicles – Technical Realisation of a Contact-Analogue Head-Up Display under Automotive Capable Aspects; Usefulness Exemplified through Night Vision Systems*. In *FISITA World Automotive Congress*, 2008. 4

[BF12] Burlet, J. and M. D. Fontana: *Robust and Efficient Multi-Object Detection and Tracking for Vehicle Perception Systems Using Radar and Camera*

Sensor Fusion. In *Proceedings of IET and ITS Conference on Road Transport Information and Control (RTIC)*, pages 1–6, 2012. 15

[BFP09] Badino, H., U. Franke and D. Pfeiffer: *The Stixel World – A Compact Medium Level Representation of the 3D World*. In *Proceedings of the 31st Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM) Symposium [German Conference on Pattern Recognition (GCPR)]*, pages 51–60, 2009. 10, 35, 37

[BHL15] Beyer, L., A. Hermans and B. Leibe: *Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels*. In *Proceedings of German Conference on Pattern Recognition (GCPR)*, pages 157–168, 2015. 62, 63

[BM06] Bartie, P. J. and W. A. Mackaness: *Development of a Speech-Based Augmented Reality System to Support Exploration of Cityscape*. Transactions in GIS, 10(1):63–86, 2006. 2

[BMW11a] BMW Group: *BMW Vision ConnectedDrive*. `https://www.press.bmwgroup.com/usa/article/detail/T0097647EN_US/bmw-vision-connecteddrive`, 2011. [Online; accessed 31-08-2017]. 5

[BMW11b] BMW Group: *Head-up Display 2.0 – Augmented Reality*. `http://www.bmwblog.com/2011/10/07/head-up-display-2-0-augmented-reality`, 2011. [Online; accessed 31-08-2017]. 5

[BR05] Bimber, O. and R. Raskar: *Spatial Augmented Reality: Merging Real and Virtual Worlds*. A K Peters, Ltd., 2005. 16

[BRWF16] Braun, M., Q. Rao, Y.-K. Wang and F. Flohr: *Pose-RCNN: Joint Object Detection and Pose Estimation Using 3D Object Proposals*. In *Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1546–1551, 2016. 22

[CDA+05] Calhoun, G. L., M. H. Draper, M. F. Abernathy, F. Delgado and M. Patzek: *Synthetic Vision System for Improving Unmanned Aerial Vehicle Operator Situation Awareness*. Proceedings of SPIE, 5802:219–230, 2005. 2

[CF13] Cashman, T. and A. Fitzgibbon: *What Shape are Dolphins? Building 3D Morphable Models from 2D Images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1):232–244, 2013. 61

[CKZ+15] Chen, X.-Z., K. Kundu, Y.-K. Zhu, A. Berneshawi, H.-M. Ma, S. Fidler and R. Urtasun: *3D Object Proposals for Accurate Object Class Detection*. In *Advances in Neural Information Processing Systems (NIPS)*, pages 424–432, 2015. 63, 64, 83, 84, 85

[CPK+14] Chen, L.-C., G. Panpandreou, I. Kokkinos, K. Murphy and A. L. Yuille: *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. arXiv preprint, 2014. 63

[CPMA11] Charissis, V., S. Papanastasiou, L. Mackenzie and S. Arafat: *Evaluation of Collision Avoidance Prototype Head-Up Display Interface for Older Drivers*. In *Proceedings of International Conference on Human-Computer Interaction (HCI)*, 2011. 4

[CRS+17] Cordts, M., T. Rehfeld, L. Schneider, D. Pfeiffer, M. Enzweiler, S. Roth, M. Pollefeys and U. Franke: *The Stixel World: A Medium-Level Representation of Traffic Scenes*. Image and Vision Computing, 68:40–52, 2017. 38

[CS12] Carreira, J. and C. Sminchisescu: *CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7):1312–1328, 2012. 63

[CT11] Copenhagen Institute of Interaction Design and Toyota Motor Corporation: *Toyota Opens Window to the World*. `http://ciid.dk/press-release-toyota-opens-window-to-the-world`, 2011. [Online; accessed 31-08-2017]. 3, 97

[CTCG95] Cootes, T. F., C. J. Taylor, D. H. Cooper and J. Graham: *Active Shape Models – Their Training and Application*. Computer Vision and Image Understanding, 61(1):38–59, 1995. 61

[CVT+12] Caraffi, C., T. Vojíř, J. Trefný, J. Šochman and J. Matas: *A System for Real-Time Detection and Tracking of Vehicles from a Single Car-mounted Camera*. In *Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 975–982, 2012. 15

[DABP14] Dollár, P., R. Appel, S. Belongie and P. Perona: *Fast Feature Pyramids for Object Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(8):1532–1545, 2014. 64, 83

[Dai12] Daimler AG: *The Car as a Medium*. `https://www.mercedes-benz.com/en/mercedes-benz/design/the-car-as-a-medium`, 2012. [Online; accessed 31-08-2017]. 3, 4

[Dai13] Daimler AG: *Neue C-Klasse Erstmals mit Head-up Display [New C-Class with Head-up Display for the First Time]*. `http://blog.mercedes-benz-passion.com/2013/11/neue-c-klasse-erstmals-mit-head-up-display`, 2013. [Online; accessed 31-08-2017]. 3, 5

[DAK+15] Dickmann, J., N. Appenrodt, J. Klappstein, H.-L. Blöcher, M. Muntzinger, A. Sailer, M. Hahn and C. Brenk: *Making Bertha See Even More: Radar Contribution*. IEEE Access, 3:1233–1247, 2015. 5

[DB08] Du, J. and M. J. Barth: *Next-Generation Automated Vehicle Location Systems: Positioning at the Lane Level.* IEEE Transactions on Intelligent Transportation Systems, 9(1):48–57, 2008. 14

[DCT09] Doshi, A., S. Y.-H. Cheng and M. M. Trivedi: *A Novel Active Heads-up Display for Driver Assistance.* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39(1):85–93, 2009. 4

[DDWL10] Ding, J., X. Du, X. Wang and J. Liu: *Improved Real-Time Correlation-based FPGA Stereo Vision System.* In *Proceedings of International Conference on Mechatronics and Automation (ICMA)*, pages 104–108, 2010. 37

[Des23] Desiderius Erasmus Roterodamus: *Adagia.* Joa. Frobenius, 1523. 35

[dLLT12] dos Santos, A. L., D. Lemos, J. E. F. Lindoso and V. Teichrieb: *Real Time Ray Tracing for Augmented Reality.* In *Proceedings of 14th Symposium on Virtual and Augmented Reality (SVR)*, pages 131–140, 2012. 5, 16

[DNC+01] Dissanayake, M. W. M. G., P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba: *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem.* IEEE Transactions on Robotics and Automation, 17(3):229–241, 2001. 5, 15

[DPRR13] Dame, A., V. Prisacariu, C. Ren and I. Reid: *Dense Reconstruction Using 3D Object Shape Priors.* In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1288–1295, 2013. 5, 86

[DT05] Dalal, N. and B. Triggs: *Histograms of Oriented Gradients for Human Detection.* In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005. 64

[EEv+15] Everingham, M., S. M. A. Eslami, L. van Gool, C. K. I. Williams, J. Winn and A. Zisserman: *The Pascal Visual Object Classes Challenge: A Retrospective.* International Journal of Computer Vision, 111(1):98–136, 2015. 63, 64

[EF15] Eigen, David and Rob Fergus: *Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture.* In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, 2015. 62

[ESU09] Engel, D., T. Stütz and A. Uhl: *A Survey on JPEG2000 Encryption.* Multimedia Systems, 15(4):243–270, 2009. 39

[Exp16] Explorental LLC: *AR Circuits.* `http://arcircuits.com`, 2016. [Online; accessed 31-08-2017]. 2

[FB81] Fischler, M. A. and R. C. Bolles: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Communications of the ACM, 24(6):381–395, 1981. 15

[FGMR10] Felzenszwalb, P., R. Girshick, D. McAllester and D. Ramanan: *Object Detection with Discriminatively Trained Part-based Models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010. 64

[FPR+13] Franke, U., D. Pfeiffer, C. Rabe, C. Knöppel, M. Enzweiler, F. Stein and R. G. Herrtwich: *Making Bertha See.* In *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1–10, 2013. 4, 5

[Fra01] Frank Baum, L.: *The Master Key.* Bowen-Merrill Company, Indianapolis, 1901. 2

[FRBG05] Franke, U., C. Rabe, H. Badino and S. Gehrig: *6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception.* In *Proceedings of the 27th Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM) Symposium [German Conference on Pattern Recognition (GCPR)]*, pages 216–223, 2005. 15, 111, 112

[Fri02] Friedrich, W.: *ARVIKA – Augmented Reality for Development, Production and Service.* In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 3–4, 2002. 3

[GA95] Gailly, J.-L. and M. Adler: *zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library*, 1995. 39

[GBR14] Gruyer, D., R. Belaroussi and M. Revilloud: *Map-Aided Localization with Lateral Perception.* In *Proceedings of IEEE Intelligent Vehicle Symposium (IV)*, pages 674–680, 2014. 5, 17

[GDDM14] Girshick, R., J. Donahue, T. Darrell and J. Malik: *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.* In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. 62, 64

[GEM09] Gehrig, S., F. Eberli and T. Meyer: *A Real-Time Low-Power Stereo Engine Using Semi-Global Matching.* In *Proceedings of International Conference on Computer Vision Systems (ICVS)*, pages 134–143, 2009. 5, 9, 35, 37

[GFK14] Gabbard, J. L., G. M. Fitch and H. G. Kim: *Behind the Glass: Driver Challenges and Opportunities for AR Automotive Applications.* Proceedings of the IEEE, 1202(2):124–136, 2014. 4

[GGAM14] Gupta, S., R. Girshick, P. Arbeláez and J. Malik: *Learning Rich Features from RGB-D Images for Object Detection and Segmentation.* In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 345–360, 2014. 63

[Gir15] Girshick, R.: *Fast R-CNN*. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 62, 64

[GLU12] Geiger, A., P. Lenz and R. Urtasun: *Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 60, 64, 83

[GM85] Geman, S. and D. McClure: *Bayesian Image Analysis: An Application to Single Photon Emission Tomography*. In *Proceedings of Statistical Computing Section of the American Statistical Association*, pages 12–18, 1985. 86

[Gol66] Golomb, S.: *Run-Length Encodings*. IEEE Transactions on Information Theory, 12(3):399–401, 1966. 39

[GSDB07] Gunnarsson, J., L. Svensson, L. Danielsson and F. Bengtsson: *Tracking Vehicles Using Radar Detections*. In *Proceedings of IEEE Intelligent Vehicle Symposium (IV)*, pages 296–302, 2007. 15

[GSZW10] Gabbard, J. L., J. E. Swan, J. Zedlitz and W. W. Winchester: *More than Meets the Eye: An Engineering Study to Empirically Examine the Blending of Real and Virtual Color Spaces*. In *Proceedings of IEEE Virtual Reality Conference (VR)*, pages 79–86, 2010. 5

[GTFC12] George, P., I. Thouvenin, V. Fremont and V. Cherfaoui: *DAARIA: Driver Assistance by Augmented Reality for Intelligent Automobile*. In *Proceedings of IEEE Intelligent Vehicle Symposium (IV)*, pages 1043–1048, 2012. 5, 96

[Gv75] Gallager, R. and D. van Voorhis: *Optimal Source Codes for Geometrically Distributed Integer Alphabets*. IEEE Transactions on Information Theory, 21(2):228–230, 1975. 50

[GWBB09] Guan, P., A. Weiss, A. O. Bălan and M. J. Black: *Estimating Human Shape and Pose from a Single Image*. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1381–1388, 2009. 61

[HAGM14] Hariharan, B., P. Arbelàez, R. Girshick and J. Malik: *Simultaneous Detection and Segmentation*. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 297–312, 2014. 62

[HAGM15] Hariharan, B., P. Arbeláez, R. Girshick and J. Malik: *Hypercolumns for Object Segmentation and Fine-grained Localization*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 447–456, 2015. 63

[Har16] Harman International: *Navdy with Harman*. `https://services.harman.com/industries/automotive-connected-car/navdy`, 2016. [Online; accessed 31-08-2017]. 5

[HB88] Horaud, R. and M. Brady: *On the Geometric Interpretation of Image Contours*. Artificial Intelligence, 37(1–3):333–353, 1988. 61

[HBDS16] Hosang, J., R. Benenson, P. Dollar and B. Schiele: *What Makes for Effective Detection Proposals?* IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(4):814–830, 2016. 63

[HDH03] Haller, M., S. Drab and W. Hartmann: *A Real-Time Shadow Approach for an Augmented Reality Application Using Shadow Volumes*. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 56–65, 2003. 5, 16

[HIG02] Hirschmüller, H., P. R. Innocent and J. Garibaldi: *Real-Time Correlation-based Stereo Vision with Reduced Border Errors*. International Journal of Computer Vision, 47(1):229–246, 2002. 37

[Hir01] Hirschmüller, H.: *Improvements in Real-Time Correlation-based Stereo Vision*. In *Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*, pages 141–148, 2001. 37

[Hir05] Hirschmüller, H.: *Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814, 2005. 5, 9, 35, 37

[HOBS15] Hosang, J., M. Omran, R. Benenson and B. Schiele: *Taking a Deeper Look at Pedestrians*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4073–4082, 2015. 64, 83

[Hor89] Horn, B.: *Shape from Shading*, chapter 4. MIT Press, Cambridge, MA, 1989. 61

[How08] Howard, A.: *Real-Time Stereo Visual Odometry for Autonomous Ground Vehicles*. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3946–3952, 2008. 5, 15

[HU04] Hu, Z.-C. and K. Uchimura: *Real-Time Data Fusion on Tracking Camera Pose for Direct Visual Guidance*. In *Proceedings of IEEE Intelligent Vehicle Symposium (IV)*, pages 842–847, 2004. 5, 95

[Huf52] Huffman, D. A.: *A Method for the Construction of Minimum-Redundancy Codes*. Proceedings of the IRE, 40(9):1098–1101, 1952. 39

[HV93] Howard, P. G. and J. S. Vitter: *Fast and Efficient Lossless Image Compression*. In *Proceedings of Data Compression Conference (DCC)*, pages 351–360, 1993. 39

[HZRS16] He, K., X. Zhang, S. Ren and J. Sun: *Deep Residual Learning for Image Recognition*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 60

[Int03] International Organization for Standardization: *ISO 11898-1: Road Vehicles – Controller Area Network (CAN) – Part 1: Data Link Layer and Physical Signaling*, 2003. 10, 36, 53

[Int04] International Organization for Standardization: *ISO/IEC 15948: Information Technology – Computer Graphics and Image Processing – Portable Network Graphics (PNG): Functional Specification*, 2004. 39

[Int15] International Organization for Standardization: *ISO/IEC 21320-1: Information Technology – Document Container File – Part 1: Core*, 2015. 39

[JMC93] Janin, A. L., D. W. Mizell and T. P. Caudell: *Calibration of Head-Mounted Display for Augmented Reality Applications.* In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 246–255, 1993. 2

[JSD+14] Jia, Y.-Q., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell: *Caffe: Convolutional Architecture for Fast Feature Embedding.* In *Proceedings of the 22nd ACM International Conference on Multimedia (MM)*, pages 675–678, 2014. 74

[Kal60] Kalman, R. E.: *A New Approach to Linear Filtering and Prediction Problems.* Transactions of the ASME – Journal of Basic Engineering, 82(Series D):35–45, 1960. 15

[KCV13] Köhler, P., C. Connette and A. Verl: *Vehicle Tracking Using Ultrasonic Sensors & Joined Particle Weighting.* In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2900–2905, 2013. 15

[KD09] Kim, S.-J. and A. K. Dey: *Simulated Augmented Reality Windshield Display as a Cognitive Mapping Aid for Elder Driver Navigation.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 133–142, 2009. 4

[KK12] Kán, P. and H. Kaufmann: *High-Quality Reflections, Refractions, and Caustics in Augmented Reality and their Contribution to Visual Coherence.* In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99–108, 2012. 16

[KLZN03] Kusuma, A., L. Li, C.-G. Zhu and W.-S. Ng: *Photorealistic Rendering for Augmented Reality Using Environment Illumination.* In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 208–216, 2003. 16

[KSES14] Kholgade, N., T. Simon, A. Efros and Y. Sheikh: *3D Object Manipulation in a Single Photograph Using Stock 3D Models.* ACM Transactions on Graphics, 33(4):1–12, 2014. 61

[KSH12] Krizhevsky, A., I. Sutskever and G. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 60, 62, 64, 74

[KTCM15] Kar, A., S. Tulsiani, J. Carreira and J. Malik: *Category-Specific Object Reconstruction from a Single Image*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1966–1974, 2015. 5

[KWGP13] Kim, H.-I., X.-F. Wu, J. L. Gabbard and N. F. Polys: *Exploring Head-up Augmented Reality Interfaces for Crash Warning Systems*. In *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI)*, pages 224–227, 2013. 4

[Lau94] Laurentini, A.: *The Visual Hull Concept for Silhouette-based Image Understanding*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(2):150–162, 1994. 61

[Law05] Lawrence, L.: *Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models*. The Journal of Machine Learning Research, 6:1783–1816, 2005. 62

[LBBH98] LeCun, Y., L. Bottou, Y. Bengio and P. Haffner: *Gradient-based Learning Applied to Document Recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998. 62

[LBD+89] LeCun, Y., B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel: *Backpropagation Applied to Handwritten Zip Code Recognition*. Neural Computation, 1(4):541–551, 1989. 62

[LC87] Lorensen, W. E. and H. E. Cline: *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*. ACM SIGGRAPH Computer Graphics, 21(4):163–169, 1987. 62, 70

[LFU13] Lin, D.-H., S. Fidler and R. Urtasun: *Holistic Scene Understanding for 3D Object Detection with RGBD Cameras*. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1417–1424, 2013. 63

[LGSB12] Laitinen, H., J. Granström, M. Ström and D. J. Baños: *RTK + System for Precise Navigation in Shadowed Areas*. In *Proceedings of the 6th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, pages 1–7, 2012. 14

[LK81] Lucas, B. D. and T. Kanade: *An Iterative Image Registration Technique with an Application to Stereo Vision*. In *Proceedings of the 7th International Joint Conference on Artificial intelligence (IJCAI)*, volume 2, pages 674–679, 1981. 15

[L'O15] L'Oréal S.A.: *Makeup Genius*. `http://makeupgenius.com.au`, 2015. [Online; accessed 31-08-2017]. 2

[LQY+10] Liu, Z.-L., Y. Qian, L.-Y. Yang, Y. Bo and H. Li: *An Improved Lossless Image Compression Algorithm LOCO-R*. In *Proceedings of International Conference on Computer Design and Applications (ICCDA)*, volume 1, pages 328–331, 2010. 50

[LSD15] Long, J., E. Shelhamer and T. Darrell: *Fully Convolutional Networks for Semantic Segmentation*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 62

[LSL15] Liu, F.-Y., C.-H. Shen and G.-S. Lin: *Deep Convolutional Neural Fields for Depth Estimation from a Single Image*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5162–5170, 2015. 62

[LSvdHR16] Lin, G.-S., C.-H. Shen, A. van dan Hengel and I. Reid: *Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3194–3203, 2016. 63

[LT10] Levinson, J. and S. Thrun: *Robust Vehicle Localization in Urban Environments Using Probabilistic Maps*. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4372–4378, 2010. 5, 15

[Lyt15] LyteShot: *Assassin*. `http://www.lyteshot.com`, 2015. [Online; accessed 31-08-2017]. 2

[Mah09] Mahoney, M.: *Incremental Journaling Backup Utility and Archiver*. `http://mattmahoney.net/dc/zpaq.html`, 2009. [Online; accessed 31-08-2017]. 39

[Man97] Mann, S.: *Wearable Computing: A First Step Toward Personal Imaging*. Computer, 30(2):25–32, 1997. 2

[MCJS13] M. C. Jr. Shall, M. L. Rusch, J. D. Lee J. D. Dawson G. Thomas N. Aksan M. Rizzo: *Augmented Reality Cues and Elderly Driver Hazard Perception*. Human Factors, 55(3):643–658, 2013. 4

[Met13] Metaio GmbH: *Cruising the Augmented City: Mercedes-Benz Presenting Prototype as World Premiere at InsideAR 2013*. `https://web.archive.org/web/20130926220849/http://augmentedblog.wordpress.com/2013/09/26/cruising-the-augmented-city-mercedes-benz`, 2013. [Online; accessed 31-08-2017]. 104

[Mic11] Microsoft Corporation: *Productivity Future Vision 2011*. `https://blogs.msdn.microsoft.com/mspowerutilities/2011/10/28/productivity-future-vision-2011`, 2011. [Online; accessed 31-08-2017]. 3

[Mic16] Microsoft Corporation: *Microsoft HoloLens*. `http://www.microsoft.com/microsoft-hololens`, 2016. [Online; accessed 31-08-2017]. 2

[MIT15] MIT Technology Review: *Magic Leap*. `https://www.technologyreview.com/s/534971/magic-leap`, 2015. [Online; accessed 31-08-2017]. 16

[Mor78] Moré, J. J.: *The Levenberg-Marquardt Algorithm: Implementation and Theory*. In *Numerical Analysis*, pages 105–116, 1978. 71

[Nem18] Nemerson, E.: *Squash Compression Benchmark*. `http://quixdb.github.io/squash`, 2018. Data retrieved on 2018-02-18. 39

[Nia12] Niantic, Inc.: *Ingress*. `http://www.ingress.com`, 2012. [Online; accessed 31-08-2017]. 2

[Nia16] Niantic, Inc.: *Pokémon Go*. `http://www.pokemongo.com`, 2016. [Online; accessed 31-08-2017]. 2

[NNB04] Nistér, D., O. Naroditsky and J. Bergen: *Visual Odometry*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 652–659, 2004. 5, 15

[NS08] Nam, J.-H. and D.-G. Sim: *Lossless Video Coding based on Pixel-wise Prediction*. Multimedia Systems, 14(5):291–298, 2008. 39

[OTNC13] Oswald, M. R., E. Töppe, C. Nieuwenhuis and D. Cremers: *Innovations for Shape Analysis: Models and Algorithms*, chapter 16. Springer, Heidelberg, Germany, 2013. 61

[PDT+09] Plavšić, M., M. Duschl, M. Tönnis, H. Bubb and G. Klinker: *Ergonomic Design and Evaluation of Augmented Reality Based Cautionary Warnings for Driving Assistance in Urban Environments*. In *Proceedings of the International Ergonomics Association (IEA)*, 2009. 4

[PF10] Pfeiffer, D. and U. Franke: *Efficient Representation of Traffic Scenes by Means of Dynamic Stixels*. In *Proceedings of IEEE Intelligent Vehicle Symposium (IV)*, pages 217–224, 2010. 10, 35, 37

[PF11] Pfeiffer, D. and U. Franke: *Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data*. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 51.1–51.12, 2011. 10, 35, 36, 37, 38, 40, 41, 42, 44

[PGS13] Pfeiffer, D., S. Gehrig and N. Schneider: *Exploiting the Power of Stereo Confidences*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 297–304, 2013. 10, 35, 37, 47

[PGSS12] Pepik, B., P. Gehler, M. Stark and B. Schiele: *3D2PM – 3D Deformable Part Models*. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume VI, pages 356–370, 2012. 64

[Pio12] Pioneer Corporation: *AR Navigation*. `http://pioneer.jp/carrozzeria/cybernavi/avic-vh99hud_avic-zh99hud`, 2012. [Online; accessed 31-08-2017]. 5

[PML+10] Pessoa, S., G. Moura, J. Lima, V. Teichrieb and J. Kelner: *Photorealistic Rendering for Augmented Reality: A Global Illumination and BRDF Solution*. In *Proceedings of IEEE Virtual Reality Conference (VR)*, pages 3–10, 2010. 16

[PR12] Prisacariu, V. and I. Reid: *PWP3D: Real-Time Segmentation and Tracking of 3D Objects*. International Journal of Computer Vision, 98(3):333–354, 2012. 133

[PSG+15] Pepik, B., M. Stark, P. Gehler, T. Ritschel and B. Schiele: *3D Object Class Detection in the Wild*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–10, 2015. 62, 63

[PSGS15] Pepik, B., M. Stark, P. Gehler and B. Schiele: *Multi-View and 3D Deformable Part Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(11):2232–2245, 2015. 64, 83, 85

[PSR12] Prisacariu, V., A. Segal and I. Reid: *Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction*. In *Proceedings of Asian Conference on Computer Vision (ACCV)*, pages 593–606, 2012. 5, 60, 62, 64

[RAS13] Rad, R. M., A. Attar and A. Shahbahrami: *A Predictive Algorithm for Multimedia Data Compression*. Multimedia Systems, 19(2):103–115, 2013. 39

[RBM+07] Rusu, R. B., N. Blodow, Z. C. Marton, A. Soos and M. Beetz: *Towards 3D Object Maps for Autonomous Household Robots*. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3191–3198, 2007. 11

[RBW05] Regenbrecht, H., G. Baratoff and W. Wilke: *Augmented Reality Projects in the Automotive and Aerospace Industries*. IEEE Computer Graphics and Applications, 25(6):48–56, 2005. 3

[RC19a] Rao, Q. and S. Chakraborty: *Efficient Lossless Compression for Depth Information in Traffic Scenarios*. 2019. Springer Multimedia Systems. 22

[RC19b] Rao, Q. and S. Chakraborty: *In-Vehicle Object-level 3D Reconstruction of Traffic Scenes*. Submitted, 2019. 22

[RDGF16] Redmon, J., S. Divvala, R. Girshick and A. Farhadi: *You Only Look Once: Unified, Real-Time Object Detection*. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 62

[RGHC14a] Rao, Q., C. Grünler, M. Hammori and S. Chakraborty: *Design Methods for Augmented Reality In-Vehicle Infotainment Systems.* In *Proceedings of the 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014. 22

[RGHC14b] Rao, Q., C. Grünler, M. Hammori and S. Chakraborty: *Stixel on the Bus: An Efficient Lossless Compression Scheme for Depth Information in Traffic Scenarios.* In *Proceedings of International Conference on Multimedia Modeling (MMM)*, volume I, pages 568–579, 2014. 22

[RHGS15] Ren, S.-Q., K.-M. He, R. Girshick and J. Sun: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 62, 64

[RKD16a] Rao, Q., L. Krüger and K. Dietmayer: *3D Shape Reconstruction in Traffic Scenarios Using Monocular Camera and Lidar.* In *Proceedings of Asian Conference on Computer Vision Workshops (ACCVW)*, volume 2, pages 3–18, 2016. 22

[RKD16b] Rao, Q., L. Krüger and K. Dietmayer: *Monocular 3D Shape Reconstruction Using Deep Neural Networks.* In *Proceedings of IEEE Intelligent Vehicle Symposium (IV)*, pages 310–315, 2016. 22

[RMB+09] Rusu, R. B., Z. C. Marton, N. Blodow, A. Holzbach and M. Beetz: *Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments.* In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3601–3608, 2009. 11

[RMWF10] Rabe, C., T. Müller, A. Wedel and U. Franke: *Dense, Robust, and Accurate Motion Field Estimation from Stereo Image Sequences in Real-Time.* In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 582–595, 2010. 15

[RR12] Ren, C. Y. and I. Reid: *A Unified Energy Minimization Framework for Model Fitting in Depth.* In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 72–82, 2012. 62

[RTG+14] Rao, Q., T. Tropper, C. Grünler, M. Hammori and S. Chakraborty: *AR-IVI – Implementation of In-Vehicle Augmented Reality.* In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 3–8, 2014. 22

[RWD12] Roth, R. M., B. H. Wixom and A. Dennis: *System Analysis and Design.* John Wiley & Sons, Hoboken, New Jersey, 5th edition, 2012. 93

[Sch15] Schmidhuber, J.: *Deep Learning in Neural Networks: An Overview.* Neural Networks, 61:85–117, 2015. 62

[SCR+16] Schneider, L., M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys and S. Roth: *Semantic Stixels: Depth is Not*

Enough. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 110–117, 2016. 38

[SDYT09] Sandhu, R., S. Dambreville, A. Yezzi and A. Tannenbaum: *Non-rigid 2D-3D Pose Estimation and 2D Image Segmentation.* In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 786–793, 2009. 62

[SH06] Strobl, K. H. and G. Hirzinger: *Optimal Hand-Eye Calibration.* In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4647–4653, 2006. 33

[Sha48] Shannon, C. E.: *A Mathematical Theory of Communication.* 27:623–656, 1948. 39, 41

[She13] Sheffield Machine Learning Group: *Sheffield Machine Learning Software.* `https://github.com/SheffieldML`, 2013. [Online; accessed 31-08-2017]. 128

[Sho85] Shoemake, K.: *Animating Rotation with Quaternion Curves.* In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 245–254, 1985. 30

[Sor85] Sorenson, Harold W.: *Kalman Filtering: Theory and Application.* IEEE Computer Society Press, 1985. 100

[SQLG15] Su, H., C. R. Qi, Y.-Y. Li and L. Guibas: *Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views.* In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2686–2694, 2015. 62, 63

[SRW08] Schubert, R., E. Richter and G. Wanielik: *Comparison and Evaluation of Advance Motion Models for Vehicle Tracking.* In *Proceedings of the 11th International Conference on Information Fusion (FUSION)*, pages 1–6, 2008. 99

[Ste94] Steinmetz, R.: *Data Compression in Multimedia Computing – Principles and Techniques.* Multimedia Systems, 1(4):166–172, 1994. 39

[Sut68] Sutherland, I. E.: *A Head-Mounted Three Dimensional Display.* In *Proceedings of the Fall Joint Computer Conference (AFIPS)*, volume I, pages 757–764, 1968. 2

[Ter16] Terminal Eleven: *SkyView.* `http://www.terminaleleven.com/skyview`, 2016. [Online; accessed 31-08-2017]. 2

[TFO+94] Todoriki, T., J. Fukano, S. Okabayashi, M. Sakata and H. Tsuda: *Application of Head-up Displays for In-Vehicle Navigation/Route Guidance.* In *Proceedings of Vehicle Navigation and Information Systems Conference (VNIS)*, pages 479–484, 1994. 4

[TKBO11] Takagi, K., H. Kawanaka, Md. S. Bhuiyan and K. Oguri: *Estimation of a Three-dimensional Gaze Point and the Gaze Target from the Road Images.* In *Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 526–531, 2011. 5

[TM15] Tulsiani, S. and J. Malik: *Viewpoints and Keypoints.* In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1510–1519, 2015. 62, 63

[TMH+12] Tasaki, T., A. Moriya, A. Hotta, T. Sasaki and H. Okumura: *Depth Perception Control by Hiding Displayed Images Based on Car Vibration for Monocular Head-up Display.* In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 323–324, 2012. 5

[Tuf97] Tufano, D. R.: *Automotive HUDs: The Overlooked Safety Issues.* Human Factors, 39(2):303–311, 1997. 5

[UIS+09] Uranishi, Y., A. Ihara, H. Sasaki, Y. Manabe and K. Chihara: *Real-Time Representation of Inter-Reflection for Cubic Marker.* In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 271–218, 2009. 5, 16

[U.S17] U.S. Air Force: *Official U.S. Government Information about the Global Positioning System (GPS) and Related Topics.* `http://www.gps.gov`, 2017. [Online; accessed 31-08-2017]. 14, 17

[UvdSGS13] Uijlings, J. R. R., K. E. A. van de Sande, T. Gevers and A. W. M. Smeulders: *Selective Search for Object Recognition.* International Journal of Computer Vision, 104(2):154–171, 2013. 63, 84

[Ven10] Venerable Master Hsing Yun: *For All Living Beings: A Guide to Buddhist Practice.* Buddha's Light Publishing, Hacienda Heights, California, 2010. 59

[VSH07] Verlinden, J., C. Suurmeijer and I. Horvath: *Which Prototype to Augment? A Retrospective Case Study on Industrial and User Interface Design.* In *ICProceedings of IEEE Virtual Reality Conference (VR)*, pages 574–583, 2007. 2

[Wat03] Watson, B.: *Zhuangzi: Basic Writings.* Columbia University Press, New York, 2003. 1

[Way16] Waymo LLC: *Google Self-Driving Car Project.* `https://waymo.com`, 2016. [Online; accessed 31-08-2017]. 5

[WM97] Wu, X. and N. Memon: *Context-based, Adaptive, Lossless Image Coding.* IEEE Transactions on Communications, 45(4):437–444, 1997. 39

[WON16] Wang, X., S. K. Ong and A. Y. C. Nee: *A Comprehensive Survey of Augmented Reality Assembly Research.* Advances in Manufacturing, 4(1):1–22, 2016. 2

[Wor89] World Wide Web Consortium: *Graphics Interchange Format Specification, Version 89a*, 1989. 39

[WSK+15] Wu, Z.-R., S.-R. Song, A. Khosla, F. Yu, L.-G. Zhang, X.-O. Tang and J.-X. Xiao: *3D ShapeNets: A Deep Representation for Volumetric Shapes.* In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 62

[WSS96] Weinberger, M. J., G. Seroussi and G. Sapiro: *LOCO-I: A Low Complexity, Context-based, Lossless Image Compression Algorithm.* In *Proceedings of Data Compression Conference (DCC)*, pages 140–149, 1996. 36, 39, 50

[WSS00] Weinberger, M. J., G. Seroussi and G. Sapiro: *The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS.* IEEE Transactions on Image Processing, 9(8):1309–1324, 2000. 36, 39, 50

[X D15] X Development LLC: *Google Glass.* `http://www.google.com/glass`, 2015. [Online; accessed 31-08-2017]. 2

[XCLS17] Xiang, Y., W.-G. Choi, Y.-Q. Lin and S. Savarese: *Subcategory-Aware Convolutional Neural Networks for Object Proposals and Detection.* In *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 924–933, 2017. 64, 83, 85

[XLF05] Xu, F.-L., X. Liu and K. Fujimura: *Pedestrian Detection and Tracking with Night Vision.* IEEE Transactions on Intelligent Transportation Systems, 6(1):63–71, 2005. 15

[XMS14] Xiang, Y., R. Mottaghi and S. Savarese: *Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild.* In *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 75–82, 2014. 74

[YJS06] Yilmaz, A., O. Javed and M. Shah: *Object Tracking: A Survey.* ACM Computing Surveys, 38(4):Article 13, 2006. 15

[YKN06] Yamaguchi, K., T. Kato and Y. Ninomiya: *Vehicle Ego-Motion Estimation and Moving Object Detection Using a Monocular Camera.* In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 610–613, 2006. 5, 15

[ZCW+03] Zhang, K.-Q., S.-C. Chen, D. Whitman, M.-L. Shyu, J.-H. Yan and C.-C. Zhang: *A Progressive Morphological Filter for Removing Nonground Measurements from Airborne LIDAR Data.* IEEE Transactions on Geoscience and Remote Sensing, 41(4):872–882, 2003. 78

[ZDB08] Zhou, Feng, Henry B.-L. Duh and Mark Billinghurst: *Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR.* In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 193–202, 2008. 16

[Zee10] Zeeb, E.: *Daimler's New Full-Scale, High-dynamic Driving Simulator – A Technical Overview.* In *Proceedings of Driving Simulation Conference (DSC)*, pages 157–165, 2010. 104, 105

[ZJRP+15] Zheng, S., S. Jayasumana, B. Romera-Paredes, V. Vineet, Z.-Z. Su, D.-L. Du, C. Huang and P. H. S. Torr: *Conditional Random Fields as Recurrent Neural Networks.* In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, 2015. 62, 63, 72

[ZL77] Ziv, J. and A. Lempel: *A Universal Algorithm for Sequential Data Compression.* IEEE Transactions on Information Theory, 23(3):337–343, 1977. 39

[ZLR12] Ziraknejad, N., P. D. Lawrence and D. P. Romilly: *The Effect of Time-of-Flight Camera Integration Time on Vehicle Driver Head Pose Tracking Accuracy.* In *Proceedings of IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 247–254, 2012. 5

[ZS11] Zimmermann, W. and R. Schmidgall: *Bussysteme in der Fahrzeugtechnik – Protokolle und Standards [Bus Systems in Vehicle Technology – Protocols and Standards].* Vieweg, 2011. 8

[ZSSS11] Zia, M. Z., M. Stark, B. Schiele and K. Schindler: *Revisiting 3D Geometric Models for Accurate Object Shape and Pose.* In *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 569–576, 2011. 61