Occlusion-aware Perception and Planning for Automated Vehicles

by

Yuanxin Zhong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2023

Doctoral Committee:

Professor Henry Liu, Co-Chair
Professor Huei Peng, Co-Chair
Professor Bogdan Epureanu
Assistant Professor Maani Ghaffari Jadidi
Assistant Professor Xun Huan

Yuanxin Zhong
zyxin@umich.edu
ORCID iD: 0000-0001-6970-4135

DEDICATION

In Loving Memory of Professor Huei Peng,

I dedicate this dissertation to the cherished memory of my dear advisor, Professor Peng. His unwavering support and immense wisdom have shaped not only this research but also guided my academic journey. Despite the occasional differences of opinion we had, I remain immensely grateful for his profound guidance and keen insights. His mentorship, always offered with grace and understanding, has been pivotal in refining my research skills and fostering both my personal and professional growth.

More than a mentor, Professor Peng was an illuminating source of inspiration, a wellspring of knowledge, and a role model. He taught me the importance of being not just a dedicated student but a genuinely good person. He emphasized the value of asking 'why' before investigating 'how', and the significance of addressing overarching issues without overlooking the intricate details. His generosity in sharing knowledge and making the latest advancements accessible to people worldwide mirrors an old Chinese proverb: "春蚕到死丝方尽，蜡炬成灰泪始干".

Aside from his professional demeanor, Professor Peng was an affable individual who loved to bring us together for heartwarming gatherings. His personal warmth and congeniality were always on full display, whether we were doing experiments or playing a game of tennis. I will always cherish these memories, and his warm smile will continue to inspire me.

Having been one of his last students is a privilege that I deeply appreciate. I am determined to pursue his ambition of making vehicles safer and more efficient.

May his soul rest in eternal peace.

Yuanxin

ACKNOWLEDGMENTS

Extending my profound gratitude to my advisor, Professor Huei Peng. I am equally and deeply grateful to Professor Henry Liu, who kindly welcomed me as a student and provided critical assistance during pivotal moments. Beyond this, Professor Liu's helpfulness, support in addressing academic challenges, and innovative ideas in the field of intelligent transportation systems have been profoundly inspiring. His guidance and constant encouragement have significantly shaped my research trajectory and spurred me toward excellence.

Next, my sincere appreciation goes to my committee members: Professors Bogdan Epureanu, Maani Ghaffari Jadidi, and Xun Huan. Their astute guidance, insightful feedback, and extensive expertise throughout the dissertation process have been invaluable. Their thoughtful inputs and constructive criticism have considerably elevated the quality and rigor of my research.

I am eternally thankful to my parents for their unwavering love, constant support, and unshakeable faith in my abilities. Their continuous encouragement across the Pacific has fueled my achievements. Concurrently, I must express my heartfelt appreciation to my girlfriend Yuguo for her enduring support, understanding, and patience throughout this demanding journey. Her steady presence has been a source of infinite strength and inspiration and I can not row across the river of depression without her smile.

To my cherished friends who have become an integral part of my life and growth in Ann Arbor, including Fucong, Songan, and my comrades Xingmin, Xintao, Siyuan, and Lu who journeyed to Michigan with me, I offer my deepest gratitude. The springs and falls in Ann Arbor would not have been so enchanting without your presence.

My colleagues at the Vehicle Dynamics Lab and the Michigan Traffic Lab deserve special mention for their camaraderie, intellectual discussions, and shared experiences that have been instrumental in overcoming challenges and broadening the horizons of knowledge. My heartfelt gratitude extends to all of you: Shaobing, Pingping, Yiqun, Steven, Su-Yang, Geunseob, Songan, Nauman, Juhui, Zhong, Julia, Boqi, Minghan, Han, Tinghan; Sean, Yan, Haowei, Haojie, Rusheng, Ronan, Depu, Zachary, Zihao and Junkai.

Lastly, I want to thank Mcity and Toyota Research Institute for their financial support. Your funding has made a significant impact on the progress and success of my research.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF PROGRAMS

LIST OF ACRONYMS

ADAS Advanced Driver Assistance Systems,

ADS Automated Driving System,

AV Automated Vehicles,

BEV Bird's Eye View,

CLAP Cloud-and-Learning-compatible Autonomous driving Platform,

CNN Convolutional Neural Networks,

CRF Conditional Random Field,

CV Computer Vision,

DARPA Defense Advanced Research Projects Agency,

DSRC Dedicated Short-Range Communications,

FoV Field of View,

fwIoU frequency-weighted Intersection-over-Union,

GNSS Global Navigation Satellite Systems,

GPOM Gaussian Process Occupancy Map,

HD High-Definition,

HOG Histogram of Oriented Gradients,

LQR Linear-Quadratic Regulator,

ICS InConsistency Suppression,

IDM Intelligent Driver Model,

IMU Inertial Measurement Unit,

IoU Intersection-over-Union,

mAP mean Average Precision,

mIoU mean Intersection-over-Union,

MLP Multi-Layer Perceptron,

MPC Motion Predictive Control,

MRF Markov Random Field,

NDS nuScenes Detection Score,

NMS Non-Maximum Suppression,

ORU Other Road User,

PCL Point Cloud Library,

PID Proportional–Integral–Derivative,

POMDP Partially Observable Markov Decision Process,

PQ Panoptic Quality,

RCNN Region-based Convolutional Neural Network,

RL Reinforcement Learning,

RMSE Root Mean Squared Error,

ROS Robot Operation System,

RoW Right of Way,

RQ Recognition Quality,

SAE Society of Automobile Engineers,

SDF Signed Distance Function,

SLAM simultaneous Localization and Mapping,

SME Signed Mean Error,

SQ Segmentation Quality,

SVM Support Vector Machine,

TSDF Truncated Signed Distance Function,

TTC Time to Collision,

V2X Vehicle to Everything,

VIN Voxel-based Implicit Network,

ABSTRACT

The perception system is a key component of AVs, as it relies on onboard sensors to gather information. However, the system faces challenges due to occlusions that obstruct visibility. Safely navigating in highly occluded scenarios remains a significant obstacle for AVs. In this dissertation, we present a systematic approach to address this issue by preparing AVs for fully occluded areas on the road.

To effectively model the occluded areas, we introduce a joint object detection and semantic segmentation algorithm. This helps in acquiring critical environmental information with increased efficiency, enabling AVs to make decisions in the presence of occlusions. In tandem, we propose a semantic 3D mapping framework that efficiently identifies occlusions, which feeds into a layered 2D map, essential for planning, and contains the occlusion data. The experiments in SemanticKITTI dataset demonstrated that the proposed perception algorithms can generate a semantic grid map of the environment and identify the occluded grids efficiently and effectively.

To tackle the occlusion problem and produce a safe plan for the AV, an occlusion-aware object management system is introduced to generate virtual road users for the planning algorithm, and near-optimal trajectories are solved using a sampling-based method while taking the presence of virtual objects into account. The experiments in a 2D toy driving environment showed the proposed planner can achieve better safety against baseline approaches while maintaining a reasonable passing speed in several challenging testing scenarios.

Furthermore, the effectiveness of the proposed perception and planning framework is validated in both the Carla simulator and the physical Mcity testing facility, demonstrating the effectiveness of the proposed architecture and its superior safety performance over baseline approaches. Besides, a modular AV stack is described to guide the integration of the proposed perception and planning framework in the experiments.

Validation experiment results show that the proposed framework results in a reduced crash rate in comparison to several baselines, including the renowned open-source AV framework, Autoware. Notably, these outcomes were realized without needing a High-Definition (HD) map with road geometry definitions.

CHAPTER 1

Introduction

The Automated Driving System (ADS) is an emerging multidisciplinary topic both for the academic and industry. It's expected to improve the driving safety, passenger experience, traffic throughput and logistic efficiency. The boom of its public interest starts from the famous prize competitions "Grand Challenge", and later the "Urban Challenge" held from 2004 to 2007, sponsored by Defense Advanced Research Projects Agency (DARPA). Many famous companies building ADS, such as Waymo and Uber, originated from the teams participated in those challenges [170, 242]. There are also AV tests organized by research teams to demonstrate the possibilities of fully autonomous driving platforms including the VIAC[17] and PROUD[27] tests. Nevertheless, these tests did not indicate that ADS has reached maturity.

In 2014, another milestone for the development of ADS is the publish of the J3016 document [51] from the Society of Automobile Engineers (SAE), which defines six levels of capabilities for the AV. It has been one of the most cited references for driving automation. A mature ADS must be able to comprehend and monitor the driving environment, and react to road events properly and safely. The J3016 standard was published and regularly updated to help the industry evolve and provide a roadmap for companies to reach the final full driving automation. There are also other government documents [79, 1] for AV published around that time [1].

Guided by those legislative documents, there are a growing number of AV start-ups building a variety of ADS. Different companies take different approaches to achieve the driving automation. Some of them (namely Tesla and Xpeng) build the ADS with only mature and cheap automotive sensors (such as cameras and radars), while most others are trying to deploy more advanced sensors like lidars and mmWave radars. Some of them rely on the onboard sensors to build a model of the driving environment, while others incorporate pre-built HD maps or Vehicle to Everything (V2X) communication to help the vehicles achieve a higher automation ability. Yet, regardless of the different approaches, all AVs must take safe reactions even in very challenging scenarios. One of those scenarios is driving in a highly occluded environment, which is a big challenge even for

[1] A collection of laws and regulations related to AV can be found at this webpage from Stanford.

the latest competent ADS. According to the report [208] from Waymo, 2 out of 8 critical events that likely involve airbag deployment were directly related to occlusion, while the other events are caused by the incompetence or adversarial behavior of the other vehicle. Recently, Cruise [117] and Tesla [70, 71] also publicly explained their perception and planning system, and some of their efforts have been put into the occlusion issue as well.

Therefore, in this dissertation, we worked on improving the safety and roadmanship of ADS under severe occlusion, by proposing novel perception and planning algorithms with occlusion awareness. In this chapter, we will first review the architecture of AVs, and then briefly discuss the existing algorithms for AV perception, cognition, planning and control. Then, the research problem will be proposed. Lastly, the contributions and structure of this dissertation will be summarized.

1.1 System architectures of AV

Like most robotic systems, AV is required to understand the surrounding environment and independently make decisions to accomplish some given tasks. The winner of the DARPA Urban Challenge, the "Boss" from Carnegie Mellon University, was documented in [242]. Its on-board system consists of modules of perception, motion planning, mission planning, behavior reasoning. Following these successful architectures, a lot of researchers[119, 320, 13, 264, 10] and commercial companies [158, 157, 156] have explained their designs of ADS, which usually split the onboard system into three high-level modules: Sensing, Computing and Planning.

Sensing

The sensing system consists of a variety of automotive-grade sensors. Different from **intrinsic sensors** which monitor the status of the vehicles, **extrinsic sensors** are used to collect raw information of the environment surrounding the vehicle. In the remainder of this dissertation, we only discuss the extrinsic sensor (omitting the adjective "extrinsic"), which are most relevant to the driving automation. Currently, the commonly used sensors includes cameras, radars, lidars, Global Navigation Satellite Systems (GNSS), and Inertial Measurement Unit (IMU). Each sensor is usually accompanied with auxiliary hardware modules for power supply and data decoding. The configuration of the sensors on AVs is subject to the trade-off between cost and performance.

Cameras have been widely used in the automotive industry with the development of Advanced Driver Assistance Systems (ADAS) systems. It can provide images of the environment with a high information density and a relatively lower cost. However, the camera can only sense the objects within its field of view and its availability can degrade significantly due to bad weather, glare, and ill lighting condition. Therefore modern AVs [158, 157] are usually equipped with multiple

cameras so that their field of view can complement each other and the depth information can be implicitly inferred based on the extrinsic transformation between the cameras.

Radars have also gained popularity thanks to the wide deployment of ADAS systems. Radars use radio waves to detect obstacles in certain directions, it report the distance to the obstacles in a variety of range, and some radars can also report the velocity of the obstacles based on the Doppler effect. Radars are robust against weather change and it's widely used in primitive ADAS features such as automatic emergency braking, blind spot monitoring, and parking assistance. Nevertheless, most radars deployed on vehicles nowadays has a low resolution, and it's hard for them to categorize the obstacles. They can hardly tell whether the obstacle is a standing pedestrian or a pole.

Lidars, compared with the two types of sensors above, haven't been widely integrated into the automotive systems until recent years. They can precisely describe the shape of the environment through a collection of laser reflections, which is usually called the point cloud. Thanks to the high precision, lidars have attracted many AV companies including Waymo, Cruise, Baidu, etc. However, despite the desreasing trend of the lidar price, they are still much more expensive that the cameras (per unit) and it's still hard for lidars to achieve the same level of information density as cameras.

GNSS and **IMU** are the two key components for the ADS. The GNSS module receives the information from the navigation satellites and reports the position of the vehicle related to the earth, which is usually considered as the "global" position of the vehicle. On the other hand, the IMU module usually consists of a gyroscope, an accelerometer and a compass, which reports the orientation, acceleration, and angular velocity of the body it attached to. These two components are usually integrated together to obtain an accurate global pose and velocity information of the vehicle.

Besides the common sensors mentioned above, new types of sensors are being developed to meet the growing requirements of efficiency and robustness. 4D imaging radar and event cameras are some of the most promising new sensors. However, they are still experimental and the related software and algorithms for those sensors are not ready yet.

Computing

After the environment information is fetched by the sensors, the computing modules are responsible for analyzing the data and provide instant commands for the actuation module. The common computing modules include Perception, Navigation, Cognition and Planning subsystems [302].

Perception is the subsystem that process the raw sensor data input, and produces primitive abstractions of the data, such as object category, object pose and size, semantic masks and motion

flow. These information is usually produced by object detection and tracking, semantic segmentation and motion estimation algorithms.

Navigation is the subsystem that is responsible for telling where is the ego vehicle, and provide information beyond the perception range of the on-board sensors. Mapping and localization are the two parts of this system and they are usually coupled. Mapping module provides the abstraction of the static environment including terrain profile, road boundary, lane markers, etc. It's usually accomplished by a mapping algorithm based on Computer Vision (CV) algorithms or a map service provider. Localization then utilizes GNSS/IMU sensor readings, and/or the mapping results to tell the pose and velocity of the ego vehicle. Based on these information, the global route for the a trip can be generated with some routing algorithms.

Cognition[302] is the subsystem that provide higher-level abstraction of all the results from perception and navigation. In this system, the tracked targets will be associated with the static map, and a prediction module will provides estimations of the surrounding targets. Then all the related information will be organized and provided to the downstream subsystems.

Planning is the subsystem that consumes the abstracted information from upstream modules and generate safe and effective trajectories for the actuation module to follow. There are two common steps in this subsystem. The first one is behavior planning, which generates high-level intentions such as lane changing and merge-in maneuvers. The second one is motion planning, which compute an optimal or near-optimal path and then a trajectory, considering certain goals and constraints.

Actuation

The actuation module will control the ego vehicle to follow the computed trajectory. A controller will first convert the trajectory into primitive commands (steering, throttle, braking) based on the current states of the ego vehicle. Then those commands will be sent to the internal network of the vehicle, and they are usually executed by drive-by-wire hardware modules, including steer-by-wire, brake-by-wire and electronic throttle control.

Auxiliary Modules

Besides these three high level modules responsible for a competent driving behavior of the AV, there are also some auxiliary modules that provide supports for other aspects of the AV. For example, there is a safe guard in many ADS [156, 231], which monitor and assess the performance of the vehicle and provide a fall-back controller in case of system failure. Besides, remote assistance client is also an important module for robotaxi companies, which allow them to take-over the vehicle without physical contact.

Among these auxiliary modules, a very important component for ADS is the middleware. ADS is usually designed to be modular, so that the software modules are loosely coupled and the system complexity can be reduced. In a modular system, the middleware is responsible for transmitting data between the modules and provide diagnostic information when necessary. The Robot Operation System (ROS) [198] is a widely used middleware which provide a data communication framework and a rich set of utility libraries. Based on ROS, [156] developed a new middleware called Cyber-RT, which claims to have high throughput and low latency. The middleware is the foundation of many ADS frameworks [156, 119, 120, 302].

1.2 Perception and Cognition Algorithms for AV

The perception algorithms in AV are responsible for extracting the objects and model the static environment based on the data input from the sensors. For objects, the common tasks include object detection and object tracking. For the static information, the common tasks include semantic segmentation and mapping. The results produced by perception modules (and navigation modules) will be consumed by the cognition algorithms to generate a driving map for the downstream planning module [302]. Common modules of cognition include prediction and mapping.

In this section, the algorithms for AV perception and cognition will be reviewed.

Object Detection

Object detection algorithms identify the objects of interest in a sensor frame, and report basic information about the objects, usually including position and classification. The object detection algorithm usually consists of two steps: hypothesis generation and hypothesis refinement [47]. In the generation step, potential objects are identified in the sensor frame; In the verification step, these objects are verified so that false positives will be rejected. Furthermore, many algorithms also refine the location of the objects in this step.

In the early days, primitive features (e.g. colors, edges, and symmetry) are used for the 2d object detection [31, 135, 232]. These features are prone to environmental artifacts such as shades, raindrops, sensor occlusions, etc. However, these features can be cheaply computed and they are usually used in the hypothesis generation step. For the verification step, the researchers turned to robust artificial features such as Histogram of Oriented Gradients (HOG)[47] and Haar[97, 246]. Some methods also utilize the classification methods such as Support Vector Machine (SVM)[232, 222] and AdaBoost[153, 125] for the verification step. Besides the appearance-based object detection algorithm that uses the features mentioned above, there is another way to do object detection by finding the changes in consecutive sensor frames. This ap-

proach is usually called "detection by tracking", and it will be discussed in the review of tracking algorithms.

After the Convolutional Neural Networks (CNN) became popular in computer vision tasks, deep learning has been widely applied to object detection algorithms, starting from the success of the series work of Region-based Convolutional Neural Network (RCNN)[85, 84, 204]. The structure of the CNN based object detectors also follows the generation-verification structure, but it can achieve much higher precision and recall compared with the classical methods using artificial features, thanks to the robustness of the features extracted from CNN. Furthermore, the flexibility of the deep learning algorithms makes it possible to combine the hypothesis generation and verification into a single framework[204, 201], and the performance boost benefit from the end-to-end optimization of the neural networks.

Detection in 2D sensor frames (e.g. images) is a mainstream task discussed in the literature. However, detection in 3D sensor frames (e.g. point clouds) is also an emerging task arising from the widespread adoption of lidar sensors. The difference between detection in 2D and 3D sensor frames is that the feature extraction algorithms for the two types of data are different. In classical methods, the point cloud is usually split into voxels. The common features of point clouds used in the detection are the point density, relative location to the voxel center, normal direction and, reflectance [250, 223]. The detection hypothesis is usually generated with a sliding window. With the development of deep learning, some researchers also explored the application of neural networks in the 3D space. The two notable works for feature extraction from the point cloud are PointNet[196, 195] and VoxelNet[311]. The former created a paradigm for feature extraction without splitting the points into voxels. Later, there is a growing number of works that applies neural network techniques to the 3D object detection problem [282, 215, 306, 291, 307].

Object Tracking

Object tracking algorithms track the existence, identification and dynamic states of objects. It can also improve the state estimation accuracy by combining the information from the multiple frames, compared with the states estimated from a single frame using object detection algorithms. The object tracking task defined in the computer vision consists of two variants: single object tracking and multiple object tracking. The latter one is more relevant in the context of AV and we only discuss the multiple object tracking algorithm in the remainder of the thesis.

There are two paradigms for object tracking: "detection by tracking" and "tracking by detection". The first approach directly identifies the changing parts in the sensor frames. Optical flow [80, 163] or and occupancy grids [58, 88] are widely used in this approach. However, a separate module is required for extracting the objects from the optical flow or occupancy grid, and it's hard

for this approach to find static objects. Therefore, most of the tracking methods follow the second approach, which first detects the objects per frame, and then track the objects based on the detection results. In this approach, there are two steps: data association and state update.

The data association step find the correspondence between the detected objects and the previously tracked object. It is expected to re-identify the same object between multiple frames. At the second step, a state filter will update the state of the object given the current state, uncertainty and the new observation. The Hungarian algorithm [136] is widely used for the association step [269, 19, 306], which globally optimize the distance between matched targets. Other association methods include nearest neighbor [59], multi-hypothesis tracking [21], and probabilistic data association [73]. On the other hard, the Kalman filter and its variants are widely used for the state update [269, 19, 306]. With the development of deep learning, researchers has proposed to use neural networks to achieve the data association steps as well [269, 265].

Semantic Segmentation

Semantic segmentation algorithms provide dense semantic annotations for the sensor data. The annotation is usually pixel-wise for image inputs [300, 128], and point-wise for point cloud input [196, 233, 57, 48]. Semantic information is important for several downstream tasks including lane detection, drivable area detection, and traffic sign detection. These downstream tasks extract the data of interest based on the semantic annotations and use some post-processing methods to model the target. For instance, lane detection algorithms [160, 217] usually first extract the lane marker pixels, and then use some geometric models to estimate the lane parameters.

Before the wide applications of neural networks, the semantic segmentation is usually accomplished with classical machine learning models including decision forests [218, 29] and conditional random fields [224, 138]. With the development of deep neural networks, CNN has been widely used in semantic segmentation as well [300, 128, 129, 257, 305, 57]. The neural network can be directly supervised by pixel-wise or point-wise semantic losses. Researchers have also tried to utilize the temporal information from multiple sensor frames in the segmentation tasks by fusing the features from each frame [214] or selected frames [214, 180]. A more detailed review of the semantic segmentation (and object detection) can be found in Section 2.1.1.

Prediction

With the detection and tracking of the targets on the road, an important step for the ADS is to predict the future trajectories of these targets. These predicted trajectories will be provided to the planning module for risk assessments.

The prediction algorithms can be categorized into physics-based, pattern-based, and planning-based [118]. The physics-based planning methods include state estimation (e.g. constant velocity [30], Kalman filter), and reachable sets. They predict future trajectories directly based on the current state and explicit motion models. However, these methods don't predict the input to the motion models, and they are usually accurate only in a short period of time. The pattern-based methods find the most likely motion pattern (aka. mode) and the prediction is rolled out based on the pattern. Many kinds of surrounding information will be taken into consideration, such as road type, drivable areas, and traffic signs. The output of these methods is either the full trajectories or high-level maneuvers. The motion pattern can be identified using clustering [7], classification [271], or encoder-decoder algorithms. The planning-based prediction algorithms are a class of models that infer the future actions of an agent based on its implicit goals, as evidenced by past actions. They primarily follow two approaches: one determines the most likely goal and corresponding policy (action plan) of the surrounding agent using inverse Reinforcement Learning (RL) [317, 228] or imitation learning [194, 101], and the other incorporates the prediction directly into the motion planning module usually based on the Partially Observable Markov Decision Process (POMDP) [110] or game theoretic frameworks [293]. The former planning-based approach also relies on an adequate amount of data for training like the pattern-based prediction methods, while the latter one relies on motion or interaction models like the physics-based prediction methods. Please refer to Section 1.3 for a more detailed review of planning algorithms.

The neural networks for prediction usually follow the encoder-decoder structure, because it is a common strategy in deep learning algorithms to force the networks to extract high-level information. The algorithms using deep learning can be categorized as discrete ones or distributional ones based on the format of the data they operate on. They can also be categorized as spatial, temporal, spatial-temporal, or relational ones based on how the data inputs are organized. The discrete methods usually operate on discrete perception outputs such as grid maps [187], rasterized images [56] and BEV renderings[62]. The distributional approaches usually rely on the variational autoencoder structure to predict the future trajectories [161, 141]. Prediction with relational modeling has become popular in recent years and a notable method for relational modeling is the graph neural network, which has been widely applied to trajectory prediction [90].

The behavior of different targets (e.g. vehicles, pedestrians, etc.) can be considerably different. The pedestrians can have higher social interaction and their trajectories are more volatile, while the vehicles are limited by the vehicle dynamics and they can change their states, such as velocity and heading, very quickly. Therefore, the physics-based and pattern-based planning methods must have separate dynamics models for these targets. However, for planning-based algorithms, the algorithms are usually designed such that they can cover a variety kinds of targets [93, 146].

Mapping

The mapping process is an important module for both navigation and cognition. It carries all the related information from the environment in a data structure (i.e. the map) and the planning system uses it for collision avoidance. Based on the extent of the map, there are two kinds of maps: global navigation maps and local planning maps. The former ones are designed to cover the globe or part of the globe, and they usually provide the map information with precision up to road level. Famous providers of global maps include Google, Here, Baidu, Tencent, etc. In terms of the application for AV, these maps are usually used for routing [302], and they are stored on the cloud, rather than on each vehicle. In ADS, the local planning map is more relevant because it only describes the information within a certain range of the ego vehicle and usually contains up-to-date information, which is crucial for the planning system. The precision of the local map is usually higher than that of the global map as well. However, local maps are usually built on top of global maps, because most parts of the static environment do not change over time and are suitable for storage in the cloud and retrieved when needed. A more fine-grained categorization of maps is described in [283], which proposed a 7-layer HD map model. In this model, layers 1 to 4 store the global traffic information, layers 5 to 7 store the dynamic local information from the perception of AVs.

A local planning map might include several aspects [270]: topology, geometry, semantics, and road users. Different data models are required for different aspects. For example, [119] uses point clouds to represent the geometry and Lanelet2[193] to represent the road topology. On the other hand, [302] uses a unified data structure to store the topology, geometry, and road users. Currently, there lacks a consensus on the map representation and storage format for AV. A brief overview of different (HD) map formats can be found at [237].

The algorithms for the different aspects are also different. The topology information currently relies on data collection and offline pipelines, which usually rely on manual annotations. The semantic information is usually generated from semantic segmentation as mentioned above, while the attributes and dynamic states of the road users can be obtained by object detection and tracking. Lastly, the process of modeling the environment geometry is commonly referred to as the mapping process. There are two major approaches for the mapping problem, simultaneous Localization and Mapping (SLAM)[11, 144, 174] and occupancy mapping[185]. SLAM algorithms focus more on the localization and it usually produces a sparse map, while the occupancy mapping focus on the mapping coverage and it is more compatible with the planning algorithms. A more detailed review of the occupancy mapping algorithms can be found in Section 2.2.1.

The local map also provides an important context for the planning algorithms and many prediction algorithms incorporate HD maps to help regularize the trajectories. The map context is usually used by algorithms with the encoder-decoder structure as the map information can be fed into an encoder as well [167, 78].

Datasets

Different from other modules in ADS, the perception and cognition algorithms heavily rely on datasets because they usually operate on sensor data that are hard to be simulated accurately, and machine learning techniques are widely used in those algorithms. Recently, there have been more and more datasets collected by both academic organizations and commercial companies. Table 1.1 summarizes some famous autonomous driving datasets. It's worth noting that some datasets have special properties. For example, the CADC dataset[191] contains a majority of data collected in snow weather, the VKITTI[74] and Woodscape[292] datasets contain synthetic data, and the Mcity dataset [65] contains choreography data collected in a closed testing facility.

Table 1.1: A list of popular AV datasets[2]

Dataset	Release Date	Label	Bounding Box	Semantic Label	Vehicle Pose	HD Map	Camera
KITTI[81]	2012	15k frames	✓	✓ (2D)	✓		✓ (stereo)
RobotCar[162]	2015	-			✓		✓ (stereo + 3 fisheye)
KAIST[116]	2017	-			✓		✓ (stereo)
VKITTI[74]	2017	15k frames		✓			✓ (stereo)
ApolloScape[108]	2018	5k frames	✓	✓ (2D)			(separate)
MVSEC[313]	2018	-			✓		✓ (event)
SemanticKITTI[14]	2019	15k frames		✓			
nuScenes[32]	2019	40k frames	✓	✓	✓	✓	✓ (6 mono)
Lyft L5[105]	2019	55k frames	✓		✓	✓	✓ (6 mono)
Argoverse[40]	2019	20k frames	✓		✓	✓	✓ (stereo)
H3D[188]	2019	1M boxes	✓		✓		
A*3D[190]	2019	39k frames	✓				
WoodScape[292]	2019	10k frames	✓	✓ (2D)	✓		✓ (4 fisheye)
BLVD[281]	2019	250k boxes	✓	✓			✓ (mono)
Waymo[229]	2020	20k frames	✓		✓		✓ (5 mono)
CADC[191]	2020	7k frames	✓		✓		✓ (stereo + 8 mono)
Cirrus[262]	2020	6k frames	✓		✓		✓ (mono)
KITTI360[150]	2020	80k frames	✓	✓	✓		✓ (stereo + 2 fisheye)

1.3 Planning and Control Algorithms for AV

With the environment model produced by perception and cognition modules, the ADS is then required to make decisions and generate proper commands to let the AV drive to the desired position, which are usually accomplished with the planning and control modules. The planning process is usually split into a high-level behavior planning (aka. decision making) step and a low-level motion/trajectory planning step. After the planning steps, the vehicle controller will generate commands for the actuators based on the current vehicle states.

[2]None exhaustive. Please refer to the AD-Datasets website [22] for a more complete list.

In this section, the algorithms for AV planning and control will be reviewed.

Behavior Planning

The behavior planning module is responsible for determining high-level intentions such as lane change, highway merge-in and yield on unsignalized intersections. In the early days, the behavior planning is usually built on simple heuristics or finite state machines. The highway scenarios are particularly interesting for the researchers, because the challenge for AV driving on highways are mostly related to making proper decisions. The related maneuvers include lane following, car following, lane changing, passing, overtaking, merging and highway toll [49]. The lane change behavior was specifically well-studied by transportation engineers for the purpose of microscopic simulation. Gipps [83] proposed the first heuristic decision model for lane changing, and it takes various traffic properties into consideration with an accompanying longitudinal behavior model such as the Intelligent Driver Model (IDM)[238]. Later in 2003, [236] proposed a comprehensive gap acceptance model for the lane changing decision. Based on these work, [123] proposed a lane change model with a parameter to change the politeness of the driving behavior.

Aside from the behavior models designed for specific scenarios, some algorithms are applicable to arbitrary scenarios with a preset of candidate actions. The fuzzy logic [12, 54] method is a systematic way to composite the rules, it can emulate the human driver's behavior by defining linguistic variables. However, it's not scalable to complex scenarios. Given the states of the road users, some researchers built behavior models using machine learning classifiers, such as support vector machine [263], nearest neighbor [67]. Among these methods, the decision process is usually formulated as a Markov process, where the related techniques including Markov decision process [310, 54, 82, 316], hidden Markov model [143], and RL[102, 253, 169] are commonly applied. Another formulation of the decision process is to view the interactions between road users as a game, and the game theoretic approaches [148, 235] can be leveraged to improve the interactive-awareness of the behavior model.

With the growing amount of available driving data and the growing demand for robustness in corner cases, the deep learning [107] approaches has become more and more popular in the behavior planning problems. Deep learning is also the foundation of many RL algorithm, which either estimate the reward using neural networks (aka. Q-learning) [145, 102], or use neural networks in the actor-critic framework [288]. Some researchers classified the methods mentioned above by being cognitive / rational or learning-based / rule-based [49], which provides a great overview of the characteristics of the methods.

Trajectory Planning

With a given target behavior, the trajectory planner will generate a feasible trajectory for the ego vehicle to follow. In some papers, the trajectory planning happens after a motion / path planning step that generates a feasible path with collision check, and then assign the velocities to the waypoints of the path.

The path planning algorithms have been extensively studied by the robotics researchers, which can be categorized by five characteristics [50]: space configuration, pathfinding, symbolized forces, parametric curves, numeric optimization and machine learning. The first category of path planning algorithms relies on a specific representation of the feasible space, such as cell grids, road graphs, and state lattice. These algorithms include the Voronoi diagram [142] and probabilistic roadmap [121]. The second category is based on pathfinding algorithm on graphs, such as A* [96, 24] and rapidly-exploring random tree [140, 61, 64]. The third category symbolize the configuration space as attractive or repulsive forces. The famous family of algorithms based on artificial potential fields [126] fall into this category. The forth category are commonly used for path planning on highways, because the highway roads are usually built as simple curves[243, 248]. The fifth category is usually tightly coupled with vehicle control, and its algorithms formulate the planning problem as an optimization problem with certain goal and cost functions. The popular examples in this category include motion predictive control [152, 3, 38] and quadratic programming [278, 279, 318]. The last category emerges with the development of machine learning and deep neural networks. The machine learning models can be trained to directly output efficient and feasible paths, using RL [178, 256] or imitation learning [202].

Apart from path planning algorithms, there are also algorithms that directly generates trajectories. These algorithms can be divided into two categories [212]: geometrical curve-based and optimization-based. The algorithms in the former category first generates candidate trajectories with a certain family of curves, and then the trajectories will be selected based on some criteria. The curve families that are commonly used in ADS are Bézier curves[87], splines[255], polynomials[319, 267], arcs[225] and clothoids[28]. The latter category mainly refers to the Motion Predictive Control (MPC), which has been widely used in trajectory planning for vehicles [172, 181, 312].

Vehicle Control

After the desired trajectory has been finalized by the trajectory planning module, the vehicle controller generates commands for the actuators such that the ego vehicle can reach the desired state in time. For AVs, the control of the vehicle is usually split into longitudinal control and lateral control [8, 41], and some controllers are designed only for lateral control (i.e. they determine the steering

angle of the vehicle), such as pure pursuit [55, 301], the Stanley controller [234], and preview control [274, 278]. They work together with longitudinal controllers, which determine the throttle and braking of the vehicle, to provide a fully functional controller for the AV. However, since separating the lateral and longitudinal control makes it hard for the lateral controller to be adaptive to speed change, many approaches consider the control problem in a single formulation. The notable examples include methods based on Linear-Quadratic Regulator (LQR) [298] and MPC [149, 92]. A comparison of different controllers can be found at [287].

With the development of machine learning algorithms, researchers have also been trying to integrate machine learning models into the vehicle control problem. These methods are usually trained in an end-to-end manner in order to leverage the flexibility of machine learning. Among the methods, RL [247, 122, 289] and supervised learning [194, 42, 23] are the two most popular directions. However, the end-to-end controller usually lacks interpretability and stability guarantee, they are not of the greatest research interest.

Simulators

Different from the algorithms for perception and cognition which relies on large-scale datasets, the planning and control algorithms are usually trained and validated on simulators. The fidelity and scalability of the simulators for AV make a difference to the development cycle of AV planning and control algorithms. Some of the popular simulators for AV are listed in Table 1.2, and more information can be found at [237].

Table 1.2: A list of popular simulators related to AV.

Simulator	Specialization	Functionality					
		Multi Ego-Vehicle	Environment Traffic	Sensor Configuration	High-Fidelity Rendering	Vehicle Dynamics	Open Source
Carla[66]	Graphics	✓	✓	✓	✓	✓	✓
LGSVL[205]	Graphics		✓	✓	✓	✓	✓
AirSim[210]	Graphics			✓	✓	✓	✓
DeepDrive[199]	Graphics			✓	✓		✓
DRIVE Sim[182]	Graphics		✓	✓	✓		
Gazebo[130]	Robotics	✓		✓		✓	✓
MORSE[69]	Robotics	✓		✓			✓
CoppeliaSim[52]	Robotics	✓		✓		✓	
TORCS[275]	Racing	✓				✓	✓
CarSim[220]	Dynamics	✓	✓				
CarMaker[9]	Integration	✓	✓	✓	✓	✓	
PreScan[219]	Integration	✓	✓	✓	✓		
VI-grade[245]	Integration	✓	✓	✓	✓	✓	
CommonRoad[4]	Traffic		✓				✓
Vissim[89]	Traffic	✓	✓				
SUMO[133, 159]	Traffic	✓	✓				✓

1.4 Problem Statement

With the current design of ADS architectures as described in Section 1.1, the AV can be trained to be responsive to the visible objects and events on the road, but it lacks the ability to handle the objects beyond the sight, which can be caused by sensor degradation [244] or environment occlusions [208]. These two problems are critical for the safety of the AV and have to be addressed before the wide acceptance of AVs. There is little we can do about the former problem yet. It is usually tackled by automatic sensor cleaning devices [226] or a safeguard based on contamination detection [241, 239]. However, for the occlusion problem which is more common in normal driving, we believe it is feasible to solve it by improving the design of the ADS architecture with modifications on computing modules so that the AV can be prepared for the risk caused by occlusion and drive safely yet still efficiently in a crowded environment.

It is worth noting that a lot of the research work [249, 131] for computer vision focuses on improving the perception performance with **partial occlusions**. On the contrary, we focus on **full occlusions** in this dissertation, because conventional perception algorithms for object detection and tracking still work under partial occlusions, but not under full occlusions.

In this dissertation, we aim to address the following questions motivated by the problems mentioned above:

1. How to efficiently represent the occluded area in the environment, without the help of *a priori* maps?

2. How to let the AV drive safely with occlusions in the environment, without negatively affecting mobility?

For the first question, we require the solution to be independent of prebuilt maps (especially HD maps), because the construction of these maps can be expensive, and a method that relies on prebuilt maps cannot handle short-term changes of the environment. For the second question, we require the solution not to negatively affect the mobility of the vehicle significantly, because driving exceptionally slowly is a trivial solution for the vehicle to keep safe. However, experienced human drivers will only slow down if they determine the occlusion is severe and risky enough. This behavior is also known as defensive driving. Nevertheless, they will keep desired speed or recover to the desired speed quickly if they found out that there is no actual risk, by previous observations or reasoning. We expect a competent AV to behave similarly, by being cautious but not overly conservative under full occlusions.

1.5 Contributions

To answer the question proposed above, several contributions have been made in this dissertation. The remainder of this dissertation is organized into individual chapters, and each chapter describes the work on a specific problem. The contributions are summarized below:

1. Novel perception algorithms are proposed to efficiently represent the occluded area in a driving environment (Chapter 2):
   (a) An algorithm for joint 3D object detection and semantic segmentation is proposed.
   (b) An algorithm for fast probabilistic 3D occupancy mapping is proposed.
   (c) A framework for occlusion modeling with occupancy maps is proposed.
2. Novel planning algorithms are proposed to enable safe driving behavior under full occlusions in the environment (Chapter 3):
   (a) An object management system with imaginary targets is proposed.
   (b) An sampling-based occlusion-aware planning algorithm is proposed.
3. A novel ADS platform is proposed, and the effectiveness of the proposed perception and planning algorithms are demonstrated with the help of this platform (Chapter 4).

CHAPTER 2

Occlusion-aware Environment Modeling

To help the AV be cautious about the occluded areas in the environment, a model for the environment with occlusions is necessary. In this chapter, we present a method to represent the environment and identify occluded areas in the environment (Section 2.3). To support this method, we proposed a method for efficient detection and segmentation (Section 2.1), and a method for fast semantic occupancy mapping based on the results from detection and segmentation (Section 2.2). The occluded area will be represented in the generated occupancy map.

2.1 Joint Object Detection and Semantic Segmentation

3D object detection and scene understanding are two major perception tasks for 3D computer vision. The former task provides position and dimension information for dynamic objects of interest, while the latter task helps to understand the environment, which is usually accomplished by semantic segmentation of the sensor data. These tasks are important for autonomous driving and mapping. The detected 3D bounding boxes are useful for object behavior prediction, while the semantic information is useful for lane-keeping and static obstacle avoidance. Significant prior works exist using lidars for these tasks thanks to their superior ranging precision and robustness to certain environmental factors such as low/high lighting.

During the past decade, techniques for object detection and segmentation have advanced significantly. CNN have been widely used, including [68, 204, 230] for 2D object detection, [300, 99, 43] for 2D semantic segmentation, [311, 195, 290] for 3D object detection and [305, 315] for 3D semantic segmentation.

Few of these works combine the detection and segmentation tasks, several follow the panoptic segmentation scheme proposed by [128]. Examples include [251] on 2D images, and [315] on 3D point clouds. The simultaneous multi-task learning strategy is favorable for real-time applications, since it's usually computationally efficient to generate detection and segmentation results simultaneously, as some computations can be re-purposed.

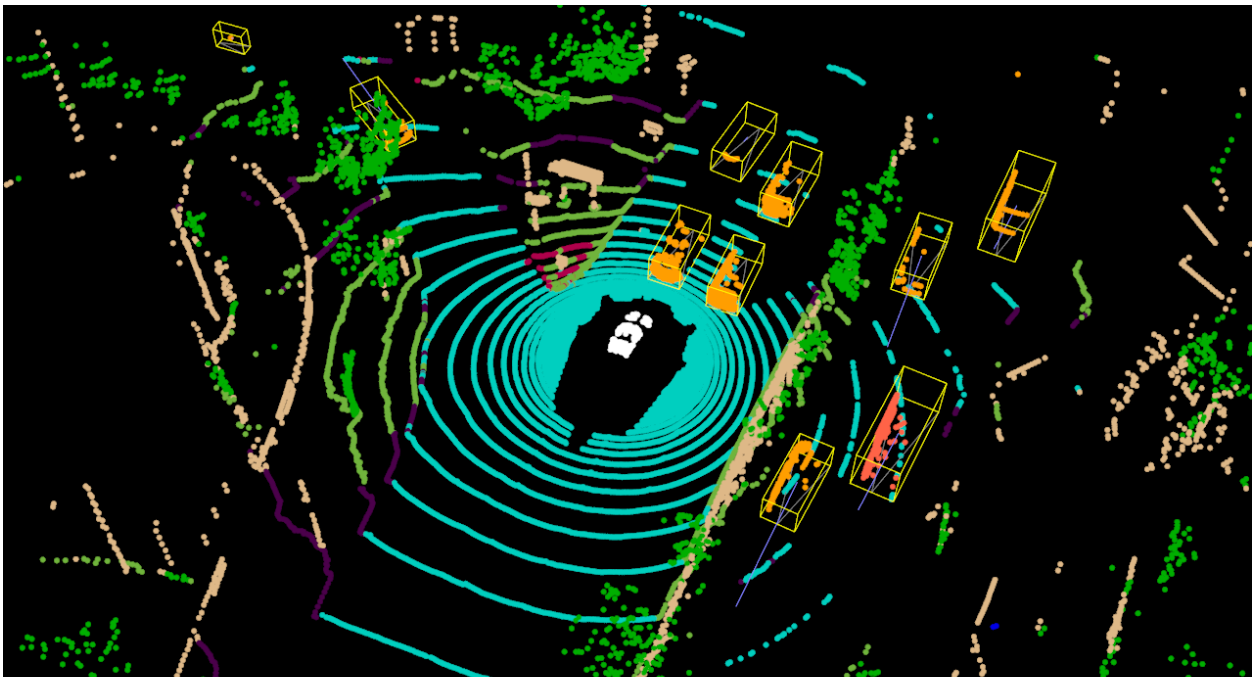


Figure 2.1: An example of simultaneous object detection and segmentation. *Legend of point clouds:* *Car* *Truck* *Pedestrian* *Vegetation* *Sidewalk* *Drivable Surface* *Terrain* *Man-made*

It's worth mentioning that 3D detection is different from 2D detection in the sense that a 2D bounding box is usually a tight bound of corresponding instance mask, while 3D detection (in outdoor scenes) provides object shape and orientation. Instance segmentation in 2D is also inherently different from 3D instance segmentation in terms of overlap, since there may exist occluded objects in 2D images where pixels of one object may locate inside the bounding box of another object, which is not the case in 3D point clouds, where bounding boxes should not overlap under normal situations. These differences make the design of panoptic segmentation methods different between 2D and 3D data.

In this section, we propose a method called Voxel-based Implicit Network (VIN), which takes a 3D lidar point cloud as input and reports both 3D object detection and semantic segmentation results as the outputs. Our main contributions include:

1. A semantic branch from a voxel-based 3D object detector which adds little computation overhead for the additional output. The semantic branch can be trained with weak supervision. The performance with just **0.1%** semantic labels, after some training, was found to be on par with a model trained with full supervision.

2. A strategy to fix inconsistency between bounding boxes and point-wise semantic labels, validated by our experiments.

3. Improved semantic and panoptic segmentation quality compared with state-of-the-art methods based on the results from the nuScenes-lidarseg dataset.

2.1.1 Related Work

2.1.1.1 3D Object Detection

Due to the wide availability of various datasets including [81, 32, 65], 3D object detection has become a hot topic in computer vision. Grouped by the modality of sensors used in the detection, the methods in literature can be categorized as image only [44, 171, 39, 261], point cloud only [195, 311, 282, 290, 215] and multi-modal fusion [221, 168]. Among the lidar-based detection algorithms, the approaches include feature extraction by points, voxels, projected images, and combinations [215, 134]. The point-based approaches [195] apply Multi-Layer Perceptron (MLP) and gather feature from points in the same group, while the voxel-based methods [311, 290, 221] first assign the point cloud into voxel grids, and then CNN modules are applied to the voxels. The projection-based methods [168] project the point cloud into images in a perspective view and then features are extracted using 2D CNN models. After the feature extraction, these methods usually generate 3D bounding boxes in a single-staged or two-staged manner ([154] and [204] respectively).

2.1.1.2 Point Cloud Semantic Segmentation

Point cloud semantic segmentation is an emerging research field taking advantage of datasets including [14, 32]. Similar to 3D object detection using point clouds, the segmentation algorithms also can use different approaches. PointNet[196] and its successors [197, 233] directly operate on a point cloud array and report point-wise semantics. Voxel-based frameworks [57, 5, 48] extract features for voxel grid convolution and reports voxel-wise semantics. Due to the cubic growth of computational cost with the number of voxel grids, sparse convolution [282, 48] is widely applied for these methods. Voxels are usually sliced in the 3D Cartesian coordinate, while there are also methods [103, 309, 315] using the polar or cylindrical coordinate systems. Methods that leverage projected images [273] for semantic segmentation take advantage of well-explored techniques developed for the 2D segmentation.

Different representations for point clouds have respective strengths and drawbacks. Point-wise feature operation preserves the granularity of the original point cloud, but suffers from heavy computation cost. Voxel-wise feature operations are fast, but suffer from lower precision introduced by the rasterization. Projection-based methods can take advantage of handy 2D convolution structures and their efficiency but need to learn to reconstruct the 3D object shape. In our approach, we en-

hance the voxel-based framework for its efficiency, by using the implicit representation introduced in Section 2.1.2, to enable precise point-wise predictions.

Recently, many researchers [86, 164, 155] work to advance the ability of point cloud semantic segmentation with fewer labels, namely weakly or semi-supervised semantic segmentation. Most of them depend on unsupervised algorithms [86, 155] to cluster the point cloud, or depends on consistency between data frames [164]. In this section, we propose a method that can handle weak supervision with the help of object detection results.

2.1.1.3 Joint 3D Detection and Segmentation

Both 3D object detection and environment segmentation are important tasks for autonomous driving. Relatively few papers combine the two tasks to provide richer information for downstream modules. [168] achieved joint 3D object detection and point cloud segmentation by combining the features from the lidar point cloud and camera image and extrinsic projection to collect additional point cloud features from the image feature map. [240] improved semantic segmentation performance by adding object detection as an auxiliary downstream stage for additional supervision, which shares a similar structure as our approach, but they started from a point-based semantic backbone. In addition, the two approaches cited above only produce results in a Field of View (FoV) limited by the camera. In this section, we aim to develop detection and segmentation results without this limitation.

There are many other methods [103, 315] that produce panoptic segmentation results of a point cloud. Their output point cloud, which are slightly different from those generated by joint 3D detection and segmentation, contain semantic labels and instance IDs. [103] proposed a shifting network module to predict whether a point belongs to a certain instance or not by location regression. [315] proposed a panoptic segmentation framework based on a cylindrical representation of the point cloud augmented by asymmetrical convolutional modules. Compared with these methods, our method achieves panoptic segmentation by generating instance labels with predicted bounding boxes.

2.1.1.4 Implicit Representation

Many recent computer vision algorithms use 2D or 3D grids which result in loss of data granularity. To preserve precision, a possible solution is to operate on the raw points, and another way is to use implicit representation such as signed distance functions [166, 186]. The key insight behind the implicit representation is to learn a function to represent the input location, instead of learning the collection of per-location predictions given the feature map input. The prediction can be occupancy of the location as used in [166], or a direct semantic label as what we used in this section. A major

benefit of this approach is that we can predict continuously, i.e. prediction can be obtained for arbitrary query positions, whether it's aligned with the original data or not.

2.1.2 Methodology

Different from most methods mentioned in Section 2.1.1, our goal is to generate 3D bounding boxes and point-wise semantic labels simultaneously given a point cloud input. On top of these results, panoptic segmentation predictions can be generated trivially. First, the backbone applicable to our algorithm will be described. Then the details of the proposed semantic branch will be explained. Finally, the panoptic segmentation will be covered. Figure 2.2 illustrates the structure of the proposed method.

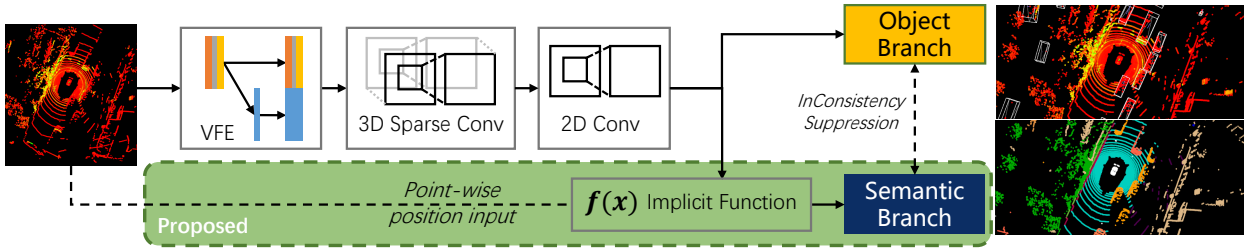


Figure 2.2: The overall structure of the proposed joint 3D object detection and semantic segmentation framework

2.1.2.1 Backbone Network

Our method can work with any voxel-based detectors that predict bounding boxes based on a feature map. To show the effectiveness of our semantic branch, we use CenterPoint [290] as our backbone. CenterPoint follows the VoxelNet [311] structure, consisting of voxel feature extraction layers, a 3D convolutional backbone, a bird's eye view 2D convolutional backbone and several detection heads (see the upper branch of Figure 2.2. After object proposals are generated for each location, they are passed through a Non-Maximum Suppression (NMS) or Circular NMS module to collect final predictions.

2.1.2.2 Semantic Rendering Branch

Inspired by [129], we propose a semantic rendering branch that learns a continuous function to generate semantic distribution prediction for each input position and reduce rasterization error. The semantic branch prevents granularity loss by using an implicit function. This process is achieved by using a lightweight MLP module with inputs coming from both point positions and local voxel

features. Different from PointRend which uses both coarse and fine-grained features, we only use the original global feature map from the convolutional backbone since our base framework is a single-stage detector instead of the two-stage MaskRCNN [99] used in [129]. In our experiment, the MLP of the semantic branch consists of 4 layers with 256, 128, 64, and 32 channels, respectively.

Given a query point $q = (x_q, y_q, z_q)$ from the point cloud input and backbone feature map $\mathbf{M} \in \mathbb{R}^{C \times D \times H \times W}$, we first find the grid position $(i, j, k) \in \mathbb{N}^{D \times H \times W}$ in the feature map where the query point lies, then the semantic distribution $s_q \in \mathbb{R}^S$ with S classes of the point are generated by the MLP module $f : \mathbb{R}^{3+C} \rightarrow \mathbb{R}^S$ formulated as

$$s_q = \text{softmax}(f([x_q - cx_i \; y_q - cy_j \; z_q - cz_k \; \mathbf{M}_{\cdot ijk}])) \tag{2.1}$$

where $(cx_i, cy_j, cz_k) \in \mathbb{R}^{D \times H \times W}$ represents the real-world center of the grid in the feature map indexed by (i, j, k). If the query point is outside of the voxel grid, then the voxel closest to the point is selected. This module f will be supervised by the point-wise semantic labels from the dataset.

Since only the position of the query point is fed into the function, unlike the method used in [315], the semantic branch can predict points that are not in the original point cloud. The decoupling between feature extraction and querying is beneficial when semantics are required to be extrapolated between points, which will be further discussed in Section 2.1.3.4. Note that in our backbone, the 3D voxel features are flattened and processed by a 2D convolutional network, therefore in our case $D = 1$. The proposed branch can be applied directly to a 3D voxel grid if the backbone framework produces a 3D tensor for the detection head.

With the network structure defined above, we define the loss function as

$$L = \alpha_{cls} L_{cls} + \alpha_{reg} L_{reg} + \alpha_{sem} L_{sem} \tag{2.2}$$

where we use focal loss with Gaussian kernels for the classification loss L_{cls} on the heatmap, and L_1 loss for the regression loss L_{reg} on box parameters, both of which are inherited from the backbone method we adopted from [290]. For semantic supervision, we use a combination of Lovász loss [16] and weighted cross-entropy loss for the semantic loss L_{sem}. In our experiment, the parameters for the weight loss are $\alpha_{cls} = \alpha_{sem} = 1, \alpha_{reg} = 0.25$.

2.1.2.3 Panoptic Post-processing

After bounding boxes and point-wise semantic labels are generated, panoptic segmentation results can be generated by assigning instance ids of the boxes to the points inside them.

Aside from panoptic label generation, object bounding boxes and semantic labels can be used to mutually recover the error in predictions. Inspired by the strategy introduced in [305], we developed a novel procedure named ICS to first fix inconsistent labels of the bounding boxes using a estimated label from point-wise semantic outputs and then inconsistent points will be fixed. The algorithm is described in the pseudocode (Program 2.1) below. The inputs for the procedure are the collection of predicted bounding boxes B and semantic point clouds P. Each box $b \in B$ has an object classification $K(b)$ with confidence score $s(b)$, and each point $p \in P$ has semantic classification $K(p)$ with confidence score $s(p)$. b also denotes the area inside the box. In the procedure description, K_{th} denotes the thing categories.

procedure ICS(B, P) ▷ It fixes inconsistent labels in-place
B: labeled bounding boxes, P: labeled point cloud
c_α, c_γ: tunable parameters with default value 1
m_p: score margin for overriding point label

Sort B descendingly by score
for $i = 1 \dots |B|$ **do** ▷ loop for fixing box label
  $P_i \leftarrow \{p \in P \cap b_i | K(p) \neq K(b_j)\ \forall j < i, p \in b_j\}$ ▷ select inconsistent points
  for $k \in K_{\text{th}}$ **do** ▷ for each semantic class k
    $P_i^k \leftarrow \{p \in P_i | K(p) = k\}$ ▷ select points with label k
    $\alpha_k \leftarrow {}^{|P_i^k|}/_{|P_i|}$ ▷ count criterion
    $\beta_k \leftarrow {}^{\sum_{p \in P_i^k} s(p)}/_{|P_i^k|}$ ▷ score criterion
    $\gamma_k \leftarrow 1 + s^{c_\gamma}(b_i) \cdot \mathbb{1}_{k=K(b_i)}$ ▷ correctness criterion
  $k^* \leftarrow \arg\max_{k \in K_{\text{th}}} \alpha_k \beta_k \gamma_k$ ▷ select the best class using the three criteria
  if $k^* \neq K(b_i)$ **then** ▷ if the best class is not the predicted one
    if $\exists\, j > i$ s.t. $K(b_j) = k^*$ **then**
      Swap $K(b_i)$ and $K(b_j)$ ▷ swap label with a box with lower score
    else
      $K(b_i) \leftarrow k^*$ ▷ override the box label by the best class
for $i = |B| \dots 1$ **do** ▷ loop for fixing point labels
  for $p \in \{p \in P \cap b_i | s(p) < s(b_i) - m_p\}$ **do** ▷ for each point with low score
    if $\exists\, j < i$ s.t. $p \in b_i \cap b_j$ **then**
      Continue ▷ ignore boxes with large overlap
    $K(p) \leftarrow K(b_i)$ ▷ override the point label by box label

Program 2.1: Pseudo code of the ICS algorithm

2.1.3 Experiment Results

Experiments are conducted on the nuScenes [32] dataset with nuScenes-lidarseg extension. This dataset is selected because it provides labels for both 3D object detection and point cloud semantic

segmentation. The nuScenes dataset contains various labels for different tasks, including 28k synchronized frames with multiple cameras, one lidar and multiple radars for training, and 6k samples each for validation and testing. The nuScenes-lidarseg extension provides point-wise semantic labels of 15 categories in total. It's a challenging dataset with adverse environment scenarios including dark nights and rainy days.

Our backbone method is CenterPoint [290] with a voxel size of 0.1m and without CBGS [314] or other test-time augmentation due to our hardware limitation. Our model is trained with 2 NVIDIA V100 16G GPUs, using AdamW optimizer with cyclic learning rate scheduling starting from $1e^{-4}$ and weight decay of 0.01. Please refer to [290] and [307] for details.

Selected qualitative results are shown in Figure 2.3. Our method is able to generate precise point-wise semantic labels while preserving the ability to predict accurate 3D bounding boxes. However, our method still suffers from the problem of ambiguity due to the sparsity of the lidar point cloud, common for all lidar-based detection or segmentation algorithms.

2.1.3.1 Quantitative Results

Semantic Segmentation Performance: For segmentation performance, our method is compared against state-of-the-art methods on the nuScenes lidar segmentation track. The main metrics used for comparison are the Intersection-over-Union (IoU) for each category. mean Intersection-over-Union (mIoU) and frequency-weighted Intersection-over-Union (fwIoU) numbers are also provided by the nuScenes benchmark. The results are presented in Table 2.1, compared with other methods, our approach performs significantly better in most "things" categories except for the traffic cones. Our backbone method does not preserve details in each voxel, thus it may not work well with small objects. On the other hand, existing methods outperform in many "stuff" categories, because our network needs to focus on objects to learn object detection well. Overall, our method outperforms the state-of-the-art methods by about 9.4 percentage points in mIoU.

3D Object Detection Performance: For the lidar-only object detection performance, we select a few other state-of-the-art methods on the nuScenes detection track. The overall performance is measured by mean Average Precision (mAP) and nuScenes Detection Score (NDS) proposed by [32] to capture not only precision performance, but also errors of other target properties. From the results shown in Table 2.2, we found that although our method suffers performance loss when adding a semantic branch, we still achieve good performance in most categories. This indicates that our method can produce semantic segmentation labels without much loss of detection performance. The most prominent gaps come from the "trailer", "bicycle" and "construction vehicle" categories, all of which are rare in the dataset. It's hard for the network to maintain the same performance when it learns to solve an additional segmentation task. Label balancing techniques can be applied in the future to solve the problem.

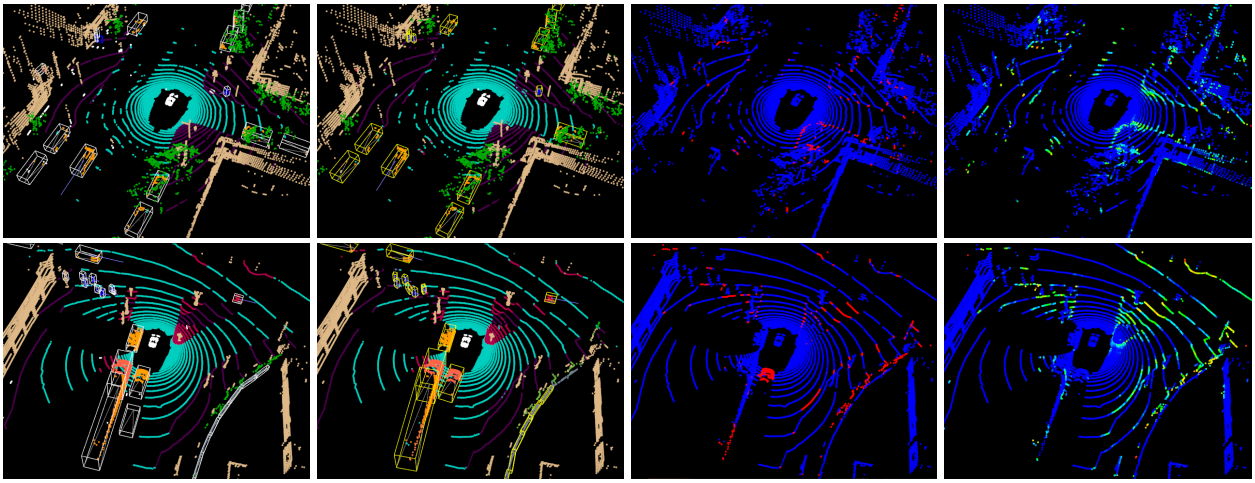


Figure 2.3: Qualitative results of VIN. First column: ground-truth semantic labels and bounding boxes; Second column: estimated semantic labels and bounding boxes; Third column: error points of semantic segmentation (labeled in red); Forth column: confidence score of the semantic segmentation (colors vary from light green to dark blue for scores from low to high). Best viewed in color and zoomed in.

Table 2.1: Comparison of semantic segmentation performance using the nuScenes validation dataset. For all metrics the higher the better, the best one is shown in boldface.

Method	Source	mIoU	fwIoU	Car	Truck	Bus	Trailer	Construction Vehicle	Pedestrian	Motorcycle	Bicycle	Traffic Cone	Barrier	Drivable Area	Other flat	Terrain	Man-made	Vegetation
SalsaNext [53]	[46]	58.8	82.8	81.0	65.7	77.1	38.3	18.4	52.8	47.5	4.7	43.5	56.6	94.2	60.0	**70.3**	81.2	80.5
MinkNet42 [48]	[46]	60.8	82.7	77.1	62.2	77.4	42.5	23.0	55.6	55.1	8.3	50.0	63.1	94.0	67.2	68.6	**83.7**	80.8
$(AF)^2$-S3Net [46]	[46]	62.2	83.0	80.0	67.4	82.3	42.2	20.1	59.0	62.0	12.6	49.0	60.3	94.2	68.0	68.6	82.9	**82.4**
VIN (ours)	-	73.7	84.3	**87.0**	82.6	91.7	63.2	49.9	79.7	82.2	**46.2**	**59.4**	74.7	94.5	67.1	68.5	**83.7**	81.2
VIN + ICS (ours)	-	**73.8**	84.4	**87.0**	**82.8**	91.7	**63.7**	**50.4**	79.8	**82.5**	46.1	**59.4**	74.7	94.5	67.1	68.5	**83.7**	81.2
VIN (seg only)	-	72.0	**86.3**	86.6	82.0	**91.8**	61.8	45.9	**80.3**	68.6	30.7	45.2	**75.4**	**95.8**	**73.8**	**72.6**	**85.3**	**83.3**

Panoptic Segmentation Performance: Panoptic segmentation is a side product of our method, however, it can be used to evaluate the combined performance of detection and segmentation. The metrics proposed in [128] is leveraged in our experiment, which includes Panoptic Quality (PQ), Segmentation Quality (SQ) and Recognition Quality (RQ). SQ and RQ are analog to point-averaged and instance-averaged IoU. We also adopt the replaced Panoptic Quality ($PQ^{\dagger}$) metric proposed by [103]. The performance of our algorithms is compared with three baseline methods in Table 2.3. Our algorithms outperform these baseline methods in most metrics except SQ. A possible fix is adding penalty terms for "stuff" points inside object boxes and "thing" points outside object boxes in future work, thus explicitly let the network distinguish between the environment and objects.

Table 2.2: Comparison of (lidar-only) 3D object detection performance on the nuScenes validation dataset. All metrics are the higher the better, the best one is underlined. Abbreviations: *CV - construction vehicle, TC - traffic cone, Ped - Pedestrian, Motor - Motorcycle, R - Reproduced*

Method	Source	mAP	NDS	Car	Truck	Bus	Trailer	CV	Ped	Motor	Bicycle	TC	Barrier
PointPillars [139]	[297]	28.2	46.8	75.5	31.6	44.9	23.7	4.0	49.6	14.6	0.4	8.0	30.0
3DSSD [286]	[297]	42.6	56.4	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
CenterPoint [290]	[297]	56.6	65.0	84.6	54.7	66.0	32.3	15.1	84.5	56.9	38.6	67.4	66.1
CenterPoint	R	50.7	60.2	82.6	50.3	63.8	30.8	12.0	79.9	43.9	22.3	60.8	60.4
VIN (Ours)	-	45.2	57.0	82.1	48.3	61.3	18.7	3.5	73.4	50.4	2.2	56.1	55.9
VIN + ICS (Ours)	-	46.3	57.6	82.1	48.1	61.1	22.7	7.2	74.3	50.4	4.1	56.6	56.4

Table 2.3: Comparison of panoptic segmentation performance on nuScenes validation set. All metrics are the higher the better, the best ones are highlighted in boldface.

Method	Source	PQ	$PQ^{\dagger}$	RQ	SQ	PQ^{th}	RQ^{th}	SQ^{th}	PQ^{st}	RQ^{st}	SQ^{st}	mIoU
Cylinder3D[309] + SECOND[282]	[103]	40.1	48.4	47.3	**84.2**	29.0	33.6	**84.4**	58.5	70.1	83.7	58.5
Cylinder3D[309] + PointPillars[139]	[103]	36.0	44.5	43.0	83.3	23.3	27.0	83.7	57.2	69.6	82.7	52.3
DS-Net[103]	[103]	42.5	51.0	50.3	83.6	32.5	38.3	83.1	59.2	70.3	**84.4**	70.7
VIN (ours)	-	**51.7**	**57.4**	61.8	82.6	**45.7**	53.7	83.6	**61.8**	**75.4**	80.9	73.7
VIN + ICS (ours)	-	**51.7**	**57.5**	**61.9**	82.6	**45.7**	**53.8**	83.6	**61.8**	**75.4**	80.9	**73.8**

2.1.3.2 Ablation Study

Inconsistency Suppression The proposed ICS procedure is used to eliminate inconsistency between bounding boxes and semantic labels, which is beneficial for downstream perception modules. This is demonstrated in Table 2.1, 2.2 and 2.3. For fixing semantic and panoptic segmentation labels, we set $m_p = 0.1$ in Algorithm 2.1 and achieve slightly better results. On the other hand, ICS made a difference on detection performance, especially in the 'barrier', 'construction vehicle' and 'bicycle' categories, indicating that ICS is effective. The improvement comes from using point predictions to help determine the object classification.

Weakly supervised segmentation Weak supervision can help to reduce label efforts when building datasets. Thanks to the nature of the implicit function representation, it doesn't require the full point cloud for supervision. Our method can handle weakly supervised segmentation tasks by feeding fewer labels during the training process. A key difference between our proposed method and single-task segmentation methods is that our method only get auxiliary information from the bounding box labels. The efficacy of our method is illustrated in Table 2.4. It can be seen that our method achieves robust segmentation performance even with just 0.1% semantic labels. However, the performance does not change monotonically with the amount of available labels, which is unexpected and requires further experiments to explain.

Table 2.4: Comparison of detection and segmentation performance on nuScenes validation set with different supervision levels. The label percentages denote the ratio of points used in training the semantic branch. All metrics are higher the better.

Method	mAP	NDS	mIoU	fwIoU	mIoU^{th}	mIoU^{st}	PQ	$\text{PQ}^{\dagger}$	PQ^{th}	PQ^{st}
Ours (full supervision)	45.2	57.0	73.7	84.3	71.3	78.3	51.7	57.4	45.7	61.8
Ours (10% label)	45.7	57.4	74.4	84.8	72.3	78.0	52.7	58.2	46.3	63.4
Ours (1% label)	44.7	56.6	72.6	84.6	69.5	77.9	50.4	55.9	42.8	63.1
Ours (0.1% label)	45.1	57.0	72.3	84.5	69.1	77.5	50.6	56.0	43.2	62.9
Ours (0.02% label)	44.5	56.6	70.6	83.8	67.1	76.5	49.0	54.5	41.3	61.8

2.1.3.3 Inference Efficiency

A major motivation to combine detection and segmentation in a single network is to reduce inference time and achieve lower latency when deployed. We compare our method with baseline CenterPoint in terms of inference time. Measured on a single NVIDIA 2080Ti graphics card, the original CenterPoint network with 0.1m voxel size achieves 6.0 FPS, while our method with the same voxel size and backbone configuration achieves 5.9 FPS but with the additional semantic segmentation results. This is much better than having a separate segmentation implementation. For example, a state-of-the-art method [46] alone takes 0.27 seconds (reported on nuScenes leaderboard, on an NVIDIA Tesla V100). When deploying our method on a vehicle, further optimization, including network distillation and refactoring using a highly efficient inference framework (e.g. TensorRT), can be implemented.

2.1.3.4 Utilizing the Implicit Function

Thanks to the fact that our semantic branch is merely based on the feature map produced by the convolutional backbone, semantics queries can be conducted at arbitrary positions in the space, as mentioned in Section 2.1.2.

Here we demonstrate two use cases of the query ability, semantic prediction on down-sampled point cloud and dense semantic map generation. The experiment results for the former case are reported in table 2.5. The down-sampling strategy is useful for reducing on-board latency or helping generate off-board labels for autonomous vehicles. In this experiment, only partial points (down-sampled from the original point cloud) are fed into the trained detection backbone and the semantics of the remainder of the original point cloud is predicted by either the semantic branch or nearest neighbor querying. It can be concluded from the experiment that our model can better capture the semantics in region with no lidar measurements when the original point cloud is down-sampled to reduce the inference time.

On the other hand, it's also possible to generate a dense semantic map by querying semantics

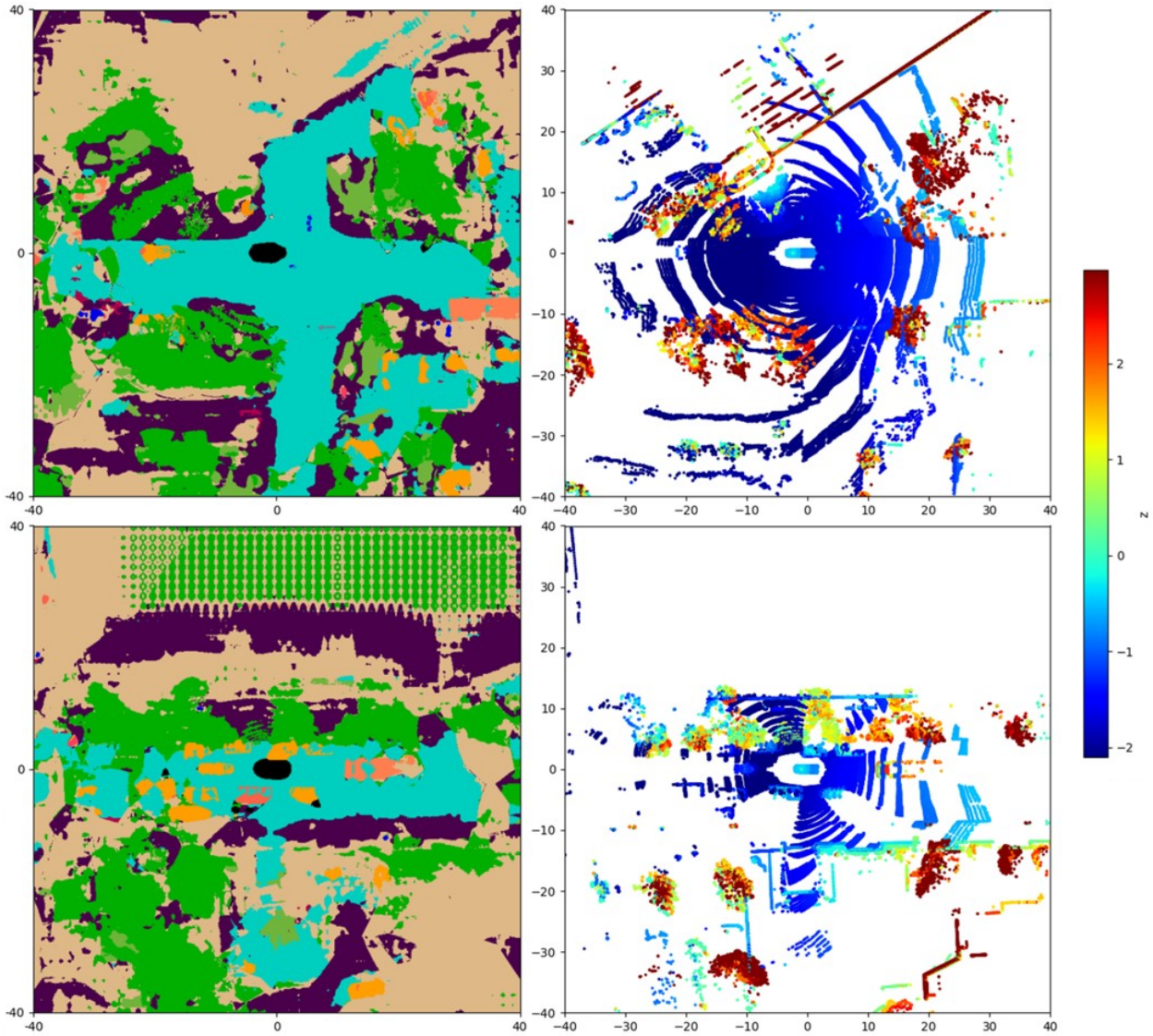


Figure 2.4: Two examples of generated dense semantic map on the nuScenes dataset. The left plots show the semantic prediction of the scene, the right plots show the input point cloud which is colored by the height. Please refer to Figure 2.3 for the color legend.

at grid points with the semantic branch. Figure 2.4 shows the semantic prediction of the scenes at certain height (the bilinearly interpolated height of the point cloud is used to create the figure). This dense map can be used as a BEV semantic map, which standalone models have been proposed in the literature (e.g. [203]) to estimate. It's useful for finding the semantics boundary for different areas and better interpreting the performance of the deep learning model.

Table 2.5: Comparison of segmentation performance with partial point cloud input on nuScenes. The best ones are highlighted in boldface.

Method	mIoU (random sample)			mIoU (beam sample)		
	100%	75%	50%	32 beams	24 beams	16 beams
Ours	73.7	**73.3**	**72.5**	73.7	**72.5**	**70.2**
Nearest Neighbor	73.7	73.1	72.2	73.7	64.6	55.9
(Inference time)	*169ms*	*133ms*	*104ms*	*169ms*	*125ms*	*96ms*

2.2 Fast Probabilistic 3D Occupancy Mapping

Mapping is an important task for robotic applications, including AVs. It is the foundation of subsequent tasks such as navigation and planning, and it's especially important for occlusion identification. If conducted offline, 3D mapping can generate HD Maps, focusing on the static portion of the environment. On the other hand, online mapping must understand both the static and dynamic elements of the environment, and must do so in real-time. This challenging application was recently explored by some researchers for robots exploring unknown environments, or for autonomous vehicles interacting with dynamic road users.

There are several 3D mapping formats including polygon mesh maps [268], volumetric maps[185, 77], surfel maps and point cloud maps [192]. Polygon meshes originate from computer graphics, representing objects by vertex points and the edges between them. Polygon meshes can be used to directly model object surfaces, but it is difficult to incorporate information with uncertainties. Point cloud maps [192] are commonly used in SLAM due to their simplicity and flexibility in transformation. However, it suffers from heterogeneous density which makes it harder for subsequent planning usage. Surfel maps [45] are based on point clouds with surface normal stored in each point. Lastly, volumetric maps are based on volumetric grids storing the map information including occupancy probabilities [185], Truncated Signed Distance Function (TSDF) [184] or semantic probabilities [77]. These grids are usually stored in a sparse data structure for better time and memory efficiency. Volumetric maps are particularly suitable for dense mapping, which produces a continuous representation of the environment and suitable for downstream tasks like path planning.

For semantic segmentation and mapping, much progress was accomplished recently using deep neural networks. Researchers have developed neural networks for 2D semantic segmentation [300] on images and 3D semantic segmentation [48] on point clouds. A novel joint object detection and semantic segmentation algorithm is also discussed in the previous section. With the ability to label point clouds, semantic mapping quality is improved by replacing traditional classifier labels [209] with ones from neural networks [77]. Semantic segmentation provides an alternative and a

direct way to handle dynamic elements in the environment, which is usually realized by detecting dynamic segments [91] in the map based on geometry.

The occupancy grid map based on Bayesian inference with sparse kernel [77] is chosen as our backbone method due to its efficiency and nature of dense mapping. In this section, we propose a method with two important features: real-time semantic mapping and dynamic object handling. The main contributions of this work are:

1. We proposed a probabilistic random sampling method for free space representation.
2. We developed an efficient free beam representation on top of a R-Tree.
3. Validation of the significant improvement in inference speed and accuracy of dynamic object removal through experiments. **100x** speed improvement was observed using a real-world point cloud dataset.

2.2.1 Related Work

2.2.1.1 Semantic Occupancy Mapping

Semantic mapping using lidars is usually based on volumetric representation due to its natural fitness for preserving a probability distribution over all occupied positions. Volumetric occupancy maps use 3D voxels with each voxel holding information about occupancy, Signed Distance Function (SDF) and semantic information. The information is updated incrementally when a new observation is available. Note that the occupancy can be regarded as a kind of binary semantics. Gaussian Process Occupancy Map (GPOM)[185] is a widely used representative framework that estimates the occupancy probability considering the correlation between grids. However, GPOM suffers from $\mathcal{O}(N^3 + N^2M)$ time complexity (for N range measurements and M query points). For faster storage, occupancy grids are usually stored using blocks indexed by a hashmap [179], resulting in a sparse voxel grid. For faster inference, [127] partition the voxels into blocks and achieve $\mathcal{O}\left(\frac{N^3}{K^2}\right)$ training and $\mathcal{O}\left(\frac{N^2M}{K^2}\right)$ inference complexity where K is the number of blocks. [63, 77] introduces Bayesian Kernel Inference into the mapping problem, further reducing the inference complexity to $\mathcal{O}(M \log N)$.

For semantic information, the Markov Random Field (MRF) [98] and Conditional Random Field (CRF) [209, 285] were frequently used in early semantic segmentation work. Later, deep neural networks became popular and showed good performance for both 2D images [300] and 3D point clouds [48]. With semantically labeled point clouds or depth images, semantic maps can be generated by fusing semantic prediction for each position by voting [209], CRF [137] or Bayesian inference [77]. The last approach [77] is used as the backbone in our work for its learning capability and efficiency.

2.2.1.2 Free space representation for Occupancy Mapping

In other volumetric maps like the TSDF map, free space is usually handled by ray casting along each scanning beam [104, 184]. However, in the occupancy map with Bayesian inference, training and testing steps are separated, which requires the free space to be represented by some geometries. Defined or sampled points along beams are commonly used in the literature [185, 200, 63, 252, 183] to represent free space. [185] uses the closest point on each beam to the query point as a free space data point. [252, 63] linearly interpolate between sensor origin and scanning hits with free space points. [63] also proposed to use point-to-line distance in the kernel calculation to reconstruct sensor beams from free space points during testing. [183] proposed adaptive order quadrature, which actually weighs sampled free points by the length of its beam. On the other hand, [200] sampled points between sensor origin and hits with uniform probabilities. Point-based free space representation can be efficient, but they are not complete in the sense that they cannot guarantee full coverage of all possible free space. Instead, the line-based representation to be presented in Section 2.2.2 can ensure the whole free space is covered.

In [104], the authors identified the problem of false negatives brought by close beams with shallow angles. This problem is not present in free space representation with sampled points, since the free points can be down-sampled before training, and points too close to each other will be eliminated. However, this problem is still relevant when free space is represented in other forms, which we face in our method. The solution to this problem will be discussed in Section 2.2.2 as well.

2.2.2 Methodology

In this section, the mapping framework will first be introduced, then the two proposed representations for free space modeling are discussed.

2.2.2.1 Semantic Occupancy Mapping

We adopt the semantic occupancy mapping framework based on Bayesian Kernel Inference with Categorical likelihood as introduced in [77]. The integration process of new range measurements in this framework is separated into two phases. First, the input point cloud will be down-sampled, and free space points will be sampled based on the measurements and added to input data points. Then the data points will be collected by each block to construct training samples per block. In the second phase, existing blocks in the occupancy map adjacent to training blocks will be collected as test blocks, and Bayesian inference will be performed with sparse kernels to update the Categorical posterior for each block. The key for fast inference is to use a sparse kernel defined below to

constrain the amount of data being considered.

$$k(x, x') = \mathbb{1}_{d<l}\sigma_0 \left[\frac{2 + \cos(2\pi r)}{3} + \frac{1}{2\pi}\sin(2\pi r)\right] \tag{2.3}$$

where x and x' are training data points and testing data points, $d = \|x - x'\|, r = \frac{d}{l}$, and kernel size σ_0 and kernel length l are hyper-parameters. With this kernel, the search of query data can be limited by l, resulting in fast local inference. Then the Bayesian update step can be formulated as

$$\alpha_*^k = \alpha_0^k + \sum_{i=1}^{N} k(x_*, x_i) y_i^k \tag{2.4}$$

where $\{\alpha_0^k\}$ is the prior Dirichlet distribution and $\{\alpha_*^k\}$ is the posterior distribution, $k \in \{1, 2, .., K\}$ is the index of categories. Please refer to [77] for more details. We further improve the whole process to meet real-time inference requirements in several aspects, as stated below:

1. Heavily templatize the code to improve computation performance.

2. Use a forward iteration to collect data points in each block, rather than query data points per block through an R-Tree, as in [63, 77]. This improves the time complexity of gathering training data from $\mathcal{O}(M \log N)$ to $\mathcal{O}(N)$, where M is usually of the same order of magnitude as N.

3. New free space representations are proposed to improve the inference speed of free space and provide extra flexibility for adjusting speed over accuracy. Details are presented in the following subsections.

2.2.2.2 Random Free Space Sampling

Point sampling is a common strategy to gather free space data points from range measurements as additional training data. Evenly spaced [252] or uniformly sampled [200] points can cover free space along a beam in arbitrary resolution. However, at locations near the sensor origin, sampled points are more crowded than those farther away, which makes the sampling inefficient. Based on this observation, we propose a sampling strategy with designed probabilities along each sensor beam, which is derived from the well-known unit-disk sampling problem.

Consider a 2D range sensor and a 2D grid occupancy map, the sampled points can cover the free space evenly with a linear probability along a beam. This behavior will be referenced as ***linear-weighted sampling*** in the remainder of the paper. The proof is presented here as follows. Assume we want to uniformly sample a point $(x, y) \in V = \{(x, y) | \sqrt{x^2 + y^2} < R\}$ in the visible

area V of the sensor, where R is the radius of the area. This implies the desired probability density function is

$$p(x, y) = \frac{1}{\pi R^2} \tag{2.5}$$

. When we convert the coordinates into a polar coordinate system, assuming the corresponding point is represented as (r, θ), then its density function should be $p(r, \theta) = |J|p(x, y) = rp(x, y) = \frac{r}{\pi R^2}$ where J is the Jacobian matrix of the polar transformation

$$x = r\cos\theta,\ y = r\sin\theta, J = \begin{pmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{pmatrix} \tag{2.6}$$

As the sampling is uniform in all directions, we can use the marginal probability of r as our sampling strategy

$$p(r) = \int_0^{2\pi} p(r, \theta)\mathrm{d}\theta = \frac{2r}{R^2} \propto r \tag{2.7}$$

Therefore we can prove that when the probability of a point sampled at a position is proportional to the distance from this position to the sensor origin, we can achieve uniform sampling in the circular FoV of the sensor.

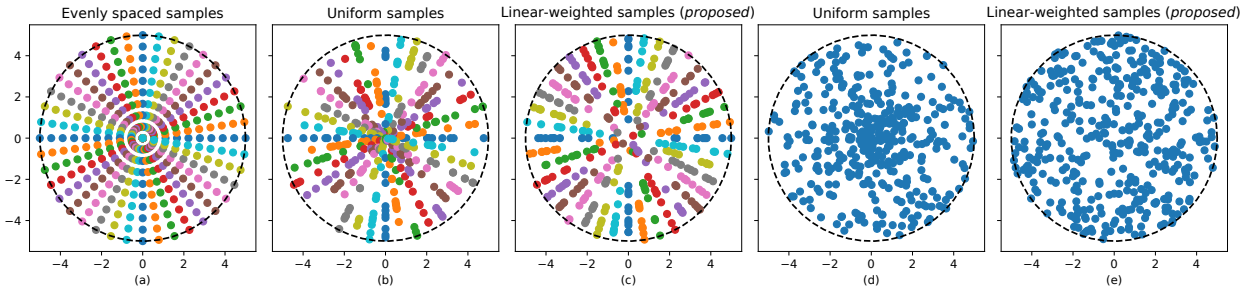


Figure 2.5: Comparison of different free space sampling strategies with fixed beam length. All plots include 135 sampled points and the color of each point represents which lidar beam the point belongs to. (a) Evenly spaced beams with evenly sampled points along beams. (b) Evenly spaced beams with uniformly sampled points along beams. (c) Evenly spaced beams with linear-weighted sampled points. (d) Points sampled uniformly with regards to angle and radius. (e) Points sampled uniformly in angle and linearly in radius.

The difference of several sampling strategies is depicted in Figure 2.5. Figure 2.5(e) corresponds to the exact case in Equation 2.7 and in Figure 2.5(c) points are linearly sampled along each beam but the beams are not randomly sampled, which corresponds to the application of sampling in the mapping process. It can be seen that points sampled evenly or uniformly along beams

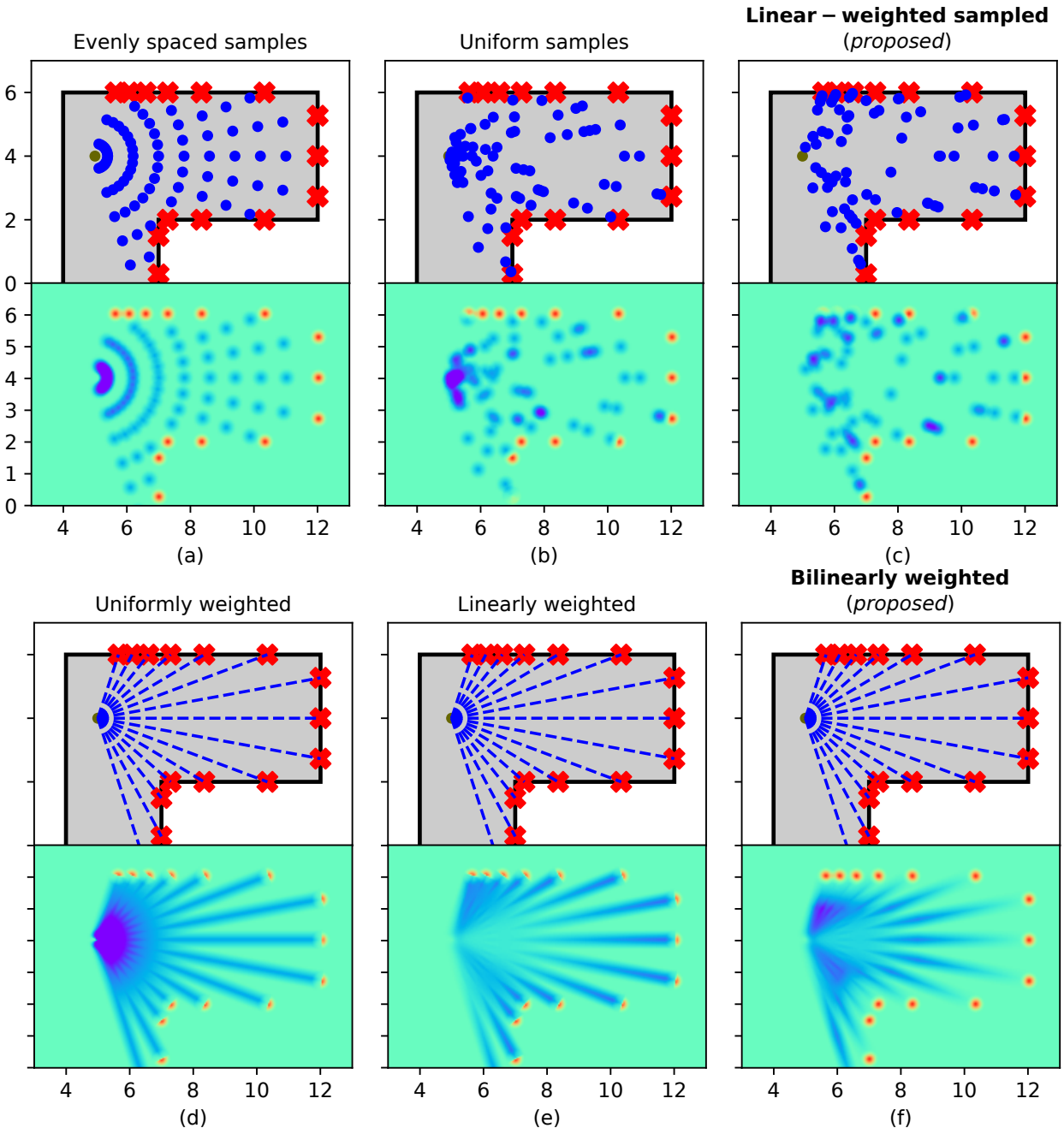


Figure 2.6: Illustrative examples of free space representation with simulated 2D scans. The top row shows the training data and the bottom row shows the inference results. Free space representations include (a) Evenly spaced sampled points, (b) Uniformly sampled points, (c) Linearly sampled points, (d) Line representation, (e) Line representation with linear weights, (f) Line representation with bilinear weights. The representations with bold titles are the most efficient ones with the least free space density overflow (purple area) and best preserving occupied space density (orange dots). Note that *evenly* sampling is deterministic while *uniformly* sampling is probabilistic.

all have clustered points around the origin, which reduces the sampling efficiency. The problem could be worse in the 3D point cloud from lidar sensors. Besides, The space near the origin is usually not a place of interest when we construct the map.

The strategies illustrated in Figure 2.5(a) and (b) are commonly used in the literature (see [252, 63] and [200] respectively). Aside from the difference in the distribution of samples, we also limit the number of samples per beam to a fixed count instead of a number dependent on beam length. This enables efficient parallel computation and will not affect free space coverage when the sampled points are very few due to resource limitations. Another example of sampled points is shown in Figure 2.6(a-c), where a similar difference can be found.

Even though this sampling strategy is more efficient than uniform sampling along each beam, a sufficiently large number of samples are still required to cover the free space without skipped grids. To mitigate this problem, we propose another free space representation in the next section.

2.2.2.3 R-Tree based free beam retrieval

Inspired by BGKOctomap-L [63], which uses sampled points to present free space, but calculates kernel (Equation 2.3) with distances from point to beamline, we directly calculate point-to-line distance without sampling points. In order to meet the real-time requirement, an efficient algorithm is required for querying point neighbors to a line or vice versa. Traditionally this problem can be handled by tree data structures including KD-Tree and R-Tree. However, for lidar beams all starting from the sensor origin, the bounding boxes of the beamlines usually have a large overlap and the leaf node will contain a large number of points, which degrades query to linear time complexity. A graphical explanation with a 2D world is shown in Figure 2.7. There are specialized tree structures for storing line segments and points, such as PMR-Quadtree [100]. Although this data structure can make points query efficient by splitting the regions along the lines, the tree can be very deep due to a large amount of split in our case.

To mitigate the problem, we use the projection of beamlines on a sphere around the sensor origin and construct R-Tree on the sphere, which helps the query of close point-line pairs to keep a logarithmic time complexity. Specifically, given sensor origin at (x_o, y_o, z_o) and a lidar range measurement at (x_i, y_i, z_i), we first calculate the beam representation in the spherical coordinate

$$\Delta_x = x_i - x_o, \qquad \Delta_y = y_i - y_o, \qquad \Delta_z = z_i - z_o \tag{2.8}$$

$$\varphi_i = \arctan\frac{\Delta_y}{\Delta_x}, \qquad \theta_i = \arccos\frac{\sqrt{\Delta_x^2 + \Delta_y^2}}{r}, \qquad r_i = \sqrt{\Delta_x^2 + \Delta_y^2 + \Delta_z^2} \tag{2.9}$$

where r denotes the range, φ denotes the azimuth angle and θ denotes the elevation angle. Then the beamline can be represented by a varying r from 0 to r_i with fixed $\varphi = \varphi_i$, $\theta = \theta_i$. After the

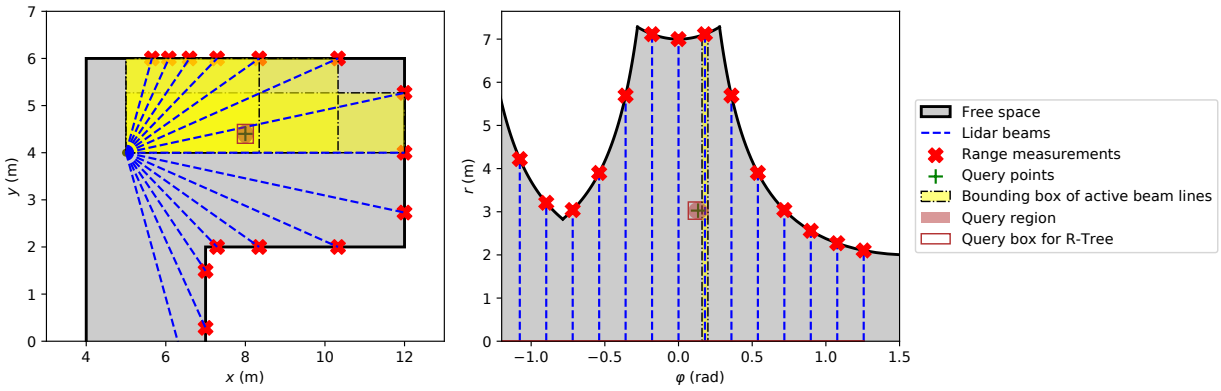


Figure 2.7: Line-based free space representation in 2D Cartesian and polar coordinate. Active beamlines are ones whose bounding box overlaps with the query ball region. In the right plot, the query region is projected from the left plot but the query box is generated using Inequalities 2.10 ($\theta_q = 0$). It can be found that there are much less activated bean lines in the right plot that those in the left plot.

coordinate conversion, R-Tree is constructed based on spherical coordinates $(r_i, \varphi_i, \theta_i)$. To make the coordinate system compatible with range queries, the R-Tree needs to be modified to consider the periodicity of coordinates.

Once the R-Tree is constructed, beamlines in proximity to the query points can be obtained using a "box" in the spherical coordinate system. By considering the query point (x_q, y_q, z_q) and the query range l (ideally equal to the kernel length), the lines in close proximity to the query point can be retrieved. This process involves converting the query point to spherical coordinates $(r_q, \varphi_q, \theta_q)$ using Equations 2.8 and 2.9. Subsequently, the retrieval operation is performed using the following box query:

$$r_q < r < \infty,\ \varphi_q - \frac{l}{r\cos\theta_q} < \varphi < \varphi_q + \frac{l}{r\cos\theta_q},\ \theta_q - \frac{l}{r} < \theta < \theta_q + \frac{l}{r} \tag{2.10}$$

Note that this query box is a close approximation of the ball around query point (see query boxes in Figure 2.7), and correct query results can be guaranteed by enlarging l slightly. Besides, the query can be problematic when data points are near the north or south pole of the sphere, but this is very unlikely in a lidar scan where elevation angles of points are usually in a small range around 0.

With the proposed spherical R-Tree data structure, we can achieve efficient kernel inference as illustrated in Figures 2.6(d) and 2.7. In Figure 2.7, the query point activates much fewer lidar beams in the polar coordinate than in the Cartesian coordinate. Nevertheless, a similar problem discussed in the previous section is encountered where the kernel value of being free will be exceptionally

large near the sensor origin. Following the same idea, linear weights can be applied to kernel values along each beam as shown in Figure 2.6(e). Furthermore, to ease the aforementioned problem of false negatives due to shallow beam angle from [104], we propose to use bilinear weights along the beam (the weight peaks at the middle of each lidar beam), which is illustrated in Figure 2.6(f).

2.2.3 Experiment Results

To exam the performance improvement of proposed free space representations, we conducted experiments on the real-world dataset SemanticKITTI [14]. SemanticKITTI is a large-scale dataset based on the KITTI dataset, with point-wise semantics from 19 categories. There are about 65k lidar measurements per frame in the SemanticKITTI dataset. We adopt the point cloud data with inferred labels provided by [77] for our experiment. The hardware used for the experiment is a desktop computer equipped with an 8-core CPU (Ryzen 2700), 16G RAM. The software environment is Ubuntu 18.04 with bundled Point Cloud Library (PCL) 1.8.1. We also adopt parameters used in [77], with kernel length $l = 0.3$, kernel scale $\sigma_0 = 0.1$ and Dirichlet prior $\alpha_0^k = 0.001$. During the experiments, the map resolution is set as 0.3m.

Our approach applies to any GPOM [185] variations, and in this section we test the performance using a semantic mapping framework to best demonstrate its efficiency. For conventional occupancy maps, classifications can be conducted using binary semantics, and map updates will be much faster than the 19-category classification. The baseline [77] used for comparison is compiled based on the official code repository provided by the authors except for minor modifications for frame time reporting.

2.2.3.1 Accuracy comparison on dynamic objects

There are two motivations for efficient free space representations: to reduce false positives in mapping and to handle dynamic objects. We evaluate **the recall of moving points detection** to assess the performance for dynamic objects. The moving label is directly provided by the SemanticKITTI dataset, we will count how many moving points are removed from the mapping process. Note that this metric is calculated across all categories, although the points from moving vehicle comprise most of the moving points. A negative example is shown in Figure 2.9(a2), several long blocks are classified as *car* due to aggregated mapping results from different frames. With proper free space representation as shown in Figure 2.9 (a1) and (c2), most false-positive car blocks on road can be eliminated, and the recall can be close to 1.

The relationship between free space representation and mapping quality is quantitatively illustrated in Figure 2.8. For point-based free space representation, key free space samples are those samples passed through voxel-grid down-sampling from PCL, which is used to collect training

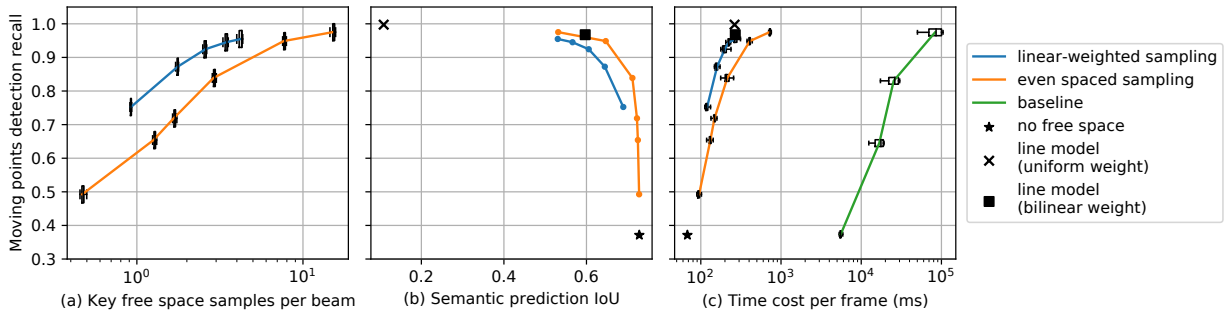


Figure 2.8: Quantitative comparison of mapping performance with different approaches. The number of samples per beam varies in the experiments, forming the lines in the plots. Baseline is from [77]. Comparison notes: (a) left-top is better. (b) right-top is better. (c) left-top is better.

data to reduce computation. More valid free space samples will lead to better mapping quality for free space, but key samples are particularly efficient using the proposed random sampling as shown in Figure 2.8b. However, with more free space samples, the semantic prediction accuracy will degrade because more voxels will be marked as free. On the other hand, for line-based free space representation, all valid space can be covered without fine-grained sampling. Thus we can achieve even better quality as in Figure 2.8b and 2.8c by a small margin. It is worth mentioning that without the bilinear weighting proposed in Section 2.2.2, there will be a lot of false negatives in the map as shown in Figure 2.9b2 and 2.8b. This demonstrates the problem originated in [104], where the query point will be falsely negative if it is too close to a lidar beam than its actual hit point.

2.2.3.2 Computation speed analysis

The quantitative results are shown in Figure 2.8 and 2.9. Our method is able to achieve real-time (>10Hz) mapping without free space consideration, and near real-time (4-10Hz) performance with the proposed free space representations. With the linear sampling strategy proposed in Section 2.2.2, good mapping quality is achieved. With the line-based representation and spherical R-Tree proposed in Section 2.2.2, the best mapping quality is achieved with around a 4Hz frame rate. The proposed line-based representation is able to achieve better mapping quality with less time consumption than point-based representations.

Further analysis of time complexity with different free space representations is evaluated in Table 2.6. It is worth noting that the number K of free space points per beam is vastly different in evenly spaced sampling and linear-weighted sampling. In the former approach, K can reach 50 if the maximum range is 50m and free space points are sampled with 1m gaps. However, K is fixed for parallelization and $K = 5$ is required to achieve similar quality in the latter approach. R-

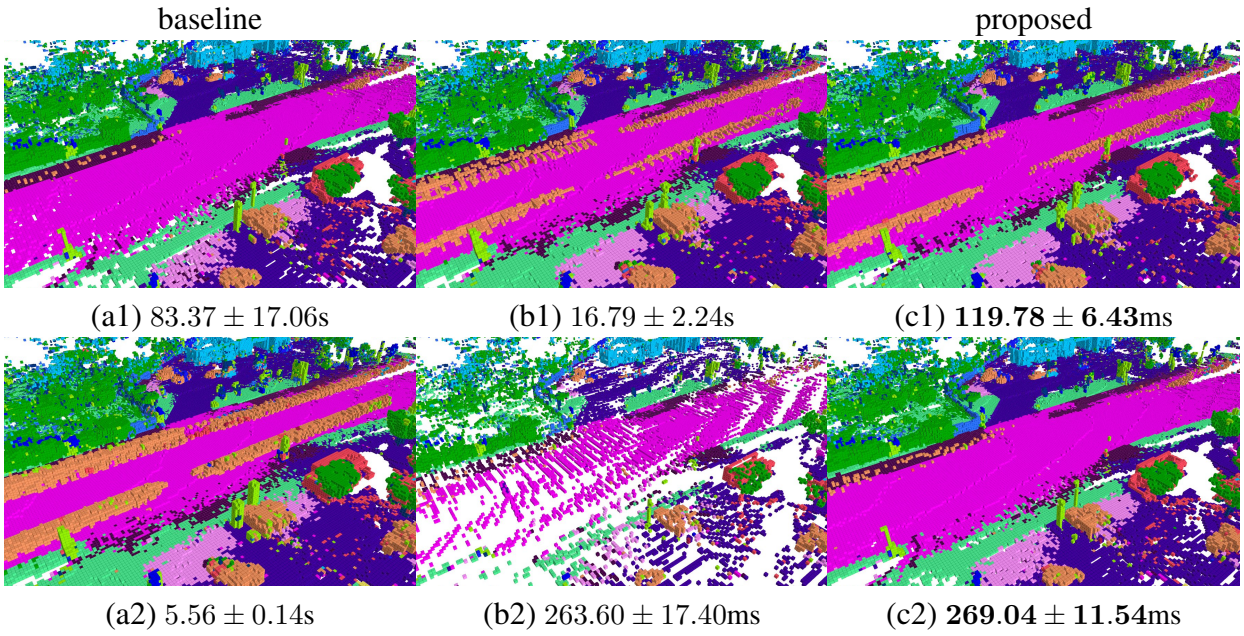


Figure 2.9: Qualitative results and frame time with different free space representations in a 3s segment of SemanticKITTI. (a) Commonly used baseline methods. (b) Other variants. (c) Proposed methods. Specifically, (a1)(b1) Evenly sampled points with 1m and 10m gap respectively (from [77]). (a2) No free space representation (from [77]). (c1) one linearly sampled point per beam. (b2)(c2) Line-based representation with uniform and bilinear weight respectively.
Legend of map grids: ■ *Car* ■ *Road* ■ *Building* ■ *Vegetation* ■ *Other ground* ■ *Terrain* ■ *Other vehicle* . Labels of the proposed methods are in bold.

Tree is used in both the baseline method and our line-based representation, resulting in logarithmic complexity. The difference is that R-Tree is used for point query in the baseline, but for line query in our method. Furthermore, bulk insertion is used in our method during the training data creation step.

Due to the better time complexity, our methods achieve more than **100x** speedup compared with the baseline method (see Figure 2.8c). Real-time performance with free space modeling can be achieved by using fewer semantics categories or limiting the range of inference positions.

2.3 Occlusion Modeling with Occupancy Maps

In the ADS, as mentioned in Section 1.2 and 1.3, an environment model is required as the input to the trajectory planner. To make the planner occlusion-aware, we first need an environment representation that takes the occlusion into consideration. In this section, we will proposed an approach to identify and represent the occluded area in an occupancy grid map.

Table 2.6: Time complexity comparison of different free space representations.
N - number of training data points. M - number of testing points. K - (average) number of free space points per beam. *: this part is parallelizable.

Method	Creating training data	Training	Testing
Baseline [77]	$\mathcal{O}(KN + N \log N)$	$\mathcal{O}^*(M \log N + N)$	$\mathcal{O}^*(M + N)$
Ours + evenly spaced sampling	$\mathcal{O}(KN)$	$\mathcal{O}^*(M + N)$	$\mathcal{O}^*(M + N)$
Ours + linear-weighted sampling	$\mathcal{O}^*(KN)$	$\mathcal{O}^*(M + N)$	$\mathcal{O}^*(M + N)$
Ours + line based	$\mathcal{O}(N)$	$\mathcal{O}^*(M \log N + N)$	$\mathcal{O}^*(M + N)$

2.3.1 Related Work

Occlusion is an important problem in the CV field. Most CV methods focus on the detection or segmentation of partially occluded and self-occluded objects. Some algorithms mitigate the occlusion issue by splitting the detection target into parts [296, 216], while many other machine learning algorithms improve the detection and segmentation performance by adding data augmentation into the training datasets [249, 308, 277].

The more challenging problem discussed in this dissertation is the full occlusion where the whole body of the object is invisible to the observer. There exist several methods that predict the existence of the hidden object using environmental or historical information. Li et al.[147] and Galceran et al.[76] proposed methods to track objects that temporarily enter the occluded area. Some other research works include [2], [227, 95, 114, 173] proposed perception algorithms that predict the existence of the pedestrian using the behavior of surrounding objects that can see and reacts to the crossing pedestrian. However, these two kinds of methods are both limited to specific scenarios. The former requires that the target is previously visible, while the latter requires that there is another road user and the occluded target has to be visible to it. On the contrary, our method does not depend on any assumption on road users and the detection of the occluded area is accomplished by modification on a grid mapping framework.

2.3.2 Occlusion Identification

Conventionally, the occlusion can be classified by the property of its source object as shown in Figure 2.10. The FoV of a vehicle is defined as the area where the onboard sensors can sense when there are no obstacles surrounding the vehicle. The areas beyond FoV are the blind spots. When there are objects surrounding the vehicle, some areas in the FoV will be occluded, which can be classified as static or dynamic occlusions. The dynamic occlusion can be trivially identified when you have a 3D model of the dynamic objects on the road (through a detection algorithm such as the one described in Section 2.1). However, identifying the static occlusion is hard with our premise

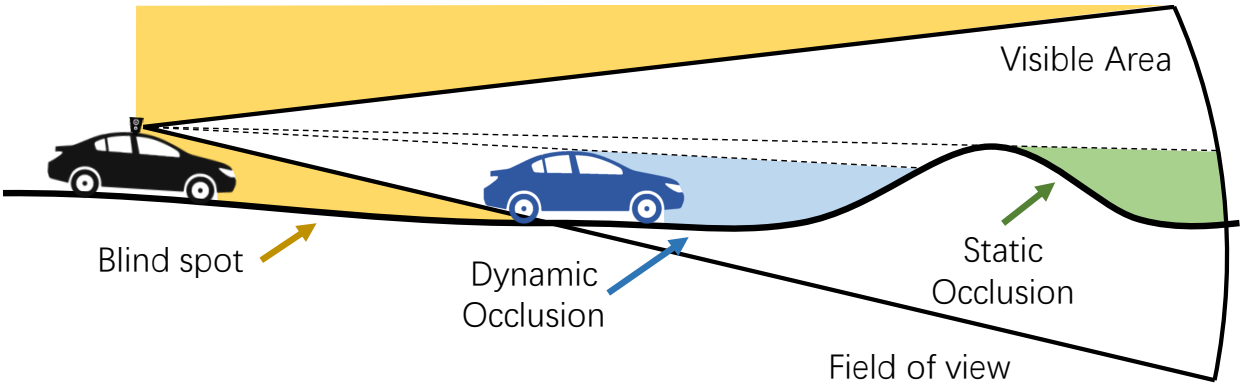


Figure 2.10: Sensor limitation and different kinds of occlusions.

to not rely on HD maps, which means that the algorithm has no access to vectorized maps. In this section, an approach to identify the occluded area in an 3D occupancy grid map will be introduced.

Given an occupancy grid map, the occluded blocks can be identified by the spherical R-Tree data structure as proposed in Section 2.2.2 with a query predicate different from Inequalities 2.10. The predicate for an query point being occluded is formulated as:

$$0 < r_q < r,\ \varphi_q - \frac{l}{r\cos\theta_q} < \varphi < \varphi_q + \frac{l}{r\cos\theta_q},\ \theta_q - \frac{l}{r} < \theta < \theta_q + \frac{l}{r} \tag{2.11}$$

The difference between the Inequalities 2.11 and 2.10 is that the range of r_q is flipped, reflecting the fact that the occluded locations are those behind the lidar measurements. Since the lidar point cloud is sparse, we don't have information between the lidar beamlines. The "thickness" of a lidar beamline is actually determined by the query range l. When l is bigger, more area will be marked as occluded given the same point cloud input.

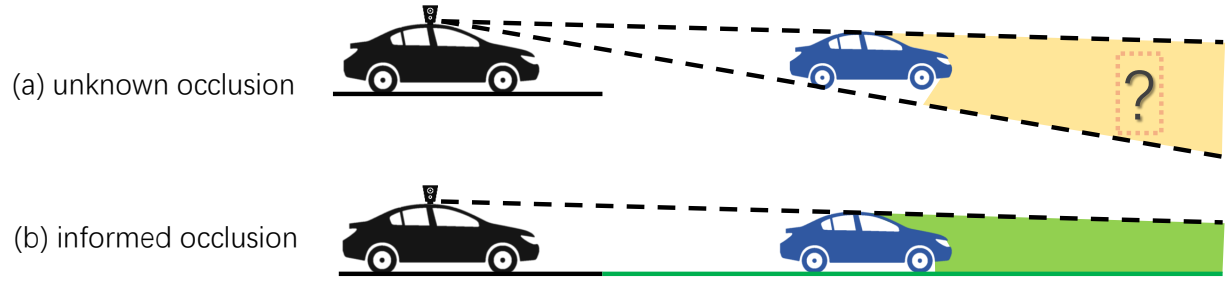


Figure 2.11: Occlusions categorized by whether prior information is available.

To identify all the occluded blocks in an existing occupancy map, we check the center points of all of the grids with previous observations. To explain the choice to ignore the grids with no prior information, we propose another way to categorize the occluded area, as shown in Figure 2.11. If

there is prior information about a grid in the map (e.g. we know this grid was free before), the occlusion of this grid is **informed**. Otherwise, the occlusion of the grid is **unknown** if there is no prior information. We ignore the grids with unknown occlusion because:

1. There are many spaces in the grid map that have no observations. Considering all the spaces would result in a lot of unnecessary computational cost.

2. Considering unknown occlusion could lead to false positives. For instance, it there is a building on a grid with no previous observations, it is not necessary to consider this grid during planning.

It is worth noting that the lack of informed locations can be reduced by storing the generated map during driving. Afterwards, the ADS can have prior information at visited locations by loading the stored occupancy map. This strategy is leveraged in the Section 4.2.2.

2.3.3 2D BEV Map Generation

With the occluded grids identified in the 3D occupancy map, we then convert the 3D map into 2D to make it easier for the planner to utilize. However, the critical information should be preserved as much as possible during the conversion. We adopt the multi-layer 2D grid map structure designed by [72], where each grid has multiple values with different semantics. Specifically, there are five relevant layers in the 2D map listed below:

1. **Ground Elevation**: height of the ground 3D grid.

2. **Ground Semantics**: semantics associated with the ground 3D grid.

3. **Occluded Height**: height (relative to the ground) of the top occluded 3D grid.

4. **Grid Status**: three states (*unknown*, *free*, and *occupied*) and a occlusion flag.

5. **Grid Status History**: the length of time since the last status change for the current grid.

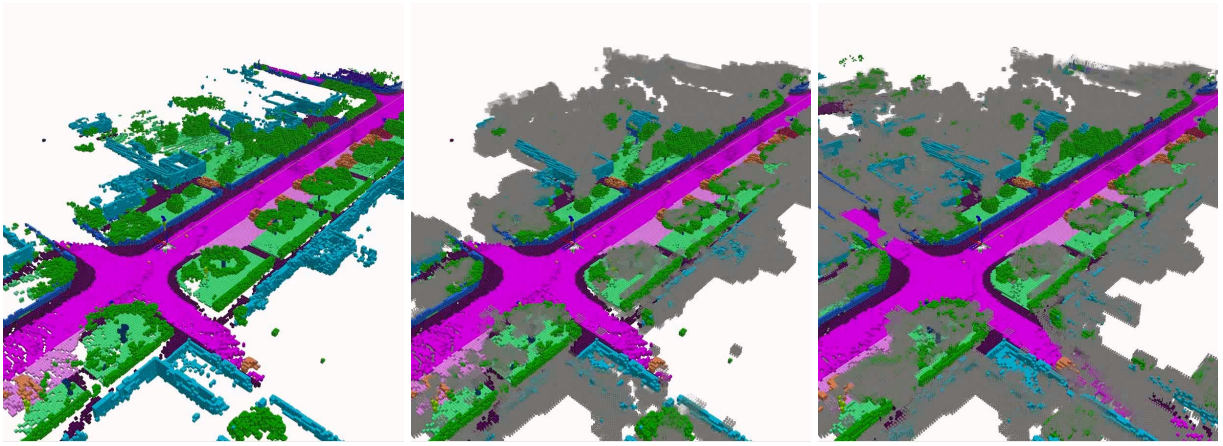
The ground 3D grid is the highest 3D grid with ground-related semantic labels (such as the *Sidewalk*, *Drivable Surface*, *Terrain* labels in Figure 2.1) at this 2D location. The top occluded grid is the the highest occluded 3D grid below a height threshold H_i at this 2D location. This threshold is used to ignore the occluded area that are too high to cause any driving risk. The occlusion flag is set only if the occlusion height is above another threshold H_l. The occlusion that is too short to case any driving risk (such as low bushes along the road) can be excluded by this threshold.

In the current design of semantic layers, the consideration of occluding object holes is omitted due to the complexity of representing multiple occlusion sections within a single grid. While modeling these sections instead of using a binary occlusion flag could potentially yield limited benefits, it may significantly escalate computational complexity.

The application of the grid status and its history will be detailed in the next chapter.

2.3.4 Experiment Results

To evaluate the proposed occlusion identification framework, we conducted experiments on the SemanticKITTI[14] dataset which is also used in Section 2.2.3. The occupancy mapping algorithm with the proposed occlusion identification function is tested on the sequence 04 of the dataset. The qualitative results are shown in Figure 2.12. The figure shows that the proposed identification method can find the occluded grids in the occupancy maps. It can also be found that more (informed) occluded areas are identified with a previously generated occupancy map.



(a) The occupancy map being constructed

(b) The occupancy map being constructed with occlusion identification.

(c) Occlusion identification with previously constructed occupancy map.

Figure 2.12: The qualitative results of occlusion identification on SemanticKITTI sequence 04. See the Figure 2.9 for the color legend. The gray dots in figure (b) and (c) are the identified occluded grids.

2.4 Summary

In this chapter, our focus was on efficiently representing occluded areas in the environment without relying on pre-existing maps. We proposed a series of three consecutive approaches for semantic

segmentation, occupancy mapping, and occlusion modeling. The key findings and conclusions of this chapter are as follows:

1. A novel framework for joint 3D object detection and semantic segmentation using lidar point clouds was proposed, demonstrating higher efficiency compared to separate implementations. The algorithm's efficacy and efficiency were confirmed through experiments on the nuScenes dataset, where it was compared with state-of-the-art methods for semantic and panoptic segmentation tasks. Furthermore, a modification allowing the learning of implicit representation of spatial semantic properties was proposed, which can be applied to any 3D object detectors that produce global feature maps.

2. Two free space representations were proposed to improve 3D occupancy mapping. The point-based representation with linear sampling offered greater speed and flexibility depending on the number of points per beam. On the other hand, the line-based representation ensured comprehensive coverage of all free space, resulting in improved mapping quality. The use of a spherical R-Tree facilitated efficient storage and querying of beamlines. Experimental validation on a real-world dataset using lidar point cloud confirmed the improvements in speed and mapping quality, enabling real-time semantic 3D occupancy mapping in large-scale outdoor scenarios.

3. A framework for the identification and representation of occluded areas in an occupancy map was introduced. The spherical R-Tree data structure was also utilized to identify occluded grids. To represent occlusion in an occupancy map, the proposed approach involved storing occlusion information as state and history values. Preliminary experiments conducted on SemanticKITTI demonstrated the effective identification of occluded areas in an occupancy map, irrespective of whether the map was generated online or offline.

In summary, this chapter contributes with novel approaches for efficiently representing occluded areas, encompassing joint 3D object detection and semantic segmentation, enhanced occupancy mapping techniques, and the identification of occluded areas within occupancy maps. The proposed methods demonstrate promising results and pave the way for AV planning with full occlusions in the environment.

CHAPTER 3

Occlusion-aware Motion Planning

With the occlusion modeling approach presented in Chapter 2, this chapter focuses on designing an occlusion-aware planning framework for AVs. The objective is to ensure safe AV behavior while avoiding excessive conservativeness in highly occluded environments. Inspired by the behavior of human drivers, the proposed framework responds to occlusions by creating virtual objects referred to as "*phantoms*".

Specifically, the first part of the framework (Section 3.1) handles the life cycle of the phantoms, and the second part (Section 3.2) finds the near-optimal trajectory based on the prediction of actual targets and the generated phantoms. The technical details of the planner are also covered in [304].

What sets our framework apart from other occlusion handling methods for AVs is its independence from HD maps or vectorized maps, such as Lanelet maps[193]. Additionally, it does not rely on assumptions about the source of occlusion or the behavior of other road users. These aspects contribute to the framework's versatility and applicability in diverse real-world scenarios.

3.1 Phantom Management System

The systematic approach for representing the occluded road users is described in this section. First, the occluded areas in the occupancy map are detected using the R-Tree data structure described in Section 2.3 and in [303]. Then imaginary moving targets are generated in these areas. Lastly, each phantom will be checked in every frame and removed if certain criteria are met.

Definition There are two kinds of imaginary targets: submerging objects $\mathcal{O}_s$ and phantoms $\mathcal{O}_p$. Submerging objects are previously detected targets that entered an occluded area at a certain time and never appeared again. On the other hand, phantoms are those that have never been detected, and their (hypothetical) positions are either beyond the field of view or in occluded areas (see Figure 2.10) all the time. Both kinds of imaginary targets can enter the visible area in the future and the planner should be prepared for them. In practice, $\mathcal{O}_s$ can be considered as $\mathcal{O}_p$ because they only carry additional prior information for initial states. So without loss of generality, we only

describe the strategy used for managing $\mathcal{O}_p$ in this section.

The management of phantom consists of three parts: generation of new phantoms, update of the phantom states, and elimination of old phantoms. These three processes are executed usually immediately after the tracking module in the AV system. We assume all the phantom have constant acceleration and constant heading angle, so the state update is trivial and efficient. The total number of phantoms is limited by a budget parameter. New phantoms will be generated only if the number of phantoms does not exceed the budget.

3.1.1 Generation

The generation of phantoms happens at each time step. To generate a new phantom, we first pick the feasible locations (occluded but not occupied grids in the occupancy map) and calculate the generation probabilities of those locations (Section 3.1.1.1), then a feasible grid is randomly chosen and the phantoms will be generated on the grid with a certain initial state (Section 3.1.1.2). At each chosen grid, a vehicle-like and a pedestrian-like phantom will be generated at the same time, with different limits on the initial states.

An objects manager will try to generate as many phantoms as possible at each time step until the budget for the number of phantoms is reached. Generation trials are not always successful, so a limit on the number of trials is also placed on the generation process so that the time consumption is constrained.

3.1.1.1 Generation Probabilities

Several aspects are modeled into the probability to make the phantom generation diverse and efficient. First, the more areas the phantoms cover, the better, so the locations with existing phantoms nearby are assigned a lower probability. Specifically, this probability can be calculated using a convolution with a matrix filled by ones as the kernel. Secondly, we put more phantoms near the ego vehicle, because they are likely more dangerous. Thirdly, it is unlikely for a newly occluded area to have targets in it if we asserted that there were no targets. Therefore, the locations with a long occlusion history will be assigned higher probability. It prevents over-conservative behaviors in our experiments. The phantom generation probability at grid (i, j) could be heuristically formulated as

$$p_{ij} \propto \frac{\max(0, \tanh\left(T_{ij}/a_T\right))}{N_{ij} \left\| c_{ij} - c_e \right\|_2} \tag{3.1}$$

where T_{ij} is the length of the occlusion history and a_T is a scaling parameter. N_{ij} is the number of existing nearby phantoms given the grid index (i, j). c_{ij} denotes the center of grid (i, j) and c_e

denotes the ego position.

3.1.1.2 Initial state determination

To make the generated phantoms cover dangerous cases more efficiently, the initial state of a phantom is selected so that it will collide with the ego vehicle in the short future (this time span is denoted as T_f). For the ego vehicle, we assume it follows the last planned trajectory S_{last} in T_f, while for the phantom, we assume its trajectory is straight and has constant linear acceleration, which will make the state update of a large number of phantoms efficient. Although the assumption can be strong for general road users, we believe that the simplistic motion models can be compensated with more phantom targets.

Given these assumptions, when generating a phantom at the grid (i, j), initial velocity v_{ij} of a phantom is uniformly sampled from the interval $[v_{ij}^{min}, v_{ij}^{max}]$, which is determined based on a predefined speed limit M_{speed}, acceleration limit M_{accel}, and a deceleration limit M_{decel}:

$$v_{ij}^{min} = \max\left\{0, \frac{1}{t}\left(d - \frac{1}{2}M_{accel} \cdot t^2\right)\right\} \tag{3.2}$$

$$v_{ij}^{max} = \min\left\{M_{speed}, \frac{1}{t}\left(d + \frac{1}{2}M_{decel} \cdot t^2\right)\right\} \tag{3.3}$$

where $d = \|S_{\text{last}}(t) - c_{ij}\|_2$, t is randomly chosen from $(0, T_f]$, $S(t)$ represents the way point along trajectory S at time t. After the velocity is determined, the acceleration of the phantom is $a_{ij} = 2(d - v_{ij} \cdot t)/t^2$. Besides, the heading angle of the phantom follows the vector from c_{ij} to $S_{last}(t)$. With the calculated behavioral parameters, the phantom target will collide with the ego vehicle at time t if the ego vehicle keeps following the current trajectory. Note that the limits M_{speed}, M_{accel}, and M_{decel} should be set separately for different types of phantom targets. The phantom generation will fail if either of the following conditions is not met:

1. There are feasible initial state candidate (most importantly $v_{ij}^{min} < v_{ij}^{max}$).
2. The line between c_{ij}, $S_{last}(t)$ does not intersect with any static obstacles.

3.1.2 Phantom elimination

The phantoms will not last forever, they will be eliminated if one of the two conditions below is met:

1. The phantom enters the visible area.
2. Newly observed occupied grids block the phantom (either the phantom or part of its trajectory is in the occupied grids).

3. The phantom becomes obsolete after it exists for too long or it is too far away from the ego vehicle.

In practice, the third condition hardly gets activated because the phantoms are generated to collide with the ego vehicle in future. Therefore the design of the related threshold will not be discussed in this section.

3.1.3 Integration with Object Tracking

Here we describe how the proposed phantom management system can be integrated with a object tracking module of an ADS in the Program 3.1. The integrated module will be referred to as "object management system" in the remainder of the dissertation.

---

procedure OBJECT MANAGEMENT STEP(B, P)
Inputs Δt: duration since last update, O: list of detected objects, M: the occupancy map with occlusion status and history, c_e: the position of the ego vehicle
States T: list of tracked objects, P: list of tracked phantoms
Parameters M_P: budget of phantoms, M_i: maximum iterations

Update T given O and Δt
Remove unnecessary phantoms according to Section 3.1.2.
Get all the occluded grids M_{occl} given T and M.
Get all the feasible locations L for phantom generation given M_{occl}.
Calculate the generation probabilities of these locations as p_L according to Equation 3.1.
$i \leftarrow 0$ ▷ number of iterations
while $|P| < M_P$ and $i < M_i$ **do** ▷ each iteration tries to generate a phantom
    Randomly sample a location c_p from L according to p_L.
    Randomly sample a collision time t from $[0, T_f)$ uniformly.
    Calculate the motion parameters based on c_e, c_p and t.
    if the motion parameters are valid **then**
        Add a new phantom p given the motion parameters to P
    $i \leftarrow i + 1$
for each phantom $p \in P$ **do**
    Update the state of the phantom given its motion parameters and Δt.

Program 3.1: The brief explanation of a update step of the object management system. The update of T in the procedure can be implemented using a conventional object tracker, such as the Kalman Filter.

3.2 Occlusion-aware Planner with Deterministic Sampling

In this section, the details of the planning algorithm will be discussed. Firstly, the structure of the fundamental algorithm will be reviewed. Then it is followed by the composition of additional planning costs for occlusion.

3.2.1 Related Work

There have been many researchers exploring the planning strategy for AV under occluded scenarios. The methods in the literature mainly fall into three categories according to the representation of occlusion: geometry, reachable sets, and raw sensor data. The methods using geometric representations [151, 207, 109, 176, 60, 177, 26] identify the occlusion by finding the corner points of the occluding objects. These methods usually rely on HD maps and only consider the occluded area on the road. Moreover, some of the methods [151, 109, 60, 177] represent the occluded area by edge positions on each lane, and the positions could be tracked over time. Notably, the Responsibility-Sensitive Safety framework [211] proposed by Mobileye also use this approach to design the safety criteria for scenarios with limited visibility [213]. The methods built on reachable sets [206, 294, 295, 175, 132] represent the possible occluded area by a reachable set. The advantage of this method is that occluded areas where vehicles (or other users) are very unlikely to exist will be excluded. It can prevent the ego vehicle from being over-conservative. However, the representation and propagation of the reachable set itself are costly as well. Yu et al. [294, 295] use samples to approximate the reachable set and achieve safe driving behavior in the intersections. [299] uses samples to represent the FoV of the ego vehicle. These sampling strategies reduce the complexity of representing occluded areas, but they are still highly dependent on precise environment modeling. The raw sensor data are usually adopted by learning-based methods [113, 254**?**], which adapt the planning system to occluded scenarios by training the planner with occlusion-related driving data. The training data, however, are usually limited and it is hard for learning-based planners to generalize to the various scenarios that AVs will encounter in the real world. Different from all the methods in the literature, the planning method proposed in this dissertation represents the occlusion states as grid properties in a grid map, which makes it independent of a detailed map of the environment.

For trajectory planning, the geometry- and reachable-set-based occlusion representations are usually combined with a heuristics-based or optimization-based planner, as the risk of occlusion could be explicitly expressed. The learning-based methods directly generate actions from the underlying machine learning models. On the other hand, Some methods [227, 109, 25, 272] use the POMDP to describe the other road users, and the occlusion is implicitly modeled as hidden states. Our method is built on a sampling-based trajectory planning algorithm [280], which is fast and

reliable.

3.2.2 Deterministic trajectory generation

The base planning algorithm used in this dissertation is from [280]. This planner models the trajectory generation as an optimal control problem, i.e. given cost function $J = \int_0^T L(x,u)\mathrm{d}t + E(T, x_T)$ the optimal trajectory is

$$\begin{aligned} u^* &= \arg\min_u J \\ s.t.\ \ \dot{x} &= f(x,u) \\ C(x,u) &\leq 0 \end{aligned} \tag{3.4}$$

and f denotes the system dynamics. L and E are the running cost and endpoint cost respectively, and C is the constraint. The state x is defined in the Frenet coordinate, that is

$$x = (s, \dot{s}, \ddot{s}) \text{ or } x = \left(d, \dot{d}, \ddot{d}\right) \tag{3.5}$$

where s and d are the two components of the Frenet coordinate (see [280]). This problem is usually intractable when the cost function does not have a simple structure. Therefore we use the deterministic sampling strategy [280] to find a near-optimal solution. An optimal trajectory to a simplified version[266] of Problem (3.4) is a quintic polynomial w.r.t. the time, therefore we first generate a set of candidate polynomial trajectories based on [266], and then we select the best trajectory that has the lowest cost with constraint checking.

The constraints $C(x,u)$ on the trajectories include:

- C_1: The (longitudinal and lateral) speed and acceleration along the trajectory all fall in a feasible range.
- C_2: The trajectory is confined inside the drivable area.
- C_3: No collision with other (real) targets.

And the cost function J is a weighted sum of the following cost terms:

- J_1: Smoothness
- J_2: Deviation from the target state
- J_3: Risk of leaving the drivable area
- J_4: Risk of being too close to obstacles

- J_5: Consistency of optimal trajectories
- J_6: Risk of collision with phantoms
- J_7: Risk of lack of visibility

All the constraints C_1 – C_3 and cost functions J_1 – J_5 are from [280], please refer to the Section VI of [280] for the definition of these terms. The cost function J_6 and J_7 will be discussed in the next subsection.

3.2.3 Cost terms for occlusion

To make the ego vehicle prepared for occluded road users (vehicles or pedestrians), two additional cost terms are added to the trajectory optimization.

3.2.3.1 Risk of collision with phantoms

The first cost is designed to prevent collision of the ego vehicle with the phantoms. The cost is defined such that it encourages the ego vehicle to find a path that is unlikely to cause a collision with the phantoms. It's formulated as

$$J_6 = \Phi_1 \left(-\frac{1}{\left|\widehat{\mathcal{O}_p}\right|} \sum_{p \in \widehat{\mathcal{O}_p}} ttc_p \right) \tag{3.6}$$

Here $\widehat{\mathcal{O}_p}$ is the most critical subset of the phantoms $\mathcal{O}_p$. For each phantom p, we calculate the Time to Collision (TTC) ttc_p based on the trajectories of the ego vehicle and the phantom, then the TTC values are averaged. The set $\widehat{\mathcal{O}_p}$ is selected based on TTC as well, it consists of K phantoms with the lowest TTC. A function $\Phi_1(\cdot)$ is used to reduce the impact of this term when TTC is very large. We set $\Phi_1(x) = \tanh(x) + 1$ in our experiments so that the risk of phantoms is reduced but not ignored when they are unlikely to collide with the ego vehicle.

3.2.3.2 Risk of lack of visibility

Clear vision while driving is crucial for safety. To improve awareness in obstructed situations with autonomous vehicles, it's important to encourage the vehicle to keep its distance from obstacles and drive in a direction that will improve the field of view. To emulate this behavior, we introduce another cost term as follows:

$$J_7 = \Phi_2 \left(\frac{1}{T} \int_0^T d(t) \mathrm{d}t \right) \tag{3.7}$$

Here $d(t)$ is the distance from the ego vehicle at time t to the nearest (occluding) obstacle around it. Similar to Eq. (3.6), the distance is averaged over the planning horizon and passed through a function Φ_2 so that its impact is reduced when the drivable area is not very narrow. Specifically, the function is set to be $\Phi_2(x) = (\max\{0, M_{dist} - x\})^2$ so that the optimal trajectory will not be skewed when it's far away from the occluding object (i.e. obstacles).

The existence of $\Phi(\cdot)$ functions is necessary because both the TTC and the distance can easily become large in a normal driving environment. Nevertheless, the definitions of the $\Phi(\cdot)$ functions are not critical as long as the cost function can be bounded by them. With all the proposed cost terms, the optimized trajectory should be encouraged to be more prepared for the potential risks introduced by occlusion on the road by slowing down and getting away from the occluding object.

3.2.4 Experiment Results

3.2.4.1 Experiment Setup

To evaluate the effectiveness of the proposed planning framework, we test the algorithm in several predefined risky scenarios that are common in real-world driving, based on [208] and our experience. These scenarios are illustrated and described in Figure 3.1. For the intersection scenario, we do not use real-world maps to create it like in [294], because most of the real intersections do not have severe occlusions, and the authors in [294] still have to place virtual obstacles to create the occlusions. Furthermore, for scenarios (b)-(e) in Figure 3.1, the occlusion is caused by dynamic objects. These scenarios have little to do with the road geometry and were not discussed in [294] and [295].

Experiments will be carried out using a 2D simulator, where all road users are treated as rigid bodies with standard dynamic states. In the simulation, every road user is assumed to have a flawless controller. As a result, during each state update, road users transition to their next state based on their designated trajectory. Specifically, the ego vehicle follows the trajectory from our proposed planner, while an Other Road User (ORU) follows the constant velocity trajectory defined by the scenario with a given initial speed.

During the following experiments, the initial speed and the target speed for the ego vehicle are both 10 m/s, and the initial speed for the ORU is 10 m/s in the intersection scenario and 4 m/s in the other scenarios. The path for the ORU is fixed and the initial location is randomly selected within 1m around the default initial position. The default initial positions and paths deliberately place the ORU in the occluded area of the ego vehicle at the beginning, **leading to a potential collision** if the ego vehicle maintains its original speed. Consequently, this creates challenging scenarios for the algorithms to handle.

Some of the key parameters of the proposed algorithm are set as the following: The maximum

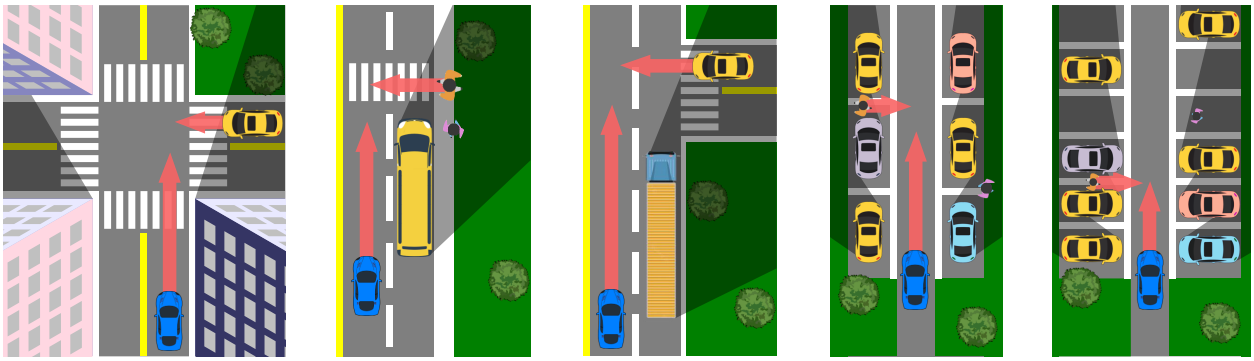


Figure 3.1: Illustration of experiment scenarios. In all the scenarios, the ego vehicle is driving upwards, when (a) An occluded car is driving into the intersection **without** the right of way (e.g. running a red light). (b) Pedestrians are coming out from the occlusion caused by the parked bus. (c) An occluded car is driving out without the right of the way. (d) Pedestrians are coming out from the occluded area between parked vehicles. (e) Pedestrians are coming out from the occluded area between parked vehicles.

iterations of phantom generation are set to half of the phantom targets budget. For vehicle-like phantoms, $M_{speed} = 20m/s$, $M_{accel} = 2m/s^2$, $M_{decel} = 2m/s^2$; For pedestrian-like phantoms, $M_{speed} = 5m/s$, $M_{accel} = 0.5m/s^2$, $M_{decel} = 0.5m/s^2$. Note that these values don't reflect the capabilities of actual vehicles and pedestrians, but only for phantoms. Besides, the distance threshold for J_7 is $M_{dist} = 10m$ and the default budget for the phantom count is 100.

The grid map input for the planner (described in Section 3.2) is generated based on the scenario and the position of the ego vehicle. The resolution of the occupancy grid map is set to 1.6m (the side length of each grid is 1.6). For each scenario in Fig 3.1 and each setting, we run 20 separate tests with different random seeds. We use the original planner as described in [280] as the baseline in our comparisons.

3.2.4.2 Evalution Metrics

To judge the performance of the proposed framework, we evaluate the phantom management (Section 3.1) and the motion planning module (Section 3.2) separately. For the former part, we want the planner to be **efficient** in computation and capture the **trajectories of real targets** well. A good manager should use fewer phantoms while retaining a great coverage of possible trajectories of the occluded real targets. While for the latter part, we want the manager to generate **smooth** trajectory and to be **safe** even under severe occlusion. A good planner should avoid hard deceleration while retaining a large distance between the ego vehicle and the ORU. Therefore three metrics are leveraged in our experiments:

Phantom count and closest distance between phantoms and the ORU: To evaluate the ef-

ficiency of phantom management, multiple test runs will be executed with different budgets for the number of phantoms. During each run, we will collect the lowest distance D_p between any phantoms and ORU o, calculated as

$$D_p = \min_{p \in \mathcal{O}_p} \min_{t \in T_{\text{occl}}} d(p, o) \tag{3.8}$$

where T_{occl} denotes the set of timestamps when o is occluded. In our experiment, there is only one road user, so o is always that road user and will be omitted.

Peak and median deceleration: To evaluate the smoothness, we collect the deceleration data of the ego vehicle during the test runs. The peak and median values of the deceleration will be reported and compared.

Closest distance between the ego vehicle and the ORU: To evaluate the safety, since there is not always a collision during the experiments, we collect the lowest distance D_e between the ego vehicle and the ORU as an indication of driving safety. D_e is formulated as

$$D_e = \min_{t \in [0,T]} d(ego, o) \tag{3.9}$$

where T is the time horizon of the whole run. A collision happened when $D_e = 0$

Aside from these main metrics, we also measure the scenario passing time and the minimum speed during each test run to capture the conservativeness of the methods. Each scenario will have a preset finish point to determine when the ego vehicle passed the scenario.

3.2.4.3 Experiment Results

In this section, 100 test runs will be executed for each scenario. Even though there is no limitation on the number of road users for our planning framework, only one ORU will be generated at the designed starting point in each test run for better consistency.

First, we compare the safety of the methods evaluated by the minimum distance D_e and maximum braking in each run, which is depicted in Fig 3.2. From the results, we can find that although the levels of difficulty of driving safely through these scenarios are not the same, our proposed planner can achieve lower collision risk by maintaining a larger distance from the other road user and avoiding most of the hard braking required for collision avoidance in all scenarios, thanks to the generated phantoms and the related cost on these phantoms.

It is worth noting that, in the parking lot scenario there is a huge variation in the deceleration profile from Figure 3.2(a). The reason behind it is that the original planner [280] generates diverging trajectories in the scenario where the pedestrian is appearing at different times. In some cases, the planner determines that it is safer to not decelerate (and even accelerate), which usually leads

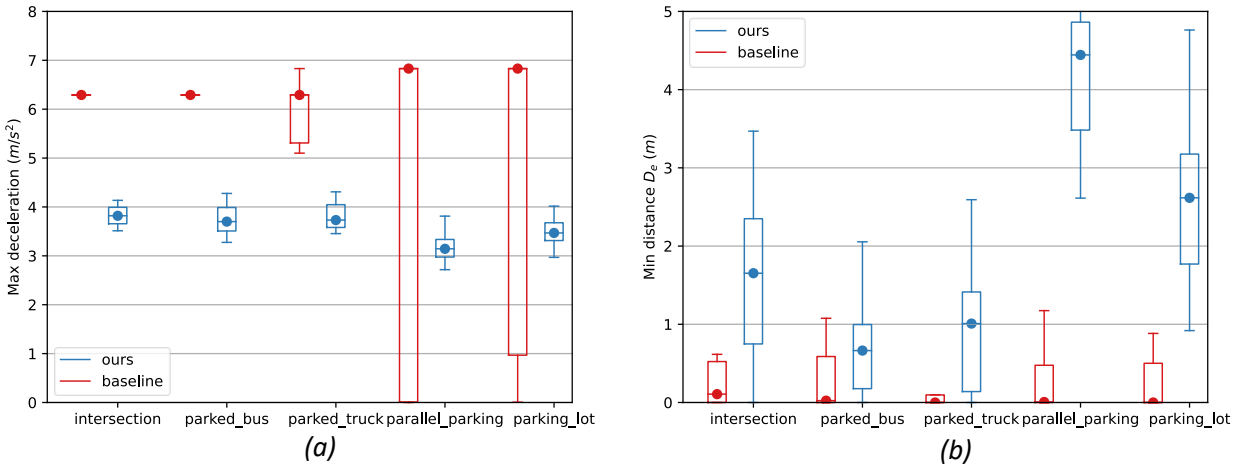


Figure 3.2: The safety comparison of the methods. (a) compares the maximum deceleration under each scenario (lower the better) (b) compares the minimum distance between the ego vehicle and the other real road user under each scenario (high the better). A distance of zero means collisions happened in that scenario. Note that the max deceleration in the scenario *intersection* and *parked bus* are very deterministic, and the quartile lines coincide. The baseline method is [280]

to collisions as shown in Figure 3.2(b). On the other hand, the proposed method always tends to decelerate when the ego vehicle is driving in a highly occluded scenario.

Secondly, in terms of the phantom generation efficiency, we plot the trade-off between the number of phantoms and the phantom effectiveness in Figure 3.3. The result shows that the effectiveness of the phantoms depends on the scenario. The larger area is occluded, the more phantoms are required to get the same level of coverage on the ORU, which is consistent with common sense. Nevertheless, regardless of the scenario, our phantom generation algorithm has good efficiency. With only **120 phantoms**, our method can cover the ORU with an error of less than 0.4m, which is much less than the resolution of the grid map input. For context, in [294] the planner needs as many as $\mathbf{2^{15}}$ *particles* for **each lane** to capture the potential behaviors of occluded objects.

Some overall statistics are reported in Table 3.1 as well. The results reported in this table are separated into two categories because [294] only supports the intersection scenario. The method [294] marked with † is the occlusion-unaware version of their algorithm used as a baseline in their experiments. Compared with the non-occlusion-aware method (first two columns), the proposed framework significantly reduced the collision rate and increased the minimum distance between the ego vehicle and the other road user. Since we set the scenario to be very challenging, it is hard to achieve zero collision rates without a huge influence on mobility. It can be noticed that the method proposed in [294] achieves better safety, but that is due to it being overly conservative. Under the setting in our experiments, the intersection is severely occluded and the planner in [294] will almost always come to a full stop before it can see through the roads of another direction, even

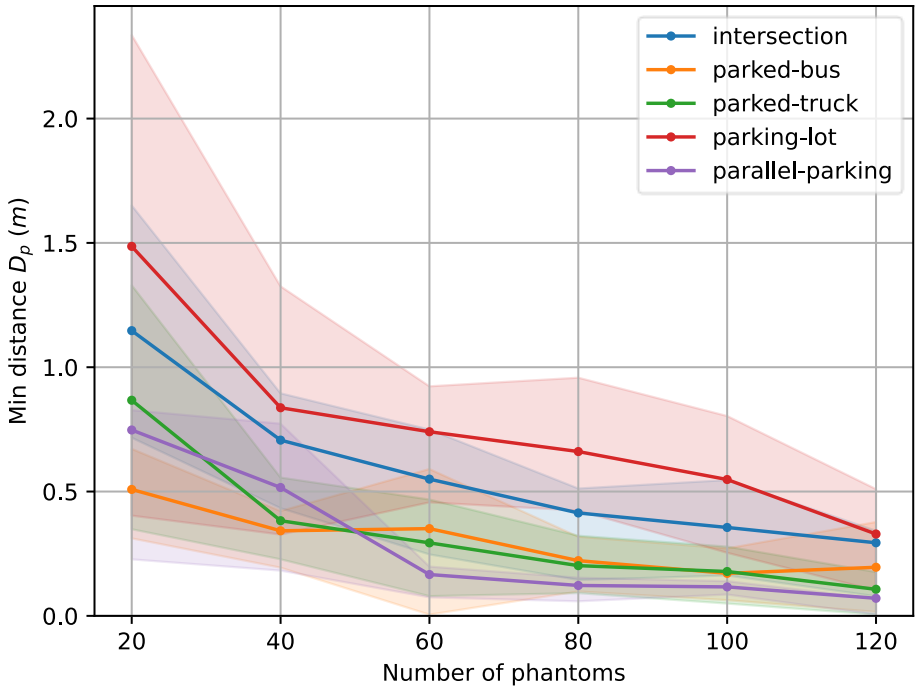


Figure 3.3: The efficiency comparison of different budgets for the phantoms. The lines and areas respectively denote the median and quartiles of the results from multiple test runs.

Table 3.1: Safety and mobility statistics of the planners.

Method	[280]	[294]†	[294]	[295]	Proposed
	(averaged over all scenarios)				
Crash Rate ↓ (%)	58	-	-	-	**5**
Min Distance D_e ↑ (m)	0.27	-	-	-	**2.27**
	(the intersection scenario only)				
Crash Rate ↓ (%)	47	43	**0**	31	4
Min Distance D_e ↑ (m)	0.25	0.42	**5.73**	1.87	1.58
Passing Time ↓ (s)	**2.91**	3.28	6.32	4.55	3.40
Min Speed ↑ (m/s)	**6.03**	3.68	0.17	1.10	5.35

if the ego vehicle has the right of way. This results in almost double passing time.

3.2.4.4 Case Study

To evaluate the behavior of the proposed planner, the experiments of two scenarios are explained in detail in this section. The Intersection and Parked Bus scenarios (Figure 3.1 (a) & (b)) are chosen due to their representativeness.

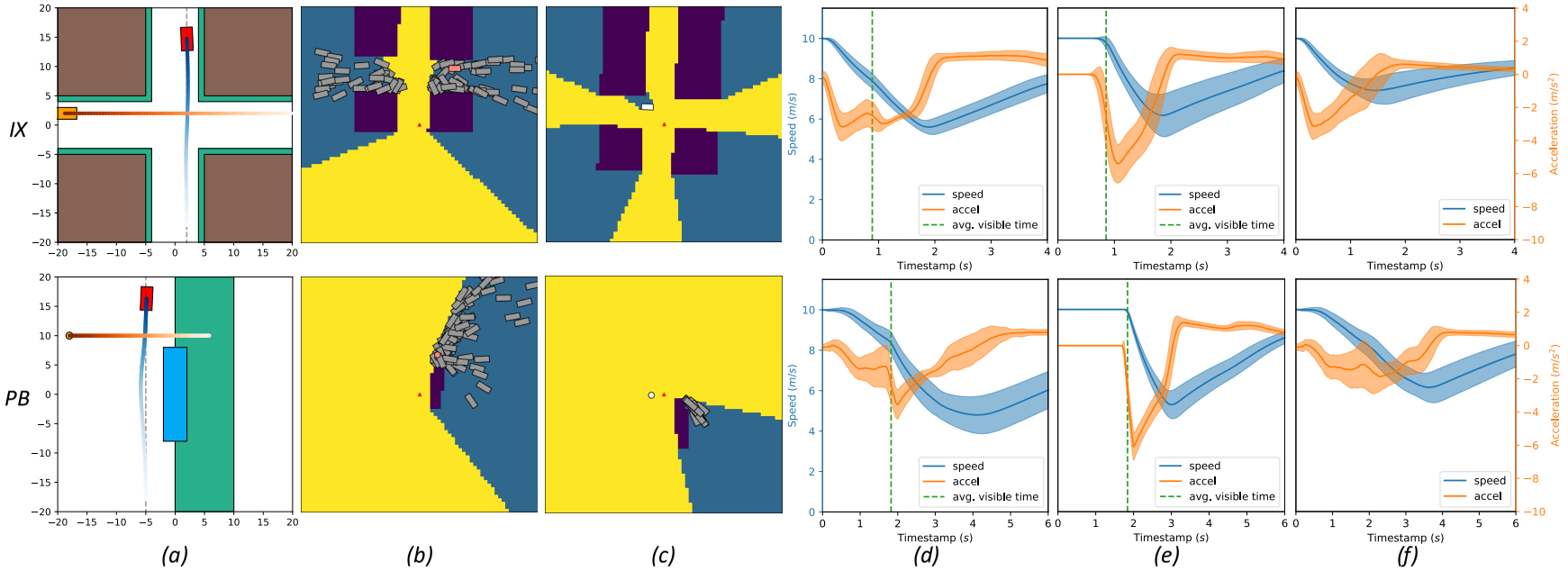


Figure 3.4: Experiment results of the intersection (IX) and parked bus (PB) scenarios. Column (a) shows the trajectories of the ego vehicle and the other road user. the brown area represents buildings, the green area represents sidewalks, and the blue rectangle represents the parked bus. Plots in (b)(c) show the simulation at two timestamps for each scenario. The environment is represented as a grid map, where the purple grids are the buildings, the yellow grids are visible areas and the blue areas are those occluded. The red triangle in the middle represents the current position of the ego car, and the boxes/dots represent vehicles/pedestrians respectively. The gray ones are the generated phantoms, the pink one is the occluded road user and the white one is the visible road user. (d)-(f) show the speed and acceleration profile of (d) the proposed occlusion-aware planner, (e) the baseline planner [280], and (f) the proposed planner with free intersection (no other road users). Note that (a)-(c) are captured in a single simulation, while (d)-(f) are calculated based on 400 simulations. The lines and areas denote the mean and standard deviation of the results from multiple simulation runs.

The summaries of the experiments conducted on these two scenarios are rendered in Figure 3.4, with 400 test runs for each scenario to obtain converged speed and acceleration profiles. In the intersection scenario, the proposed planner will slow down before it enters the intersection due to the generated phantoms. After the ego vehicle enters the intersection, all the phantoms are removed and no new phantoms are generated because all the occluded grids are newly occluded (see the strategy described in Section 3.1), then the ego vehicle started to accelerate back to the default speed. In the parked bus scenario, the proposed planner will let the ego vehicle slow down and nudge away from the parked bus, and then when it spotted the pedestrian suddenly appearing in front of the bus, it started to brake harder and move to the right to reduce the safety risk.

Comparing the plots in Figure 3.4 (d) and (e), we can find that the proposed framework will slow down earlier thanks to the presence of the phantoms. This results in a smoother deceleration compared with the occlusion-unaware baseline method. Compare the plots in Figure 3.4 (d) and (f), the proposed planner will slow down even if there are no real other road users around, but the ego vehicle started earlier to recover the desired speed if our planner confirmed that there is no actual risk in the surrounding environment. According to a user experience survey conducted in three Japan cities, this proactive braking behavior is acceptable for most human drivers, although it can be annoying sometimes [115].

3.2.4.5 Ablation Study

Table 3.2: The performance of the proposed planner with different weights in the parked bus scenario.

Settings	Crash Rate↓(%)	Min Dist.↑(m)	Min Speed↑(m/s)
Default	3	0.88	4.27
	(Different weight on $J6$, default $= 500$)		
$W_{J6} = 200$	5	0.87	4.68
$W_{J6} = 800$	2	0.98	4.09
	(Different weight on $J7$, default $= 10$)		
$W_{J7} = 5$	5	0.79	4.30
$W_{J7} = 20$	3	0.88	4.14

In this section, the influence of the weights of the cost function terms is discussed in Table 3.2. For the weight selection on other terms, please refer to [280]. The results imply that lower weights of J_6 and J_7 lead to higher safety risk, while a higher weight of J_7 leads to a minor degradation of the mobility. This validates the effectiveness of the objectives J_6 and J_7.

We also provide additional experiments to demonstrate the capability of the proposed algorithm with variation in numbers of ORU and the status of the occluding object in Table 3.3. In the first and

Table 3.3: The performance of the proposed planner with different scenario settings. (units omitted)

Settings	Method	Crash Rate↓	Min Dist.↑	Min Speed↑
		(the intersection scenario)		
1 ORU	ours	4	1.58	5.35
1 ORU	[280]	47	0.25	6.03
2 ORUs	ours	36	1.04	6.01
2 ORUs	[280]	52	0.20	7.04
		(the parking lot scenario)		
1 ORU	ours	0	2.68	1.93
1 ORU	[280]	69	0.35	5.86
2 ORUs	ours	41	0.75	3.44
2 ORUs	[280]	45	0.53	6.34
		(the parked truck scenario)		
static	ours	12	1.01	3.00
static	[280]	75	0.10	3.92
moving	ours	0	2.11	2.96
moving	[280]	44	0.23	3.92

second sections of the table, there will be another ORU generated in the scene with a randomized starting position and speed. In the third section, the truck is driving north at a speed of $1m/s$ in the "moving" setting. It can be found that the test cases are significantly more challenging with more ORUs but less challenging with moving occluding objects. Regardless of the change, the proposed method consistently achieves better safety compared with the baseline method without significantly affecting mobility.

3.3 Summary

In this chapter, we explored how to help AVs handle highly occluded scenarios safely and efficiently without relying on prior knowledge of road topology or environment models. The proposed framework operates under the assumption that an occupancy grid map is generated from sensor inputs by an upstream module. The key findings and conclusions of this chapter are as follows:

1. The concept of **phantoms** is introduced and elucidated. Phantoms serve as imaginary targets used to represent the behaviors of fully occluded road users with no observation history. They play a crucial role in preparing acAVs to navigate through occlusions on the road.

2. A phantom management framework is proposed, encompassing the generation, update, and

elimination of phantoms. The generation process incorporates probabilistic aspects and factors in the duration of occlusion history, thereby enhancing the efficiency of phantom generation. Importantly, the proposed phantom-based method effectively captures the possible trajectories of other road users, requiring fewer imaginary objects compared to some particle generation algorithms.

3. An occlusion-aware planning framework is presented, leveraging a deterministic sampling strategy. During experiments conducted in specially designed scenarios with high occlusion risk and collision possibility, the proposed framework achieves safer maneuvering without being overly conservative and negatively affecting mobility.

It is worth noting that the application of the proposed planner framework is not limited to autonomous driving but can be adapted for any robotic system operating in occluded environments. This design is flexible and doesn't impose great computation increase for the existing ADS. However, the proposed strategy is not a perfect solutions, and it sometimes suffer from the stochasticity brought by the random phantom generation. In future work, the quantization of occlusion risks can be further enhanced in the following ways:

1. **Calibration of the phantom generation probabilities.** The way we design phantom probabilities at occluded locations is pivotal for planning against risks associated with occlusions. The heuristic-based probabilities defined in Section 3.1.1.1, while straightforward and easily adjustable, do not accurately represent the existence likelihood of hidden road users. Exploring data-driven methods may yield probability distributions more in line with real-world behaviors.

2. **Utilizing the semantic information.** The joint detection and segmentation algorithm proposed in Section 2.1 reports the semantic information of the environment. However, the semantic information is not effectively leveraged in the phantom generation process. For example, it's usually unnecessary to generate vehicle-like phantoms on the sidewalk. Integrate semantic information into the object management system might further improve the efficiency of the phantom generation process.

3. **Quantification without phantoms.** Currently the proposed planner relies on hypothetical targets to symbolize occlusion risks serves to reduce computational demands. But with advanced data-driven techniques, directly quantifying the risk at specific locations might be desirable, bypassing the need for using phantoms as a proxy of estimation.

CHAPTER 4

Systematic Experiments

In this chapter, we will elaborate on how we test the efficacy of the proposed perception and planning algorithms when they are integrated into an ADS. First, a modular software stack will be introduced in Section 4.1 that supports both the simulated and real-world experiments with a unified framework. Then we will discuss how the experiments are designed and conducted in simulation and real-world tests sequentially through Section 4.2 and 4.3.

4.1 A Modular AV Stack

Motivated by the advancement of various techniques including machine learning, advanced range sensors, and computing units, researchers have suggested new approaches in various domains of AV [189, 111] and devoted a lot of time to evaluating them. However, due to the complexity and cost of a self-driving system, a major remaining limitation is that test vehicles are expensive, and an easy-to-use complete software stack is lacking.

To tackle this problem, we present our early attempt, named Cloud-and-Learning-compatible Autonomous driving Platform (CLAP)[302], aiming to contribute to more open and collaborative development of AV. CLAP[1] consists of essential algorithms in perception, navigation, cognition, planning, and control, and can be used both for simulations and for vehicle implementation. This platform has been used for our algorithm development. Our platform helped us rank in the top 3 among three tracks in the 2019 Carla AD Challenge[2] which aims at testing the driving performance of autonomous agents with different sensor setup. This platform has also been deployed on our experimental vehicles (see Figure 4.2).

We have published papers supported by this platform about algorithms of perception [305, 36], trajectory planning [37, 34] and control [278, 35]. Besides the essential driving function, the

[1]The platform is open-sourced under https://github.com/CLAP-Framework/clap

[2]The content of the challenge in detail can be found at https://carlachallenge.org/

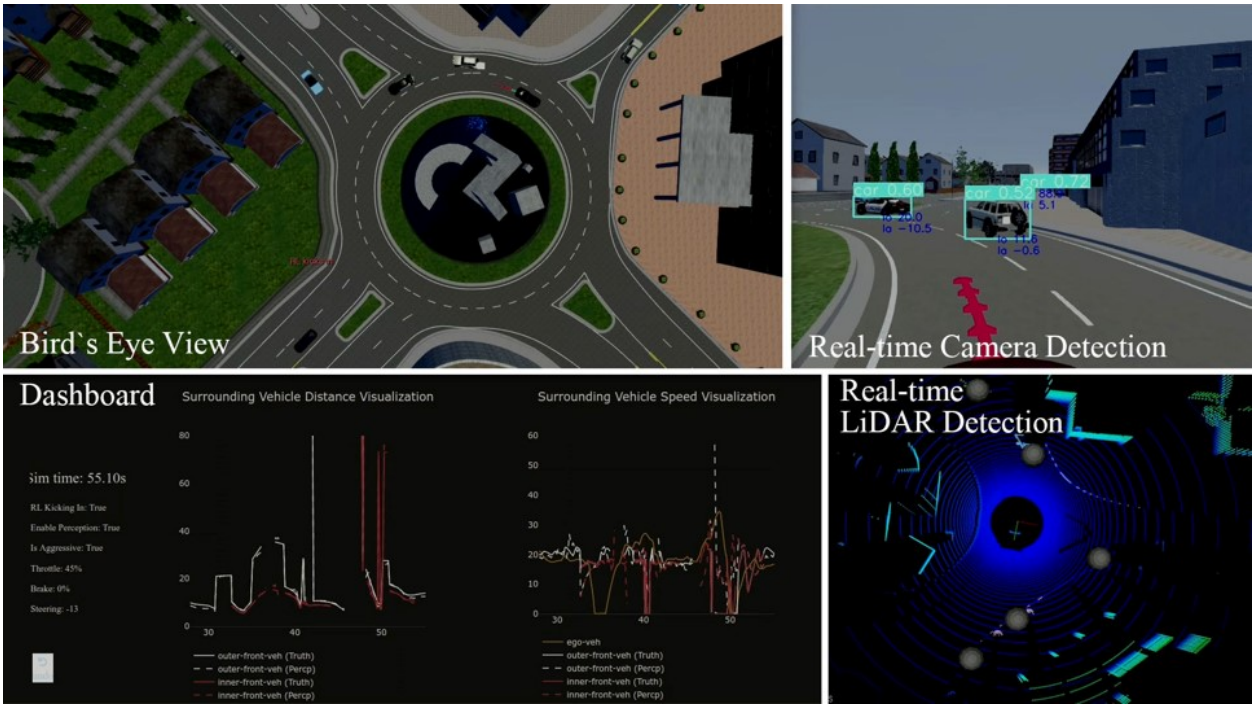


Figure 4.1: Example of visualization interface of CLAP Platform.

platform also provides utilities like a visualization server (Figure 4.1), calibration tools [276], and broadcasters.

The overall design goals of the platform are:

- Modular and flexible for easy development of algorithms
- Researcher friendly with decoupled components and great readability.
- Suitable for machine learning models and strategies through a uniform data structure.
- Compatible with both simulations (e.g. Carla [66]) and hardware. (e.g. Mcity test vehicles [65])

A desired attribute of the platform is "cloud-compatible" and "learning-compatible". The platform can run in cloud containers (e.g. Docker [165]) and it's equipped with V2X interfaces, which makes it "cloud-compatible". And the data structure designed for various machine learning algorithms makes the platform "learning-compatible".

The remainder of this section is organized as follows: we will first review the literature om Section 4.1.1, then Section 4.1.2 will discuss the platform architecture, Section 4.1.3 presents examples of our research to demonstrate the cloud and learning capabilities of CLAP.

(a) The test vehicle from the University of Michigan

(b) The test vehicle from Tsinghua University

Figure 4.2: The test vehicles from the University of Michigan and Tsinghua University with the CLAP platform deployed.

4.1.1 Related work

There exist several middlewares and autonomous driving platforms from both academia and industry. A well-known one is the ROS[198], widely used by robotics and AV researchers. ROS provides essential tools to help each component to communicate with each other efficiently using TCP/UDP interfaces. Its successor, ROS2, provides many improvements such as better data distribution and Python3 support. AUTOSAR[75] is another middleware for automotive manufacturers, which provides standardized hardware abstraction of ECUs and communication interfaces in vehicles. A new component of AUTOSAR, called Adaptive Platform, was released in early 2017 which includes support for cross-domain computation platforms and remote distributed services. The AUTOSAR architecture focuses on the development of onboard devices rather than being an operating system. There are private middlewares implemented by companies such as Tesla autopilot [112] and universities [27, 10], which we are not going to compare in this section since relatively little is known in the open literature.

Platforms specifically for automated vehicles have also been developed, e.g., Apollo and Autoware. Apollo[156] is built by Baidu, consisting of software, hardware and, a cloud platform. Apollo published version 5.0 in September 2019 and claimed to achieve geo-fenced automated driving. Autoware[119] is an open platform providing an all-in-one solution for AV. Autoware.ai is based on ROS and its successor Autoware.auto is based on ROS2. Autoware includes high-precision localization algorithms and vector-based HD Map, together with other essential perception, planning, and control algorithms. Since Autoware and Apollo are developed and maintained by commercial companies, they focus more on real vehicles and explore less about state-of-the-art machine learning algorithms. Our platform, named CLAP, is developed for both being used on a

[3]The functionalities of other platforms are based on their official documentation. A white dot stands for the functionality currently missing in the documentation. It doesn't mean that it cannot be implemented using that platform.

Table 4.1: Comparison of existing AV platforms.[3]

Features	**Apollo**	**Autoware**	**CLAP**
Modular	✓	✓	✓
Running in cloud	✓	✓	✓
Hardware adapters	✓	✓	✓
ROS-based		✓	✓
Standard HD map support	✓		✓
Easy customization		✓	✓
Generalized environment model for planning			✓
Dataset as input			✓
Compact and unified message definition			✓
Reinforcement Learning interface			✓
Online learning capability			✓
Fully open-sourced		✓	✓
Major language	C++	C++	Python

test vehicle and linked with simulation software. In addition, we make sure it is well suited for working with machine-learning modules. Table 4.1 shows the comparison between our platform and other popular platforms.

4.1.2 System Architecture

CLAP was built on top of ROS, which makes it easy to take advantage of existing modules such as those from Autoware. CLAP is written mainly in Python and some critical parts are written in Cython [15] and C++ to provide native performance. The algorithm implementations and interfaces to ROS in the platform have been separated in the code, so the platform is portable to other modular middlewares such as LCM[106]. In the platform, multiple algorithms have already been implemented and are ready for use, even though some are very simple and can be vastly improved. Please refer to the code repository for details.

4.1.2.1 Building Blocks

The platform is divided into core subsystems and peripheral modules. The architecture is similar to what has been described in Section 1.1 with a few novel modules. The core subsystems play a key

role in the computation flow for AV functions including sensing, cognition, perception, navigation, and control. Peripheral modules provide various interfaces for datasets, simulators, and vehicle hardware, together with system monitoring, visualization tools, and other common utilities. The relationship between the modules is illustrated in Figure 4.3. Usually, each module has a single instance running at the same time, but there can also be multiple instances running as backups.

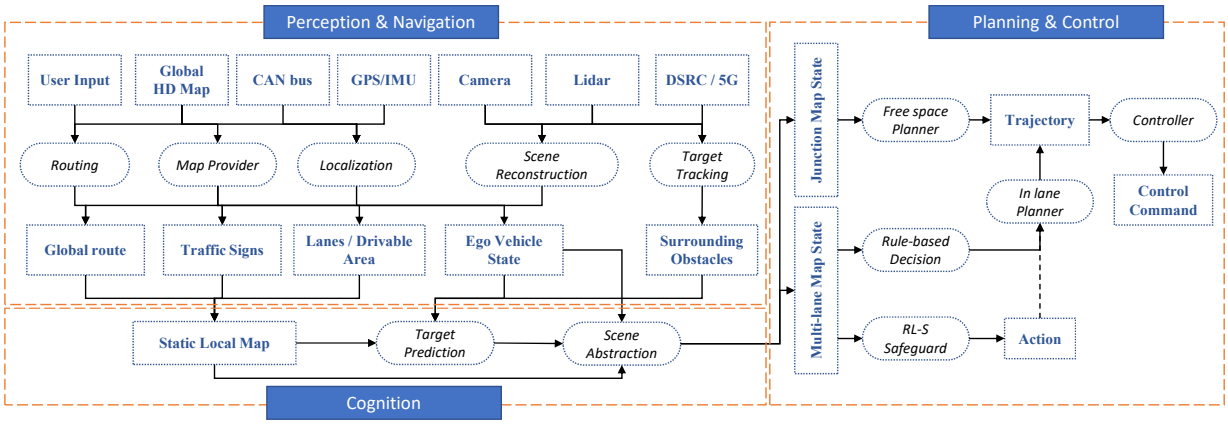


Figure 4.3: Structure of the CLAP platform. The connections in the figure only show the major data flow. In this platform, all the information can be passed from one module to another directly.

(1) Perception: The perception subsystem includes essential modules for vehicles to sense the surrounding. It converts multi-modal sensor data from the cameras, lidars, radars, etc. into a unified world model and abstracts the key information (e.g. obstacle moving speed, lane geometries, etc.) to establish a high-level understanding of the world. Data of other road users and infrastructures received from V2X can also be included. Two key aspects of perception are included in the platform: target tracking and scene reconstruction.

Target tracking incorporates algorithms to detect and track **dynamic** obstacles around the vehicle (e.g. vehicles, pedestrians, animals, etc.) The output of the module is the kinematic and geometric properties of the objects.

Scene reconstruction relates to image segmentation and point cloud segmentation, to build a structured representation of the **static** environment, such as road profile, drivable area [284] and lane properties. The output of the module is a parametric representation of the local map.

(2) Navigation: This subsystem provides information beyond the perception range of the onboard sensors and can cover three functions: localization, map provider, and routing.

Localization is an essential function to request and obtain the proper subset of global information. With precise localization, the surrounding environment can be obtained from map services or vehicular communication, which helps to decrease the computational load of the AV. Localization performance can be improved with an HD Map input, either in dense point cloud or in vectorized

format, using algorithms such as iterative closest point[18] and normal distributions transform[20].

Map provider offers a structured local map based on the location of the ego vehicle. The local map contains information similar to what the scene reconstruction module could create, but is more robust and does not require line-of-sight. This module can work with customized map data structures or using existing map tools like SUMO[133].

Routing The routing module selects a proper route for given origin-destination pairs. The route may be optimized for different metrics: travel time, distance, or energy consumption. In our platform, the routing can use dynamic information from connected vehicles and infrastructures for continuous updates. The key difference between the routing and the planning modules is that routing assigns a path beyond the vehicle's perception range and it's not updated frequently.

(3) Cognition: We group the modules that comprehend the surrounding environment and connect perception and navigation with planning, into a subsystem named cognition. It bridges data from perception and navigation modules into a dynamic local map that contains all information needed by the planning module. Its main functionality includes target prediction and scene abstraction.

Target prediction is the continuation of target tracking, which predicts the maneuver of surrounding objects based on past trajectory and object signals (e.g. turn signals, hand gesture). The interaction between road users will also be taken into account.

Scene abstraction refers to the process that local map elements from navigation and perception subsystems are further abstracted. An example is to associate detected vehicles and traffic lights with lanes. This module is an important part of our platform that distinguishes itself from others, and provides a unified interface for decision agents under different scenarios. The details will be discussed in the next section.

(4) Planning: The planning module processes all the environment information from upstream modules and generates a proper plan for the vehicle to follow. The planning module is split into two sub-modules: decision and trajectory generation.

Decision is also known as maneuver planning or mission planning. This module makes different decisions for different scenarios. For instance, the module will decide whether and when to change lanes. The decision module will create a proper task for the trajectory generator.

Trajectory generator creates a path and trajectory for the vehicle to follow in the next few seconds. The trajectory should be selected considering the following constraints: collision-free, comply with traffic rules and regulations, and stay within the dynamic limit of the vehicle.

(5) Control: The last subsystem is control. Control commands for throttle, braking, and steering should be generated at a high frequency so that the vehicle could respond to sudden events quickly. A high-fidelity dynamic model is usually used. The control module also controls auxiliary devices such as turn signals and windshield wipers.

(6) Peripheral: The peripheral module provides some other utilities, e.g. adapters to use datasets as sensor input. These modules ensure interoperability with both simulations and physical hardware. The platform also includes a library for commonly used kinematics and dynamics computation.

4.1.2.2 Environment Model

The interface connects the modules and the sub-systems within a module. In this subsection, the interface in the cognition module will be covered, to demonstrate how this platform model the environment and tasks. The key model of the data flow is a representation of the dynamic local environment called **map state**, which includes two map categories (see Figure 4.4) for junction case and multi-lane case, respectively. The map state is designed to provide a uniform input for the planning algorithm (e.g. reinforcement learning models).

Road Obstacle The objects around the host vehicle are all treated as road obstacles. Each obstacle contains essential information: (1) unique id, (2) dynamic states with an assigned confidence level, and (3) physical boundary. Specifically, the dynamic states are represented by pose, velocity, and acceleration in the local Cartesian coordinate and also in the Frenet coordinate along with the centerline of the current lane or reference path. The formal representation of the dynamic state S is described below.

$$S = (\mathbf{x}_C, \mathbf{v}_C, \mathbf{a}_C, \mathbf{s}_F, \mathbf{v}_F, \mathbf{a}_F, \theta) \tag{4.1}$$

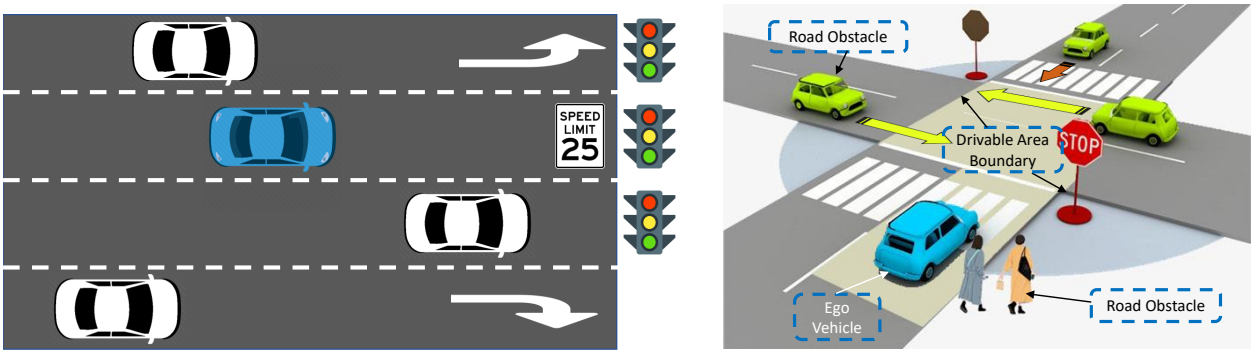
where each component is a random variable represented by its mean and covariance. The variables with subscript 'C' are in the Cartesian coordinate and those with 'F' are in the Frenet coordinate. The Frenet coordinate is suitable for generalized algorithms for driving along a lane.

The data structure also contains auxiliary information θ for planning in multi-lane scenarios, including current lane index, lane behavior (e.g. changing to left lane), and priority. For instance, at an intersection, vehicles in lanes that have Right of Way (RoW) will be assigned a higher priority than other vehicles.

Lane State is an abstraction of a predefined path on the road. The lane could be an area defined by paved lane markers, or a virtual area surrounding the reference path. The data structure contains both static and dynamic information. The geometric properties and traffic regulations of the lane are often static, while the obstacles in the lane and sudden changes of the lane's nature (e.g. lane blockage, construction) are dynamic. The underlying representation of the static part of a lane is a list of multiple lane points. Each lane point is a geometry point as well as an edge point where the speed limit or lane marker type changes.

Map State For scenarios on well-structured roads with clear lane markings, we build a map

structure called multi-lane map states (Figure 4.4a). This data structure contains multiple lane states and essential information about the target lanes. For scenarios on unstructured areas such as intersections, highway merge, and construction sections, we use a map structure named junction map states (Figure 4.4b), which contains the reference path presented as a lane state, and a drivable area which provides a constraint for the planner. When driving on the road, a junction map is always generated while the multi-lane map is only generated when the lanes are well defined and detected, and when no unusual event is happening (e.g. road blocked by construction, responding to emergency vehicles, etc).



(a) Multi-lane map state

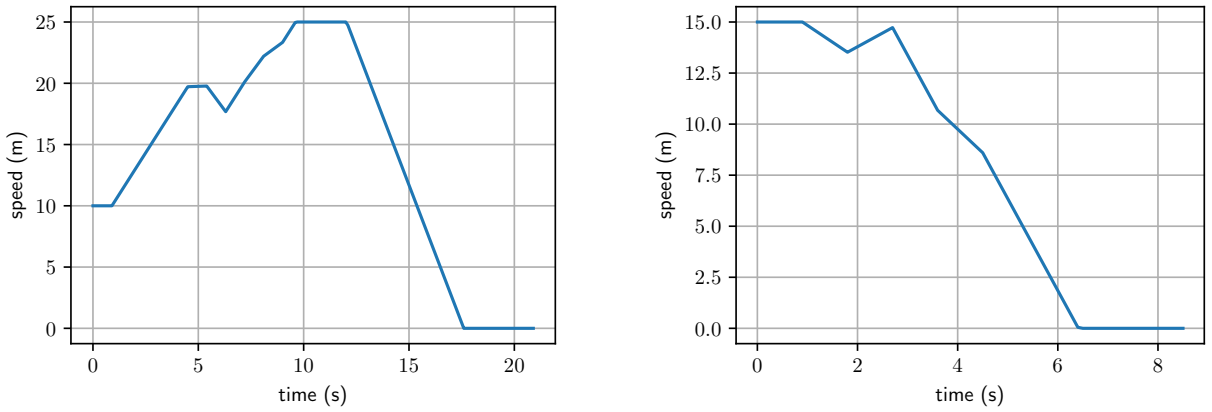
(b) Junction map state

Figure 4.4: The schematic diagram of the two map states.

By defining this data structure, reinforcement learning models can be effectively trained and the trained model can be used in similar scenarios. For example, the model [37] trained in a highway environment is successfully deployed into the roundabout scenario, the agent can navigate into and out of the roundabout safely just like driving on and off the highway.

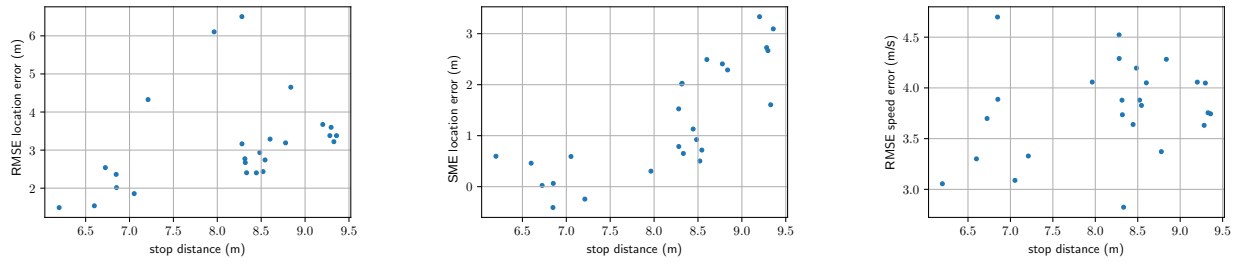
4.1.3 Use Cases

In this section, we present two examples of how this platform was used to implement and test AV algorithms. To highlight the cloud-compatibility of the platform, we tested a fully automated agent under an emergent braking scenario in the dockerized simulator and ran multiple instances at the same time in our server. To demonstrate learning-compatibility and the possibility to migrate from the simulation to real-world hardware, we trained an RL agent in simulation and deployed the agent on an experimental vehicle. The details of the deployed AV components can be found in Appendix A.



(a) $v_0 = 10m/s$ (b) $v_0 = 15m/s$

Figure 4.5: Speed profile of the front vehicle with different initial speeds.



(a) RMSE of position, $r = 0.40$ (b) SME of position, $r = 0.79$ (c) RMSE of speed, $r = 0.28$

Figure 4.6: The relationship between different error criteria and final stopping distance. The stopping distance using the true position of the front vehicle is 7.76 meters. r is the correlation coefficient.

4.1.3.1 Effect of perception error on control error using parallel simulations

In this example, we designed an emergency braking scenario to test the influence of perception error on the final vehicle distance error. Two consecutive vehicles were driving on a straight road where the ego car behind is controlled by our platform while the lead vehicle follows a designed braking profile [259]. The two cars first were driving at the same speed at a steady distance between them, and then the front vehicle braked suddenly to challenge the rear vehicle.

The ego vehicle is controlled by the CLAP platform using YOLOv3 [201] and Inverse Perspective Mapping as perception backbone, multi-lane map state with a single predefined lane as planning input, and the IDM[124] and a simple Proportional–Integral–Derivative (PID) controller as planner and controller. The parameters of IDM and PID are fixed during the experiments. Multi-

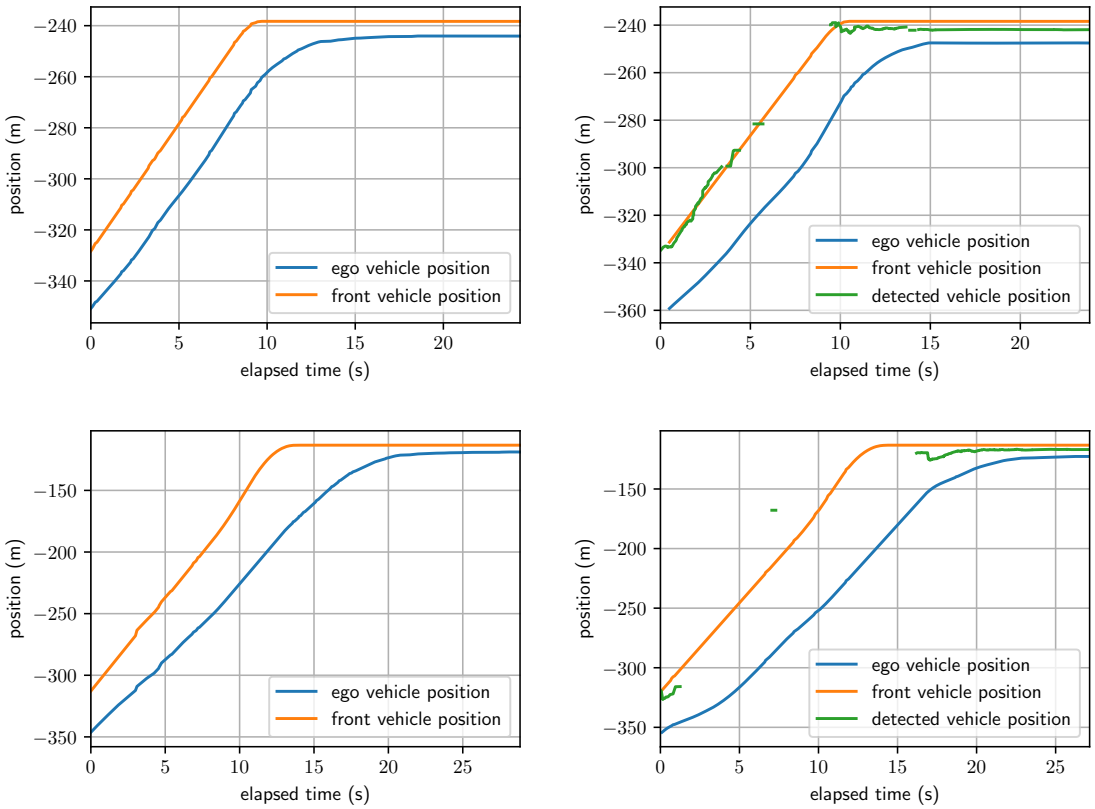


Figure 4.7: Simulation results of an emergency stopping scenario case. (a)(c) use the true position from the simulator while (b)(d) use the perceived position with error. Initial speeds of (a)(b) are 10m/s while initial speeds of (c)(d) are 15m/s

ple runs were simulated using Carla simulator [66], which generates high-fidelity camera data and simulates the vehicle dynamics. The experiment is carried out in "Town" 04 of Carla on a rainy day, and the whole test process is done in parallel with Docker by executing multiple containers with different profile settings, where each container has a simulator and an agent instance.

We compared the braking behavior of the ego vehicle using either simulation-reported location or sensor-perceived location of the front vehicle during the planning stage. Figure 4.5 shows the designed speed profile and Figure 4.6 shows the error distribution of multiple runs, which will be discussed later. From the results from simulation (as shown in Figure 4.6), perception error will usually lead to a smaller stop distance which is more dangerous. In the case described in Figure 4.7, however, the perception error results in a larger final distance between the ego vehicle and the front vehicle, because the perception error leads to a conservative detection result. The final result is highly related to detection performance during the braking period.

Therefore, to quantify the correlation between perception error and final vehicle performance, we created a signed error criterion for this scenario and we compared the correlation of these errors

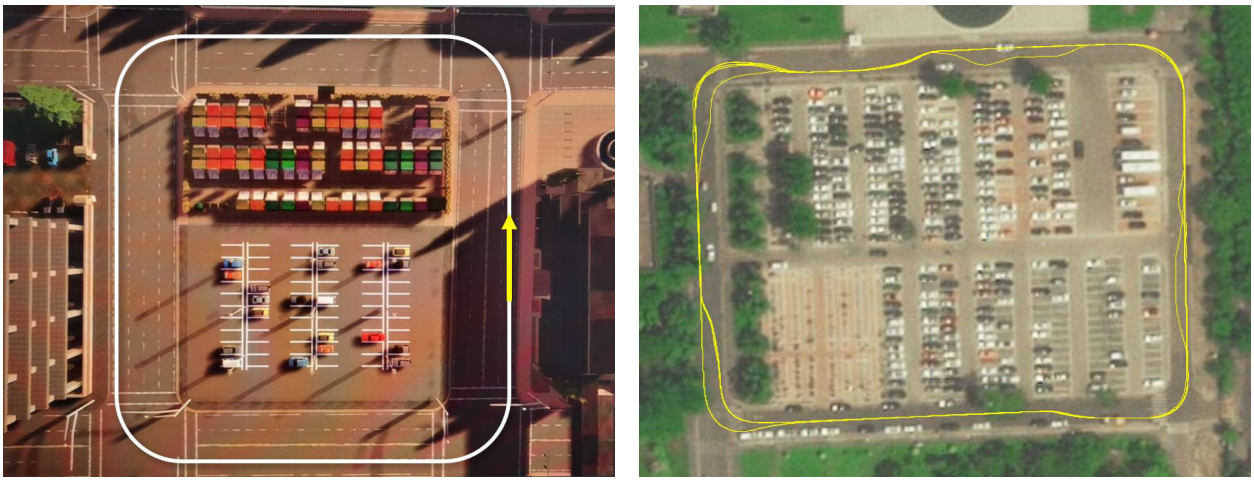
with regards to the final stop distance. The definitions of the errors are listed below:

- Root Mean Squared Error (RMSE) of position: $e_1 = \sqrt{\frac{1}{n}\sum_t (x_t - \hat{x}_t)^2}$
- Signed Mean Error (SME) of position: $e_2 = \frac{1}{n}\sum_t (x_t - \hat{x}_t)$
- RMSE of speed: $e_3 = \sqrt{\frac{1}{n}\sum_t (v_t - \hat{v}_t)^2}$

Here x, v are the position and velocity of the front vehicle in the longitudinal direction, and the variables with a hat are the perceived values. The comparison results are plotted in Figure 4.6. From 24 simulation runs, we can find that the signed error shows a higher correlation with the final stopping distance. And perception error of speed plays a less important role during the process.

This use case example shows the ability of our platform to study vehicle performance, in addition to traditional machine learning performance criteria (mAP, recall ratio, etc.). Besides, by using simulation software, the effect of weather, lighting, and various other factors can be studied quickly and efficiently. For example, in the emergency braking case, the research may focus on improving the perception module or the control module. We also want to note that, the same scenario can be repeated on real test vehicles for a faster development cycle.

4.1.3.2 Deployment of reinforcement learning agent with map states



(a) Test scenario in the Carla simulator

(b) Test scenario in the real-world

Figure 4.8: Test routes for the reinforcement learning agent. The yellow line in (b) is the actual driving path of the agent.

In the second example, we developed an AV agent based on reinforcement learning using our unified map state interface and test it in real-world driving. The agent is trained in Carla simulations

to drive around a block and the route contains both road segments and intersections. The designated routes in the simulator and real-world can be found in Figure 4.8. Then it's deployed in a vehicle for open-road testing. The test vehicle (see Figure 4.2) is an Xpeng G1 electric vehicle, equipped with lidars, GPS, and IMU. In the test, the agent used object detection and tracking results from the perception module of the platform. The agent was able to navigate through this environment safely and smoothly with several surrounding vehicles and pedestrians. The actual driving path is shown in Figure 4.8b with several lane changes accomplished during the test. The details of the RL-S Safeguard algorithm used here can be found in [36].

This example shows that the platform can be used for a seamless transition between simulations and real-world experiments.

4.2 System Validation in Simulated Scenarios

The proposed occlusion-aware perception and planning algorithms can be evaluated on top of the previously discussed AV stack. The interconnection between the involved modules is depicted in Figure 4.9. For the purpose of focusing on the occlusion-aware techniques presented in this dissertation and to ensure the integrity of the validation results, certain modules such as localization have been omitted, as compared to the comprehensive AV system introduced in Section 4.1.2.

This architecture establishes the connectivity among all the modules proposed in Chapter 2 and Chapter 3, defining their interoperation. To demonstrate the potential of deploying a system based on this architecture in a real vehicle, its implementation will be carried out on the ROS platform and tested in the Carla simulator [66] within this section.

It is important to note that, due to the substantial computational requirements of the neural networks involved, the (joint) detection and segmentation module is excluded during testing. Instead, the simulator provides ground truth semantic labels and object locations. Nevertheless, the simulation includes the aspect of occlusion during the object detection process through a custom ROS node. This node selectively excludes objects that are not fully observable to the ego vehicle from the list of detected objects, which is then fed to the downstream modules.

4.2.1 Testing Scenarios

The testing of the system is conducted on the Carla simulator [66], which is a high-fidelity simulator designed for ADSs. It is equipped with high-quality sensor simulation, vehicle dynamics solvers, walker bone controllers, traffic manager, and traffic light simulation, which provides a comprehensive test-bed for ADS. There are more than 10 built-in maps in the simulator and we choose the map "Town 05" for the testing because it comprises a rich set of driving environments

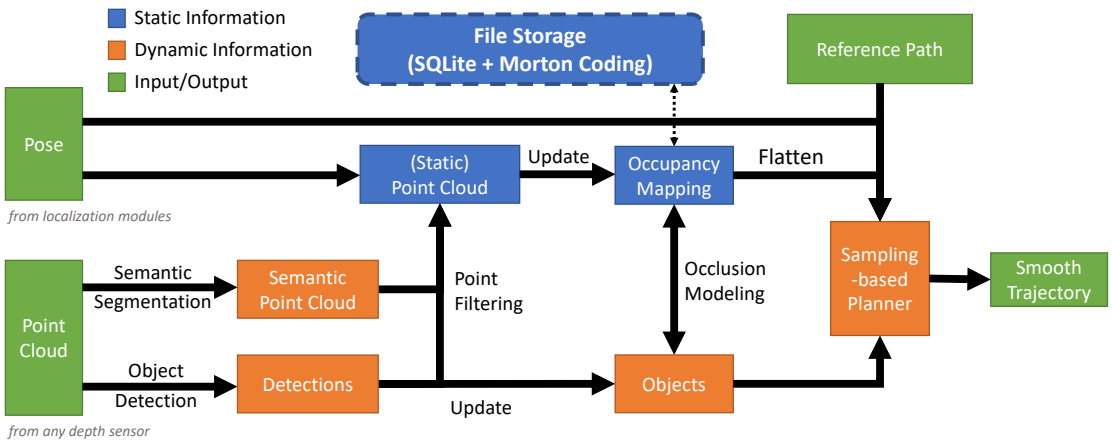


Figure 4.9: Architecture of the ADS to be tested. The necessary input for the system is the pose of the ego vehicle and the lidar point cloud. The point cloud is used to generate the static map and detect the dynamic objects.

including urban buildings, parking lots, residential areas, and viaducts.

To make it easier to test the ego vehicle controlled by the ADS, a loop route is designed in the north part of the map, as illustrated in Figure 4.10. Along the route, there are four scenarios designed analog to those in Section 3.2.4. The details of the scenarios used for testing are shown in Figure 4.11.

In the parking lot scenario, when the ego vehicle is driving between the parked vehicles, there will be two pedestrians appearing from the left and right of the ego vehicle. Both pedestrians are not visible in the beginning due to the occlusion by the parked cars along the route. In the bus station scenario, there is a parked vehicle on the sidewalk, blocking the ego vehicle so that the pedestrian behind the bus stop will not be visible until it suddenly walks out from the bus stop. In the intersection scenario, there will be a vehicle driving from the right-hand side of the ego vehicle, running a red light. Due to the walls next to the road, the ego vehicle will not detect the other vehicle until it nearly enters the intersection area. In the parked bus scenario, there will be two buses parking near the bus stop, and a pedestrian will suddenly appear in front of the leading bus. The pedestrians are initially invisible due to the occlusion caused by the two buses. In all of the above scenarios, if the ego vehicle starts off at the specified speed and does not slow down, it will always collide with other road users. During the testing, the ORUs are assumed to be not

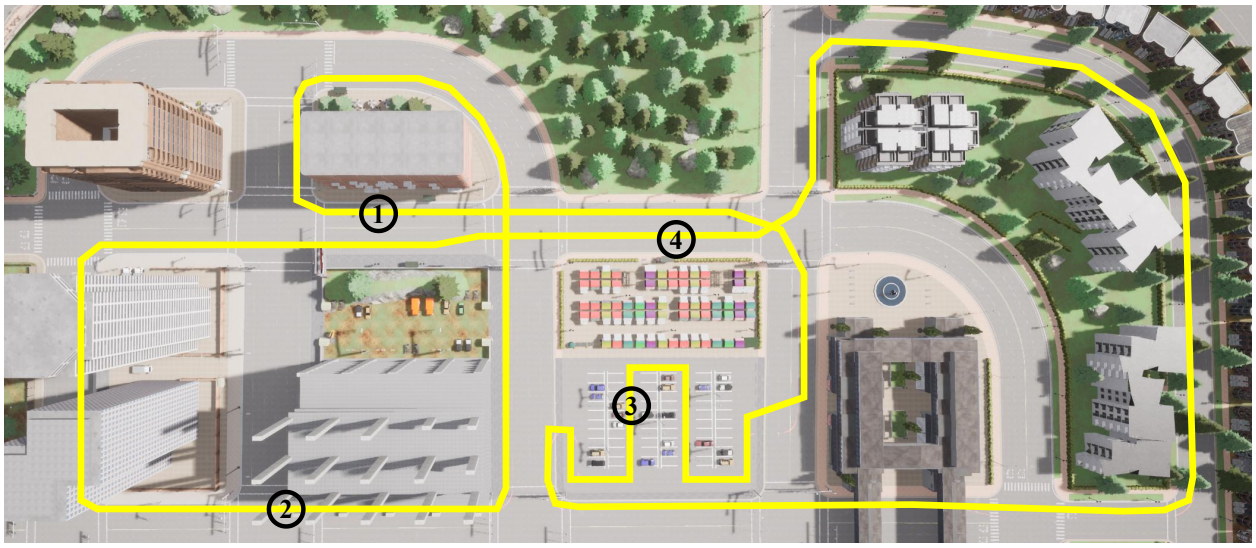


Figure 4.10: The testing loop and the location of the testing scenarios in the map "Town 05". The testing scenarios are: (1) Parking Lot (2) Bus station (3) Intersection (4) Parked Bus.

reactive, they will follow the designed route at a constant speed.

Like the scenarios designed in Section 3.2.4, these scenarios are a good representation of the common risks caused by occlusions in normal driving, and even human drivers would need an amount of experience to navigate them (relatively) safely.

4.2.2 Implementation Details

In this section, the implementation details of the ADS architecture and the testing environment will be introduced. Please refer to Figure 4.9 for an overview of the architecture.

There are two key ROS nodes implementing the mapping and planning algorithms. A mapping node is implemented to handle the point cloud filtering and the occupancy map update. The mapping algorithm relies on the Eigen and OpenMP library to perform point search and kernel calculation efficiently. The output of the mapping modules is a 2D grid map stored in the data structure provided by the GridMap ROS package [72]. Please refer to Section 2.3.3 for the layers in this grid map. Subsequently, a planning node is implemented to maintain a list of both tracked objects and phantoms, and generate an optimized trajectory, based on the grid map it receives from the mapping module. It depends on Eigen and a customized trajectory manipulation library. The output of the planning module is a trajectory of a short time horizon. Lastly, a low-level controller node is also implemented to convert the trajectory to vehicle commands, including throttle, steering, and brake values. The pose, point cloud, and object list are provided by the official Carla-ROS bridge, and the vehicle commands are also sent to the bridge for execution.

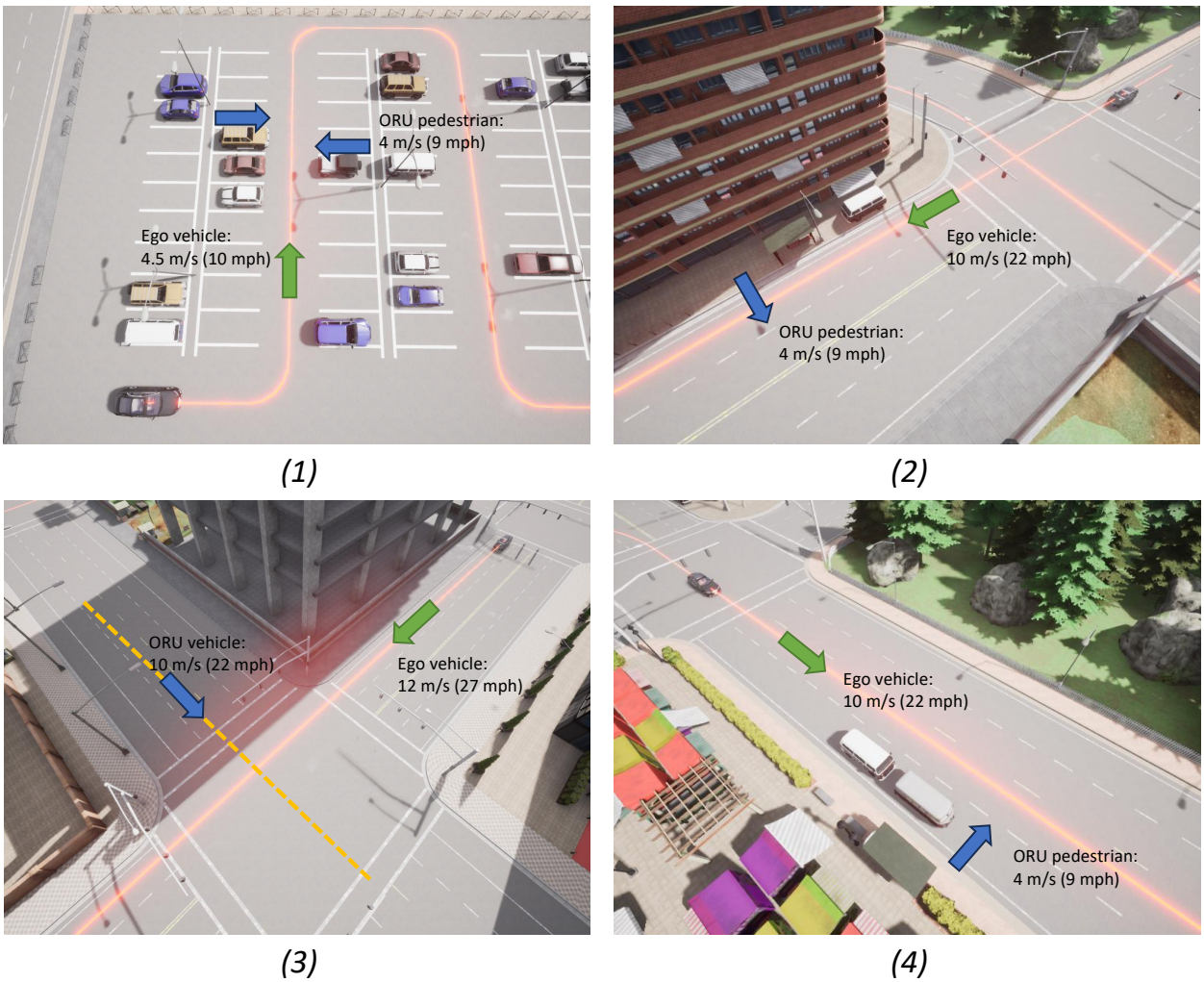


Figure 4.11: Definitions of the scenario. The testing scenarios are: (1) Parking Lot (2) Bus station (3) Intersection (4) Parked Bus. In these pictures, the green arrows denote the driving direction of the ego vehicle under test, and the blue arrows denote the driving/walking direction of the ORU after being triggered.

An important feature of the proposed ADS architecture is that the occupancy map will be loaded and dumped from a file storage automatically. This capability enables the AV to memorize the environment it visited and helps the identification of the occluded area. The occupancy map is serialized using the MessagePack library and stored in an SQLite database. Each block of the occupancy map will be indexed by Morton Code. The spatial property of the Morton code helps to find the blocks nearby the ego vehicle. During the serialization, the semantic probabilities of each grid will be compressed by only storing the probability of the highest category, the probability of the free category, and the average probabilities of the remaining categories.

To automate the testing in Carla, a loop route broadcaster and a scenario trigger are imple-

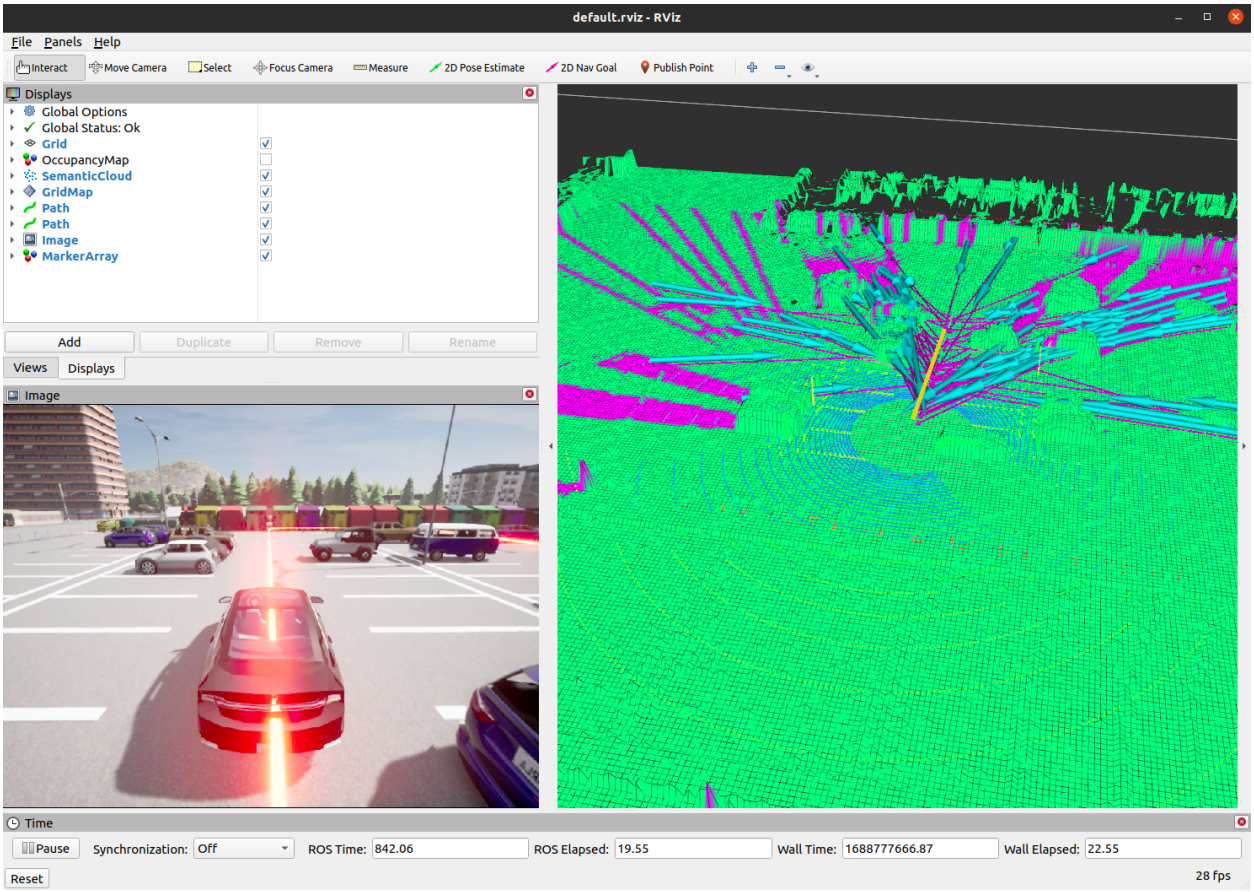


Figure 4.12: A screenshot of the ADS running with the Carla simulator in RViz. The left bottom view displays the image from a virtual camera placed behind the vehicle, and the right area displays the occupancy grid map and the planning information. In the right area, the purple grids represent the occluded grids and the green grids are non-occluded grids (including occupied grids). The yellow line in the middle denotes the current planned trajectory and the starting points of the cyan arrows denote the pose of the phantoms. The length of the arrows is proportional to the speed of the phantoms.

mented. The route broadcaster will publish part of the reference path near the ego vehicle and the scenario trigger will initialize and control the ORUs when the ego vehicle reached a preset location. The scenario trigger will also reset the environment when the ego vehicle passed another preset location. Similar to Section 3.2.4, the scenario trigger will add random noise to the trigger location and the initial speed of the ORUs, so that the behavior of the ego vehicle can be better evaluated.

Since the list of objects is directly provided by the Carla simulator, the occlusion has to be simulated when the ADS is tested, which will be implemented as a separate node. To achieve this, 7 key points of the ORU are used as representations for visibility testing, and if 5 out of 7 key points are not visible from the sensor of the ego vehicle, the ORU will be considered as occluded and not

included in the list of objects provided to the planning node. The key points are the center of an ORU and the midpoint of the six lines connecting the center of the ORU and the center of the six faces of the bounding box. In this way, the occlusion can be simulated efficiently and accurately.

The experiments will be conducted on ROS noetic with Ubuntu 20.04 operating system and Carla 0.9.13. The Figure 4.12 shows the visualization of the ADS during the experiments. Due to lacking access to complete ADS that can be easily deployed, the proposed ADS architecture without occlusion-aware cost terms in the planner will be used as the baseline in the experiment. Additionally, a publicly available AV stack, the Autoware [119, 120], will be tested for comparison as well. The Autoware.Universe framework based on the ROS2 Galatic will be used in the experiments.

4.2.3 Testing Results

Table 4.2: Quantitative results of the testing in Carla simulator

Scenario	Algorithm	Crash Rate	Average Minimum Distance
Intersection	Autoware [119]	70%	0.981m
	Baseline	90%	0.013m
	Ours	**60%**	**1.258m**
Bus Station	Autoware [119]	40%	0.852m
	Baseline	40%	0.805m
	Ours	**0%**	**3.172m**
Parked Bus	Autoware [119]	50%	0.735m
	Baseline	40%	0.369m
	Ours	**20%**	**1.441m**
Parking Lot	Autoware [119]	-	-
	Baseline	50%	0.255m
	Ours	**10%**	**1.293m**

The testing results are summarized in the table 4.2. It can be found that in all the scenarios, the proposed algorithm achieves lower crash rates and higher average minimum distances by a large margin compared with the baseline ADS without occlusion awareness. Among all the scenarios, the intersection is the most challenging one because the ORU is running the red light in the middle lane of the road, therefore the ORU is visible from the ego vehicle later than the intersection scenario tested in Section 3.2.4. However, the proposed ADS system still reduces the number of crashes and kept a higher safety margin from the ORU vehicle in the intersection scenario. These results validate the ability of the proposed algorithm to improve the safety of an ADS.

When comparing with the performance of the Autoware, the proposed planner similarly achieves better safety. The experiment data for the parking lot scenario is unavailable for Au-

toware because it requires lane maps for planning. Therefore Autoware cannot drive in the parking lot because there are no lanes in that area. It's worth noting that, the longitudinal planner of Autoware is not well tuned for high speed scenario, therefore the speed of the ego vehicle oscillates in a range of ± 2 km/h, and the average speed of the vehicle is smaller than the target speed in the intersection scenario, which partially contributes to the lower crash rate compared with the baseline approach.

4.3 System Validation in Real-world Scenarios

Aside from the validation in the simulated scenarios, we also tested the effectiveness of the proposed system with real-world scenarios. Specifically, we designed a test scenario inside the Mcity testing facility to evaluate the performance of the occlusion-aware perception and planning algorithms.

The experiments are conducted with the vehicle platform displayed in Figure 4.2a, which is also used by [278]. The vehicle is equipped with an RTK 3003 module from Oxford Technical Solutions and an Inertial Measurement Unit. These sensors report the precise pose of the testing vehicle. It's also equipped with a by-wire controller that read commands from ROS topic, and a Dedicated Short-Range Communications (DSRC) module which reports the positions of the other vehicles in the facility through V2X communication.

In the real-world experiments, we focus on testing the object management system proposed in Section 3.1, and we integrate the phantoms management into the existing planner deployed on the vehicle (developed by [279]) by injecting the phantoms into the list of objects. In order to remove other uncertainties in the system and focus on the effectiveness of the phantom-based management system, we utilize the positions reported by DSRC as the perception results. To simulate the occlusion in the perception system, we added a ROS module that removes the other vehicles which are not visible from the position of the ego vehicle similar to the module discussed in the simulation experiments (see Section 4.2).

4.3.1 Testing Scenarios

The testing scenario is an intersection inside the urban area of the Mcity testing facility. The testing route is plotted in Figure 4.13. During the test, there will be a human-driving challenge vehicle coming from the right. If the ego vehicle does not respond properly, there could be a collision at the intersection. The route of the ego vehicle and the challenge vehicle are designed to be looping so that the tests can be conducted by multiple times easily.

The testing cases are listed in Table 4.3, where the case 0 is considered as the baseline case.

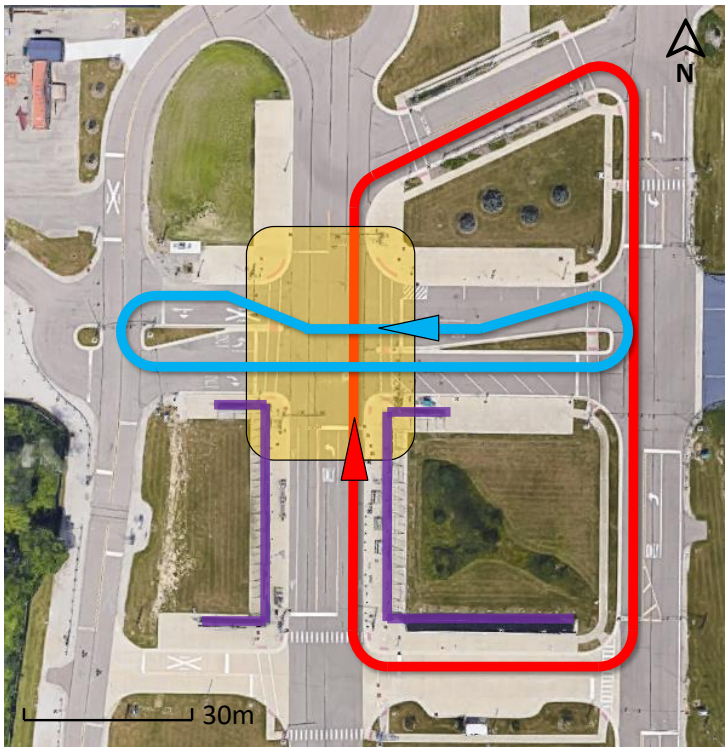


Figure 4.13: The test route for the object management system test. The blue line represents the route for the ego vehicle and the red line represents that of the challenge vehicle. There are several facade walls blocking the view of the ego vehicle, represented by the purple lines.

Table 4.3: Testing cases for the object management system test.

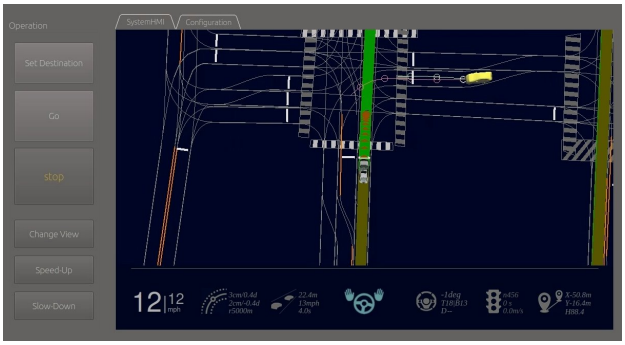
	Case 0	Case 1	Case 2	Case 3
Occluded Vehicle		✓		✓
Phantom Vehicle			✓	✓

The *occluded vehicle* row represents the existence of the challenge vehicle and the *phantom vehicle* row represents whether the proposed object management system with phantoms is enabled. It is worth noting that we limit the phantoms to be generated only on the lanes, as we know that the challenge vehicle will not drive outside the road area. This is also an example of how the phantom generation efficiency can be further improved with additional information.

4.3.2 Testing Results

The tests are conducted with fixed starting points for the ego vehicle and the challenge vehicle. The challenge vehicle always starts at the same time as the ego vehicle. However, due to the stochasticity of the human driver's behavior, the trajectories of the ego vehicle and the challenge

(a) Case 0: no phantoms, no challenge car.



(b) Case 1: no phantoms, with a challenge car.

(c) Case 2: with phantoms, no challenge car.

(d) Case 3: with phantoms, with a challenge car.

Figure 4.14: Screenshots of the trajectory planner under different testing scenarios. The gray vehicle at the bottom is the ego vehicle, the red vehicles are the generated phantoms and the yellow vehicle is the challenge vehicle.

vehicle are not the same across the test runs. Figure 4.14 illustrates the scenarios in action. The proposed phantom generation algorithm can successfully place phantom vehicles in the occluded area. Notice that there are speed gauges at the left bottom of the screenshots, which show that the ego vehicle starts to slow down earlier with the help of generated phantoms.

In Figure 4.15 where the screenshots at different timestamps are shown, the phantom elimination process has been demonstrated. There are phantoms on both sides of the road at the beginning, because there are walls on both sides of the road as shown in Figure 4.13. Later on, the left side became visible first, and then the right side became visible as well. Therefore, the phantoms to the left were cleared first and the phantoms to the right were cleared later. Finally, when the ego vehicle entered the intersection, all the branches of the intersection became visible and all phantoms are removed.

We also recorded the dynamic states of the ego vehicle during the tests, which are plotted in Figure 4.16. It can be found from Figure 4.16a that, with the proposed phantom generation approach, the ego vehicle will slow down even if there is no actual risk at the intersection. This is

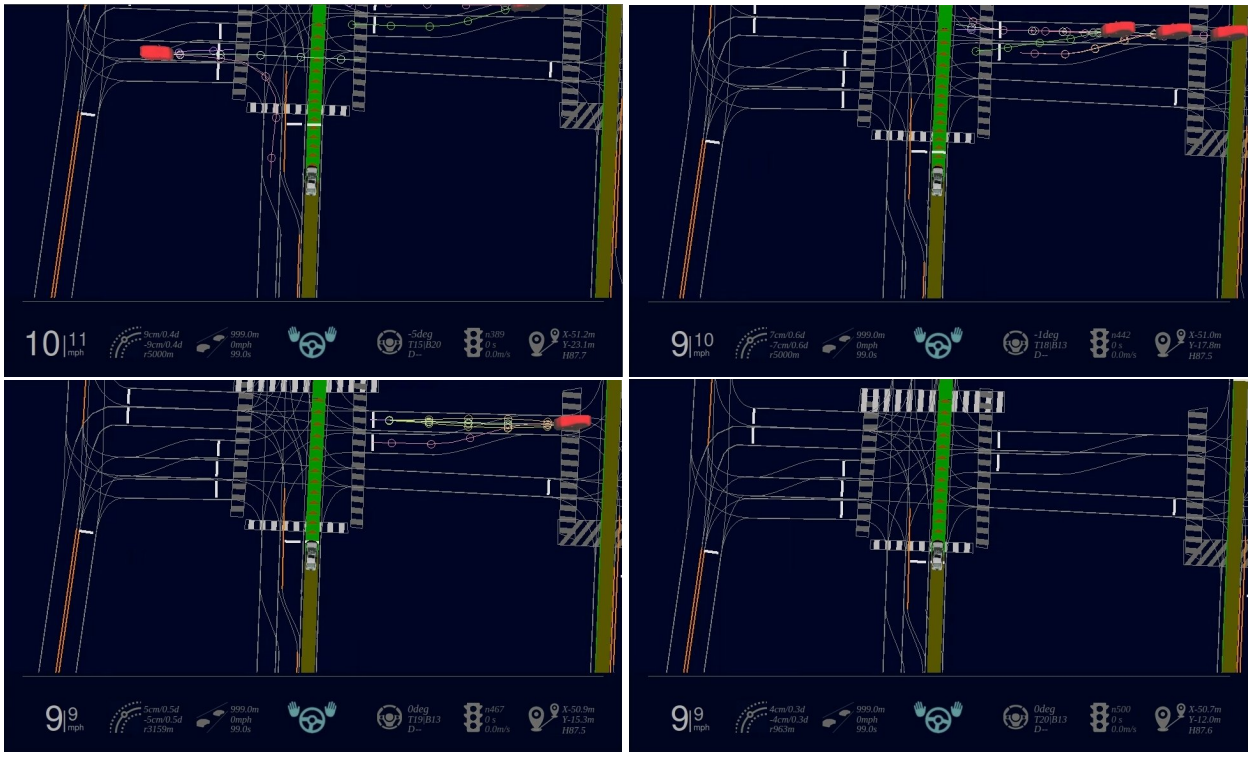


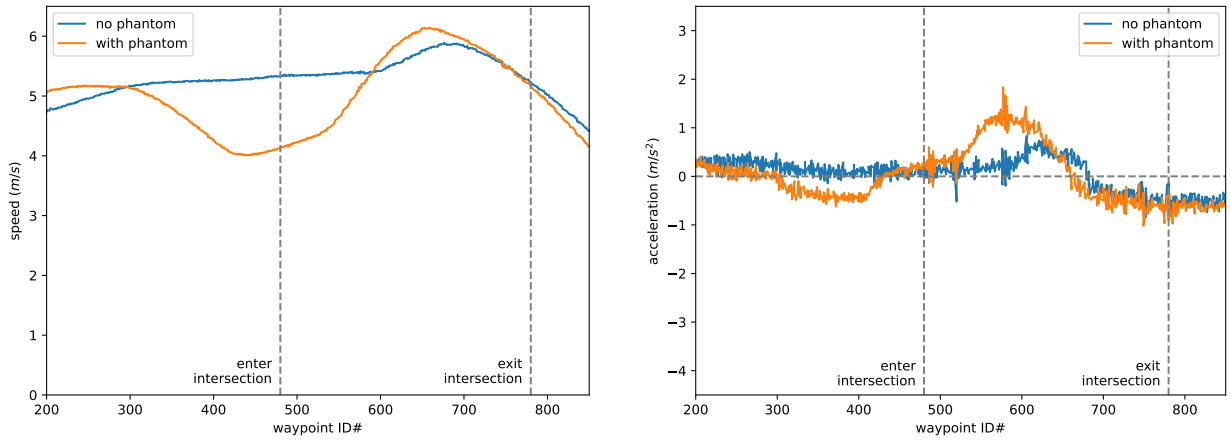
Figure 4.15: Screenshots showing the phantoms evolving in the test case 2. The temporal order of the screenshots are: left-top, right-top, left-bottom, and right-bottom.

reasonable defensive driving behavior, and slowing down is a necessary cost to pay in exchange for the safety improvement, which is demonstrated in Figure 4.16b, where the hard brake is avoided with the help of the phantom generation. The peak deceleration of the ego vehicle is around $1m/s^2$ with the phantoms compared with around $4m/s^2$ without the phantoms. According to the speed profile, the ego vehicle slowed down earlier when the phantoms exist during planning, which is consistent with the finding from Figure 4.14. A qualitative comparison of the safety is also shown in Figure 4.17. With the help of the generated phantoms, the distance was increased between the two vehicles when the challenge vehicle is at the front of the ego vehicle.

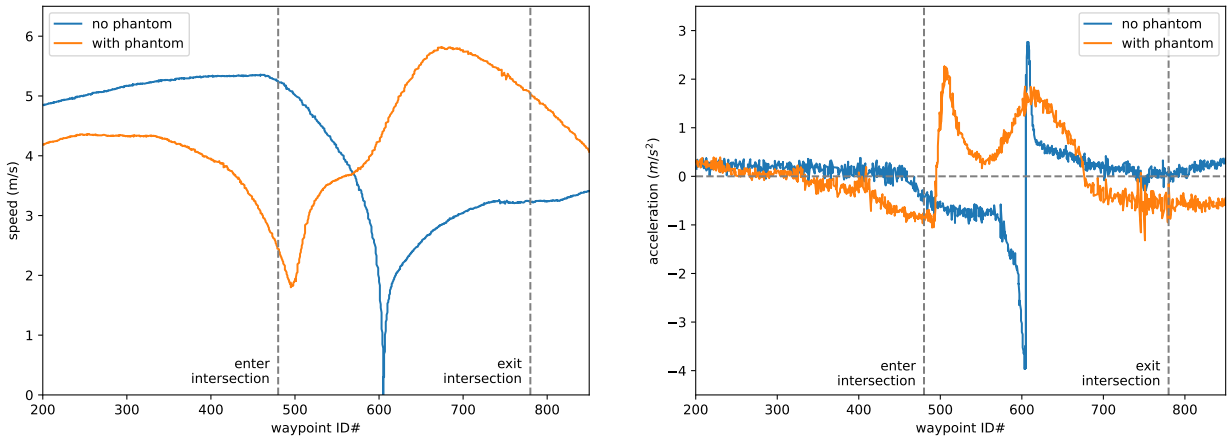
4.4 Summary

In this chapter, we first present an ADS framework as the foundation of our experiments. Then simulated and real-world experiments have been conducted to evaluate the efficacy of the proposed occlusion handling system. The key findings and conclusions of this chapter are as follows:

1. An open platform, designed to facilitate AV development, is introduced. This researcher-friendly platform comprises core subsystems and modules for essential autonomous driving



(a) The speed and acceleration profiles of case 0 and 2.

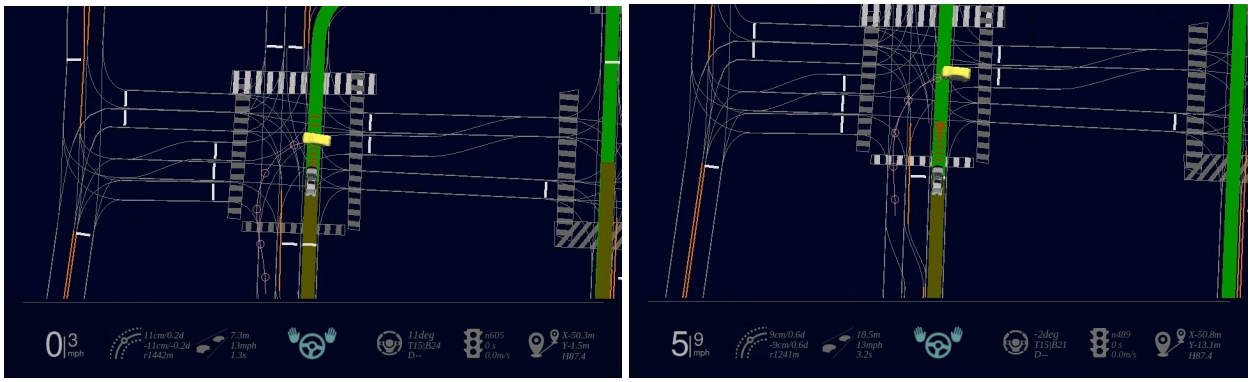


(b) The speed and acceleration profiles of case 1 and 3.

Figure 4.16: Quantitative results of the object management system tests with or without the challenge car.

tasks. It has been utilized in research conducted at the University of Michigan and Tsinghua University, offering state-of-the-art algorithms that enable researchers to easily compare new implementations with standardized input and output.

2. Simulation experiments using the Carla simulator are performed on the "Town05" map, featuring curated scenarios. The integration and testing of perception algorithms discussed in Chapter 2 and planning algorithms proposed in Chapter 3 are carried out. The performance of the vehicle under test, controlled by both the proposed and baseline ADS frameworks, reveals that the proposed solution for handling full occlusions on the road enhances driving safety compared to the baseline methods.



(a) Case 1 (without phantoms) (b) Case 3 (with phantoms)

Figure 4.17: Screenshots showing the distances between the ego vehicle and the challenge vehicle.

3. Real-world experiments are conducted at the Mcity testing facility, where a testing scenario involving an intersection with obstructed FoV is designed. By comparing the performance of the testing vehicle with and without the proposed occlusion handling methods, the effectiveness of the phantom generation and management framework is validated. The presence of phantoms enables the testing vehicle to exercise caution in the presence of occlusions and slow down before entering the intersection.

It is worth noting that the experiments conducted in this section have several limitations and can be improved in the following ways:

1. **Occlusion identification in new environments.** The current occlusion handling experiments requires prior environmental scanning, as it considers only informed occlusion for phantom generation. Although incorporating the entire occluded area would enhance the method's adaptability to new environments, it would significantly increase computational costs. A potential solution involves directly detecting occlusion in a 2D grid, integrate with coarse-grained navigational maps, and utilizing a learning-based algorithm for phantom generation. Nevertheless, the proposed solution in this dissertation remains consistent with the requirement of independence from HD maps stated at the beginning of the dissertation. The occlusion detection and phantom generation algorithm operate on a simple occupancy grid map, regardless of its resolution.

2. **Testing with perception error.** While it is feasible to deploy the system with a more comprehensive perception module (e.g., the joint detection and segmentation module presented in this Section 2.1), it was not tested in the simulation or real-world experiments due to computational optimization constraints. Future research work should explore the impact of perception errors on the final driving performance, as it would provide valuable insights.

3. **Testing in naturalistic driving environment.** The performance of the proposed ADS under normal driving conditions with naturalistic traffic was not evaluated, as it falls outside the scope of this dissertation. We do not anticipate the proposed ADS framework to outperform other frameworks in normal conditions. However, the proposed algorithm can be applied to other planners without compromising their normal driving capabilities, thanks to the design of cost terms and tuning parameters. Any optimization-based planner can be enhanced to handle occlusion by incorporating the proposed cost terms in Equation 3.6 and 3.7.

4. **Advanced design of the ORU behavior.** In both simulated and real-world experiments, the ORUs are passive and do not react to the ego vehicle. While this approach is acceptable for the scenarios described in this chapter, where the ORU is typically visible to the ego vehicle for only a short duration before a potential collision, incorporating reactive ORUs would be beneficial for other scenarios. The methodologies described in [258] and [260] can be employed to make the testing more closely resemble the experiences of human drivers in daily life.

These limitations do not undermine the effectiveness and generality of the proposed framework presented in this paper. However, conducting more comprehensive tests in the future will provide a better evaluation of the framework's capabilities and limitations.

CHAPTER 5

Conclusion

This dissertation presents a novel solution addressing the challenge faced by AVs when encountering full occlusions on the road. The framework comprises new perception algorithms for efficient identification of occluded areas without reliance on prebuilt HD maps, as well as new planning algorithms for generating safer trajectories in the presence of occlusions. The algorithms and frameworks proposed from Chapter 2 to Chapter 4 are summarized in the following section, and several future research directions are presented at the end.

5.1 Summary

Chapter 2 details the identification and representation of occluded areas using point cloud data input. A joint object detection and segmentation module generates object bounding boxes and point-wise semantic labels. A fast occupancy mapping module generates a semantic 3D occupancy map based on the semantic point cloud input. An occlusion modeling module then stores the occlusion status of the environment in a 2D occupancy map for downstream planner modules. Experiments on public datasets including nuScenes and SemanticKITTI demonstrate the improved capabilities of the proposed algorithm over state-of-the-art methods, with successful detection and visualization of occluded areas. The first question posed in Section 1.4 is answered in this Chapter.

Chapter 3 focuses on generating safe trajectories in the presence of fully occluded areas on the road. An object management system tracks objects from the upstream perception modules and generates predictions for the tracked objects. It also generates phantom objects to represent the potential behavior of occluded objects around the ego vehicle. A sampling-based trajectory planner produces near-optimal trajectories that balance safety, smoothness, and mobility. Specifically, two novel cost terms are proposed to quantify the risk caused by occlusion in the optimization. Experiments in a simple 2D simulator demonstrate the effectiveness of the novel cost design for trajectory optimization, with the proposed planner outperforming baselines in challenging scenarios with severe occlusions. The second question posed in Section 1.4 is answered in this Chapter.

Chapter 4 validated the performance of the proposed occlusion handling framework through simulated and real-world experiments. The simulated experiments are conducted in the high-fidelity Carla simulator, while the real-world experiments take place in the Mcity testing facility affiliated with the University of Michigan. In both sets of experiments, the proposed framework achieves a lower crash rate and larger safety margin compared to baseline methods. These findings indicate that the proposed perception and planning modules work together effectively to mitigate the risk of occlusions.

The novel occlusion-aware perception and planning framework proposed in this dissertation is a two-stage solution to improve the safety of the AVs. It first identifies the occlusions and then reacts to them using phantoms. The framework is not limited by the choice of algorithms for object detection, point cloud semantic segmentation, or motion planning. Any combination of perception and planning algorithms could benefit from the idea of using phantoms to represent the occluded road users, which is also reportedly adopted by automotive companies including Cruise and Tesla. Moreover, the two-stage framework can be generalized to other safety-critical robotic applications where collision avoidance with dynamic objects in occluded environments is crucial.

5.2 Future Directions

In this dissertation, we propose to introduce imaginary targets in the planning to offset the challenges posed by occlusions, which limit onboard sensing capabilities. While this approach is versatile and doesn't significantly burden the computational capabilities of existing ADS, it is not without its shortcomings. Apart from the limitations outlined in Section 3.3 and Section 4.4, there are several ways in which the proposed solution can be further refined:

Integration with other information sources

1. **Coarse-grained maps**: Leveraging coarse-grained or lower resolution maps could provide the ADS with broader contextual data about the environment. These maps might not offer intricate details but can provide generalized information about road layouts, potential occlusions, or traffic patterns, assisting the ADS in making more informed decisions when faced with occlusions.

2. **Social perception**: Humans inherently understand social cues and behaviors when navigating roads, be they as drivers or pedestrians. Integrating a system that can interpret such social signals, like gestures, eye contact, body language, or even trajectories of other vehicles [227], could immensely improve how the AV responds to unpredictable scenarios, especially

in occluded conditions. Complemented with the social perception capabilities, the AV could handle the scenarios with occlusions better.

3. **Roadside perception**: Incorporating data from roadside infrastructure, like traffic cameras or IoT-enabled devices, could provide an external viewpoint to AVs, aiding in the identification and understanding of occlusions and the road users in the occluded areas. This extra layer of perception, independent of the vehicle's onboard sensors, could enhance its overall awareness and responsiveness to dynamic road conditions.

Investigating the trade-off between safety and mobility

1. **User experience**: While ensuring safety is paramount, the overall passenger experience should not be compromised. An AV constantly taking overly-cautious measures due to occlusions might result in a slower, less efficient journey. Some researchers have investigated the human drivers' acceptance of proactive braking systems [115]. The proposed ADS framework can also produce proactive braking behavior, which should also be examined from a human factor perspective.

2. **Cooperative behavior** As roads become populated with a mix of autonomous and manually-driven vehicles, fostering cooperative behavior becomes essential. For example, when the AV takes cautious actions under occlusions, it should prevent making surrounding vehicles, especially those behind, annoyed by braking hard. Instead of solely relying on reactive measures, the AV could proactively communicate its intentions to other road users, ensuring safer and more harmonious interactions. This could be particularly useful in occluded scenarios where both human and AV drivers need to make split-second decisions.

APPENDIX A

Structure Details of the CLAP framework

In this appendix, the design details of the CLAP framework will be explained. The full architecture of the framework as deployed on the testing vehicle from Tsinghua University [33] is illustrated in Figure A.1 [302, 33].

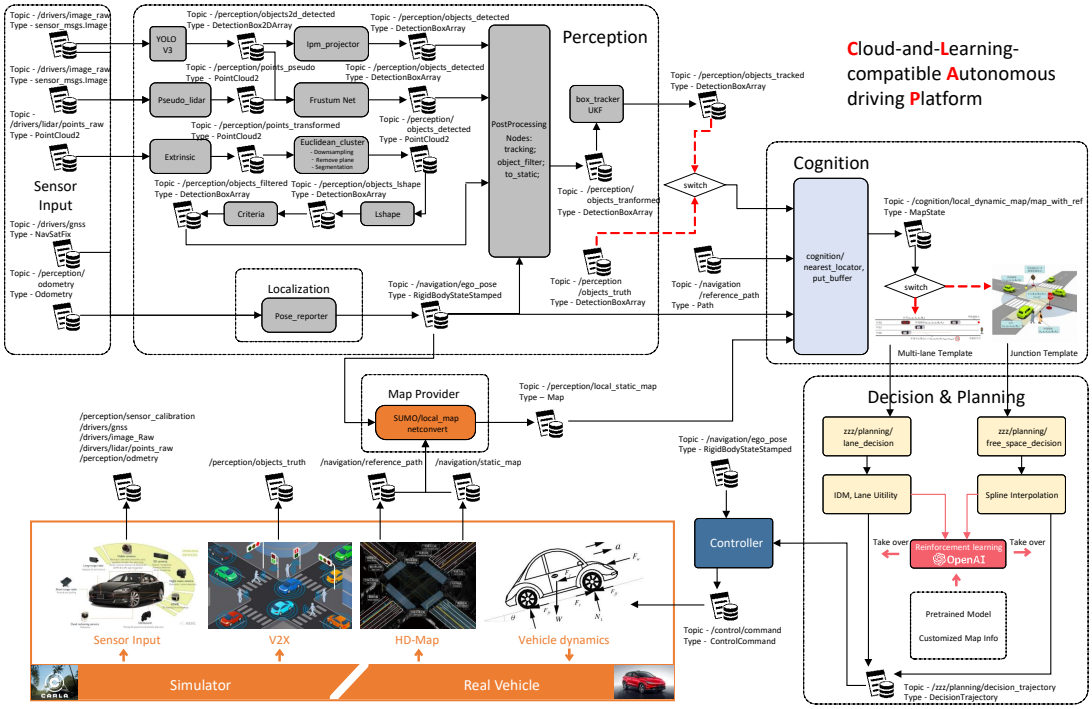


Figure A.1: Detailed structure of the platform components deployed in the use cases.

For the input and output of this framework, we designed abstraction layers (in ROS) to adapt to different data sources and sinks. For inputs, the platform supports reading data from datasets, simulators and physical sensors. For outputs, the platform supports converting the vehicle commands to simulator signals or commands for physical actuators. With the abstraction layers, the transition from dataset training and simulation to real world testing can be eased.

In this platform we also support various map formats through the SUMO library [133]. We have tested the conversion from OpenStreetMap [94] and OpenDrive [6] formats. Other formats including Lanelet [193] and Vissim [89] are also supported by not tested yet. All these formats are converted to the internal format of SUMO, and the local map states are constructed with the help of SUMO interface.

We also have built-in support for perception and planning algorithms. The perception algorithms include YOLOv3 [201], PseudoLidar [261], etc. and the planning algorithms include IDM, MOBIL [123], etc. An important feature of the CLAP platform is that it is friendly to machine learning frameworks by design. Most parts of the code base are written in Python and a OpenAI gym wrapper is officially supported by this platform. Please refer to [33] for applications of RL with this platform.

BIBLIOGRAPHY

[1] National Highway Traffic Safety Administration. Preliminary statement of policy concerning automated vehicles, 2013. [File; accessed 2023-01-04].

[2] Oladapo Afolabi, Katherine Driggs-Campbell, Roy Dong, Mykel J Kochenderfer, and S Shankar Sastry. People as sensors: Imputing maps from human actions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2342–2348. IEEE, 2018.

[3] Florent Altché, Philip Polack, and Arnaud de La Fortelle. High-speed trajectory planning for autonomous vehicles using a simple dynamic model. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 1–7. IEEE, 2017.

[4] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.

[5] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[6] ASAM. ASAM OpenDRIVE, 2021. [Online; accessed 2023-01-04].

[7] Stefan Atev, Grant Miller, and Nikolaos P Papanikolopoulos. Clustering of vehicle trajectories. *IEEE transactions on intelligent transportation systems*, 11(3):647–657, 2010.

[8] Rachid Attia, Rodolfo Orjuela, and Michel Basset. Combined longitudinal and lateral control for automated vehicle guidance. *Vehicle System Dynamics*, 52(2):261–279, 2014.

[9] IPG Automotive. CarMaker — IPG Automotive. [Online; accessed 2022-12-30].

[10] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Paixão, Filipe Mutz, et al. Self-driving cars: A survey. *arXiv preprint arXiv:1901.04407*, 2019.

[11] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

[12] Esmaeil Balal, Ruey Long Cheu, and Thompson Sarkodie-Gyan. A binary decision model for discretionary lane changing move based on fuzzy inference system. *Transportation Research Part C: Emerging Technologies*, 67:47–61, 2016.

[13] Sagar Behere and Martin Törngren. A functional architecture for autonomous driving. In *Proceedings of the First International Workshop on Automotive Software Architecture*, pages 3–10, 2015.

[14] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[15] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31, 2011.

[16] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018.

[17] Massimo Bertozzi, Luca Bombini, Alberto Broggi, Michele Buzzoni, Elena Cardarelli, Stefano Cattani, Pietro Cerri, Alessandro Coati, Stefano Debattisti, Andrea Falzoni, et al. Viac: An out of ordinary experiment. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 175–180. IEEE, 2011.

[18] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[19] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[20] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 3, pages 2743–2748. IEEE, 2003.

[21] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.

[22] Daniel Bogdoll, Felix Schreyer, and J Marius Zöllner. Ad-datasets: a meta-collection of data sets for autonomous driving. *arXiv preprint arXiv:2202.01909*, 2022.

[23] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[24] Farid Bounini, Denis Gingras, Herve Pollart, and Dominique Gruyer. Modified artificial potential field method for online path planning applications. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 180–185. IEEE, 2017.

[25] Maxime Bouton, Alireza Nakhaei, Kikuo Fujimura, and Mykel J Kochenderfer. Scalable decision making with sensor occlusions for autonomous driving. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2076–2081. IEEE, 2018.

[26] Cillian Brewitt, Massimiliano Tamborski, and Stefano V Albrecht. Verifiable goal recognition for autonomous driving with occlusions. *arXiv preprint arXiv:2206.14163*, 2022.

[27] Alberto Broggi, Pietro Cerri, Stefano Debattisti, Maria Chiara Laghi, Paolo Medici, Daniele Molinari, Matteo Panciroli, and Antonio Prioletti. Proud—public road urban driverless-car test. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3508–3519, 2015.

[28] Alberto Broggi, Paolo Medici, Paolo Zani, Alessandro Coati, and Matteo Panciroli. Autonomous vehicles control in the vislab intercontinental autonomous challenge. *Annual Reviews in Control*, 36(1):161–171, 2012.

[29] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer, 2008.

[30] Nina Brouwer, Horst Kloeden, and Christoph Stiller. Comparison and evaluation of pedestrian motion models for vehicle safety systems. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2207–2212. IEEE, 2016.

[31] Shashi D Buluswar and Bruce A Draper. Color machine vision for autonomous vehicles. *Engineering Applications of Artificial Intelligence*, 11(2):245–256, 1998.

[32] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[33] Zhong Cao. *Principle and Self-Learning Hybrid Planning for Automated Vehicles*. PhD thesis, Tsinghua University, 2020.

[34] Zhong Cao, Jiaxin Liu, Weitao Zhou, Xinyu Jiao, and Diange Yang. Lidar-based object detection failure tolerated autonomous driving planning system. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 122–128. IEEE, 2021.

[35] Zhong Cao, Diange Yang, Kun Jiang, Tinghan Wang, Xinyu Jiao, and Zhongyang Xiao. End-to-end adaptive cruise control based on timing network. In *Society of Automotive Engineers (SAE)-China Congress*, pages 839–852. Springer, 2017.

[36] Zhong Cao, Diange Yang, Kun Jiang, Shaobing Xu, Sijia Wang, Minghan Zhu, and Zhongyang Xiao. A geometry-driven car-following distance estimation algorithm robust to road slopes. *Transportation research part C: emerging technologies*, 102:274–288, 2019.

[37] Zhong Cao, Diange Yang, Shaobing Xu, Huei Peng, Boqi Li, Shuo Feng, and Ding Zhao. Highway exiting planner for automated vehicles using reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):990–1000, 2020.

[38] Vinicius Cardoso, Josias Oliveira, Thomas Teixeira, Claudine Badue, Filipe Mutz, Thiago Oliveira-Santos, Lucas Veronese, and Alberto F De Souza. A model-predictive motion planner for the iara autonomous car. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 225–230. IEEE, 2017.

[39] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017.

[40] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[41] Christoforos Chatzikomis, Aldo Sorniotti, Patrick Gruber, Mattia Zanchetta, Dan Willans, and Bryn Balcombe. Comparison of path tracking and torque-vectoring controllers for autonomous electric vehicles. *IEEE Transactions on Intelligent Vehicles*, 3(4):559–570, 2018.

[42] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.

[43] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[44] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[45] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019.

[46] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. *arXiv preprint arXiv:2102.04530*, 2021.

[47] Minkyu Cheon, Wonju Lee, Changyong Yoon, and Mignon Park. Vision-based vehicle detection system with consideration of the detecting location. *IEEE transactions on intelligent transportation systems*, 13(3):1243–1252, 2012.

[48] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[49] Laurène Claussmann, Marc Revilloud, Sébastien Glaser, and Dominique Gruyer. A study on ai-based approaches for high-level decision making in highway autonomous driving. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3671–3676. IEEE, 2017.

[50] Laurene Claussmann, Marc Revilloud, Dominique Gruyer, and Sébastien Glaser. A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1826–1848, 2019.

[51] SAE On-Road Automated Driving (ORAD) Committee. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, 2012.

[52] Ltd. Coppelia Robotics. CoppeliaSim - CoppeliaRobotics. [Online; accessed 2022-12-30].

[53] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving. *arXiv preprint arXiv:2003.03653*, 2020.

[54] Serdar Coskun and Reza Langari. Predictive fuzzy markov decision strategy for autonomous driving in highways. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1032–1039. IEEE, 2018.

[55] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[56] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019.

[57] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[58] Radu Danescu, Florin Oniga, and Sergiu Nedevschi. Modeling and tracking the driving environment with a particle-based occupancy grid. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1331–1342, 2011.

[59] Somnath Deb, Murali Yeddanapudi, Krishna Pattipati, and Yaakov Bar-Shalom. A generalized sd assignment algorithm for multisensor-multitarget state estimation. *IEEE Transactions on Aerospace and Electronic systems*, 33(2):523–538, 1997.

[60] Ezequiel Debada, Adeline Ung, and Denis Gillet. Occlusion-aware motion planning at roundabouts. *IEEE Transactions on Intelligent Vehicles*, 6(2):276–287, 2020.

[61] Jory Denny, Evan Greco, Shawna Thomas, and Nancy M Amato. Marrt: Medial axis biased rapidly-exploring random trees. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 90–97. IEEE, 2014.

[62] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020.

[63] Kevin Doherty, Tixiao Shan, Jinkun Wang, and Brendan Englot. Learning-aided 3-d occupancy mapping with bayesian generalized kernel inference. *IEEE Transactions on Robotics*, 35(4):953–966, 2019.

[64] Yiqun Dong, Yuanxin Zhong, and Jiajun Hong. Knowledge-biased sampling-based path planning for automated vehicles parking. *IEEE Access*, 8:156818–156827, 2020.

[65] Yiqun Dong, Yuanxin Zhong, Wenbo Yu, Minghan Zhu, Pingping Lu, Yeyang Fang, Jiajun Hong, and Huei Peng. Mcity data collection for automated vehicles study. *arXiv preprint arXiv:1912.06258*, 2019.

[66] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[67] Katherine Driggs-Campbell, Vijay Govindarajan, and Ruzena Bajcsy. Integrating intuitive driver models in autonomous planning for interactive maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3461–3472, 2017.

[68] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[69] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, and Séverin Lemaignan. Modular open robots simulation engine: Morse. In *2011 IEEE International Conference on Robotics and Automation*, pages 46–51. IEEE, 2011.

[70] Ashok Elluswamy. Tesla Talk on CVPR 2022 Workshop on Autonomous Driving, 2022. [Video].

[71] Ashok Elluswamy. Tesla AI day 2022: FSD Planning, 2022. [Video].

[72] Péter Fankhauser and Marco Hutter. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In Anis Koubaa, editor, *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, chapter 5. Springer, 2016.

[73] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983.

[74] Engelmann Francis, Kontogianni Theodora, Hermans Alexander, and Leibe Bastian. Exploring spatial context for 3d semantic segmentation of point clouds. In *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*, 2017.

[75] Simon Fürst, Jürgen Mössinger, Stefan Bunzel, Thomas Weber, Frank Kirschke-Biller, Peter Heitkämper, Gerulf Kinkelin, Kenji Nishikawa, and Klaus Lange. Autosar–a worldwide standard is on the road. In *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, volume 62, page 5, 2009.

[76] Enric Galceran, Edwin Olson, and Ryan M Eustice. Augmented vehicle tracking under occlusions for decision-making in autonomous driving. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3559–3565. IEEE, 2015.

[77] Lu Gan, Ray Zhang, Jessy W Grizzle, Ryan M Eustice, and Maani Ghaffari. Bayesian spatial kernel smoothing for scalable dense semantic mapping. *IEEE Robotics and Automation Letters*, 5(2):790–797, 2020.

[78] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[79] Tom M Gasser, Clemens Arzt, Mihiar Ayoubi, Arne Bartels, Lutz Bürkle, Jana Eier, Frank Flemisch, Dirk Häcker, Tobias Hesse, Werner Huber, et al. Rechtsfolgen zunehmender fahrzeugautomatisierung. *Berichte der Bundesanstalt für Straßenwesen. Unterreihe Fahrzeugtechnik*, (83), 2012.

[80] Andreas Geiger and Bernd Kitt. Object flow: A descriptor for classifying traffic motion. In *2010 IEEE Intelligent Vehicles Symposium*, pages 287–293. IEEE, 2010.

[81] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[82] Tobias Gindele, Sebastian Brechtel, and Rudiger Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, 2015.

[83] Peter G Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986.

[84] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[85] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[86] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. *arXiv preprint arXiv:2102.08945*, 2021.

[87] David González, Joshue Pérez, Ray Lattarulo, Vicente Milanés, and Fawzi Nashashibi. Continuous curvature planning with obstacle avoidance capabilities in urban scenarios. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1430–1435. IEEE, 2014.

[88] David Sierra González, Anshul Paigwar, Özgür Erkent, Jilles Dibangoye, and Christian Laugier. Leveraging dynamic occupancy grids for 3d object detection in point clouds. In *ICARCV 2020-16th IEEE International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE, 2020.

[89] PTV Group. Traffic Simulation Software — PTV Vissim — PTV Group. [Online; accessed 2022-12-30].

[90] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021.

[91] Vitor Guizilini, Ransalu Senanayake, and Fabio Ramos. Dynamic hilbert maps: Real-time occupancy predictions in changing environments. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4091–4097. IEEE, 2019.

[92] Jinghua Guo, Yugong Luo, Keqiang Li, and Yifan Dai. Coordinated path-following and direct yaw-moment control of autonomous electric vehicles with sideslip angle estimation. *Mechanical Systems and Signal Processing*, 105:183–199, 2018.

[93] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018.

[94] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18, 2008.

[95] Josiah P Hanna, Arrasy Rahman, Elliot Fosong, Francisco Eiras, Mihai Dobre, John Redford, Subramanian Ramamoorthy, and Stefano V Albrecht. Interpretable goal recognition in the presence of occluded factors for autonomous vehicles. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7044–7051. IEEE, 2021.

[96] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[97] Anselm Haselhoff and Anton Kummert. A vehicle detection system based on haar and triangle features. In *2009 IEEE intelligent vehicles symposium*, pages 261–266. IEEE, 2009.

[98] Hu He and Ben Upcroft. Nonparametric semantic segmentation for 3d street scenes. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3697–3703. IEEE, 2013.

[99] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[100] Gisli R Hjaltason and Hanan Samet. Speeding up construction of pmr quadtree-based spatial indexes. *The VLDB Journal*, 11(2):109–137, 2002.

[101] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[102] Carl-Johan Hoel, Krister Wolff, and Leo Laine. Automated speed and lane change decision making using deep reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2148–2155. IEEE, 2018.

[103] Fangzhou Hong, Hui Zhou, Xinge Zhu, Hongsheng Li, and Ziwei Liu. Lidar-based panoptic segmentation via dynamic shifting network. *arXiv preprint arXiv:2011.11964*, 2020.

[104] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[105] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *Conference on Robot Learning*, pages 409–418. PMLR, 2021.

[106] Albert S Huang, Edwin Olson, and David C Moore. Lcm: Lightweight communications and marshalling. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4057–4062. IEEE, 2010.

[107] Ling Huang, Hengcong Guo, Ronghui Zhang, Haiwei Wang, and Jianping Wu. Capturing drivers' lane changing behaviors on operational level by data driven methods. *IEEE Access*, 6:57497–57506, 2018.

[108] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 954–960, 2018.

[109] Constantin Hubmann, Nils Quetschlich, Jens Schulz, Julian Bernhard, Daniel Althoff, and Christoph Stiller. A pomdp maneuver planner for occlusions in urban scenarios. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2172–2179. IEEE, 2019.

[110] Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. *IEEE transactions on intelligent vehicles*, 3(1):5–17, 2018.

[111] Rasheed Hussain and Sherali Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2018.

[112] Shantanu Ingle and Madhuri Phute. Tesla autopilot: semi autonomous driving, an uptick for future autonomy. *International Research Journal of Engineering and Technology*, 3(9), 2016.

[113] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

[114] Masha Itkina, Ye-Ji Mun, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Multi-agent variational occlusion inference using people as sensors. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4585–4591. IEEE, 2022.

[115] Takuma Ito, Ryosuke Matsumi, Yuichi Saito, Akito Yamasaki, Shintaro Inoue, Tsukasa Shimizu, Masao Nagai, Hideo Inoue, and Minoru Kamata. Comparison of proactive braking intervention system acceptability via field operation tests in different regions. *International Journal of Intelligent Transportation Systems Research*, pages 1–26, 2022.

[116] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. Complex urban dataset with multi-level sensors from highly diverse urban environments. *The International Journal of Robotics Research*, page 0278364919843996, 2019.

[117] Yun Jiang. Cruise Under the Hood 2021: Building the Most Advanced AV, 2021. [Video].

[118] Phillip Karle, Maximilian Geisslinger, Johannes Betz, and Markus Lienkamp. Scenario understanding and motion prediction for autonomous vehicles-review and comparison. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[119] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[120] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[121] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[122] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[123] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

[124] Arne Kesting, Martin Treiber, and Dirk Helbing. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605, 2010.

[125] Ayoub Khammari, Fawzi Nashashibi, Yotam Abramson, and Claude Laurgeau. Vehicle detection combining gradient analysis and adaboost classification. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pages 66–71. IEEE, 2005.

[126] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505. IEEE, 1985.

[127] Soohwan Kim and Jonghyuk Kim. Gpmap: A unified framework for robotic mapping based on sparse gaussian processes. In *Field and service robotics*, pages 319–332. Springer, 2015.

[128] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019.

[129] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020.

[130] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154. IEEE, 2004.

[131] Adam Kortylewski, Qing Liu, Huiyu Wang, Zhishuai Zhang, and Alan Yuille. Combining compositional models and deep networks for robust object classification under occlusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1333–1341, 2020.

[132] Markus Koschi and Matthias Althoff. Set-based prediction of traffic participants considering occlusions and traffic rules. *IEEE Transactions on Intelligent Vehicles*, 6(2):249–265, 2020.

[133] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012.

[134] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[135] Andreas Kuehnle. Symmetry-based recognition of vehicle rears. *Pattern recognition letters*, 12(4):249–258, 1991.

[136] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[137] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*, pages 703–718. Springer, 2014.

[138] L'ubor Ladickỳ, Chris Russell, Pushmeet Kohli, and Philip HS Torr. Associative hierarchical crfs for object class image segmentation. In *2009 IEEE 12th international conference on computer vision*, pages 739–746. IEEE, 2009.

[139] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[140] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics*, pages 303–307, 2001.

[141] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 336–345, 2017.

[142] Unghui Lee, Sangyol Yoon, HyunChul Shim, Pascal Vasseur, and Cedric Demonceaux. Local path planning in a complex environment for self-driving car. In *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, pages 445–450. IEEE, 2014.

[143] Stéphanie Lefevre, Yiqi Gao, Dizan Vasquez, H Eric Tseng, Ruzena Bajcsy, and Francesco Borrelli. Lane keeping assistance with learning-based driver model and model predictive control. In *12th International Symposium on Advanced Vehicle Control*, 2014.

[144] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE international conference on robotics and automation*, pages 4372–4378. IEEE, 2010.

[145] Guofa Li, Shenglong Li, Shen Li, Yechen Qin, Dongpu Cao, Xingda Qu, and Bo Cheng. Deep reinforcement learning enabled decision-making for autonomous driving at intersections. *Automotive Innovation*, 3(4):374–385, 2020.

[146] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Conditional generative neural system for probabilistic trajectory prediction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6150–6156. IEEE, 2019.

[147] Jiachen Li, Wei Zhan, and Masayoshi Tomizuka. Generic vehicle tracking framework capable of handling occlusions based on modified mixture particle filter. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 936–942. IEEE, 2018.

[148] Nan Li, Ilya Kolmanovsky, Anouck Girard, and Yildiray Yildiz. Game theoretic modeling of vehicle interactions at unsignalized intersections and application to autonomous vehicle control. In *2018 Annual American Control Conference (ACC)*, pages 3215–3220. IEEE, 2018.

[149] Xiaohui Li, Zhenping Sun, Dongpu Cao, Daxue Liu, and Hangen He. Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles. *Mechanical Systems and Signal Processing*, 87:118–137, 2017.

[150] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[151] Xiao Lin, Jiucai Zhang, Jin Shang, Yi Wang, Hongkai Yu, and Xiaoli Zhang. Decision making through occluded intersections for autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2449–2455. IEEE, 2019.

[152] Chang Liu, Seungho Lee, Scott Varnhagen, and H Eric Tseng. Path planning for autonomous vehicles using model predictive control. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 174–179. IEEE, 2017.

[153] Tie Liu, Nanning Zheng, Li Zhao, and Hong Cheng. Learning based symmetric features selection for vehicle detection. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 124–129. IEEE, 2005.

[154] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[155] Zhengzhe Liu, Xiaojuan Qi, and Chi-Wing Fu. One thing one click: A self-training approach for weakly supervised 3d semantic segmentation. *arXiv preprint arXiv:2104.02246*, 2021.

[156] Baidu LLC. Apollo Open Platform, 2017. [Online; accessed 29-August-2019].

[157] Cruise LLC. How we're wired. [Online; accessed 2022-12-10].

[158] Waymo LLC. Waymo Driver. [Online; accessed 2022-12-10].

[159] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pages 2575–2582. IEEE, 2018.

[160] Pingping Lu, Shaobing Xu, and Huei Peng. Graph-embedded lane detection. *IEEE Transactions on Image Processing*, 30:2977–2988, 2021.

[161] Hengbo Ma, Jiachen Li, Wei Zhan, and Masayoshi Tomizuka. Wasserstein generative learning with kinematic constraints for probabilistic interactive driving behavior prediction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2477–2483. IEEE, 2019.

[162] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[163] Elisa Martinez, Marta Díaz, Javier Melenchón, Josh A Montero, Ignasi Iriondo, and Joan Claudi Socoró. Driving assistance system based on the detection of head-on collisions. In *2008 IEEE Intelligent Vehicles Symposium*, pages 913–918. IEEE, 2008.

[164] Jilin Mei, Biao Gao, Donghao Xu, Wen Yao, Xijun Zhao, and Huijing Zhao. Semantic segmentation of 3d lidar data in dynamic scene using semi-supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2496–2509, 2019.

[165] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[166] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[167] Kaouther Messaoud, Nachiket Deo, Mohan M Trivedi, and Fawzi Nashashibi. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 165–170. IEEE, 2021.

[168] Gregory P Meyer, Jake Charland, Darshan Hegde, Ankit Laddha, and Carlos Vallespi-Gonzalez. Sensor fusion for joint 3d object detection and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[169] Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2156–2162. IEEE, 2018.

[170] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[171] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[172] Eduardo Rauh Müller, Rodrigo Castelan Carlson, and Werner Kraus Junior. Intersection control for automated vehicles with milp. *IFAC-PapersOnLine*, 49(3):37–42, 2016.

[173] Ye-Ji Mun, Masha Itkina, Shuijing Liu, and Katherine Driggs-Campbell. Occlusion-aware crowd navigation using people as sensors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12031–12037. IEEE, 2023.

[174] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[175] Yannik Nager, Andrea Censi, and Emilio Frazzoli. What lies in the shadows? safe and computation-aware motion planning for autonomous vehicles using intent-aware dynamic shadow regions. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5800–5806. IEEE, 2019.

[176] Maximilian Naumann, Hendrik Konigshof, Martin Lauer, and Christoph Stiller. Safe but not overcautious motion planning under occlusions and limited sensor range. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 140–145. IEEE, 2019.

[177] Garrison Neel and Srikanth Saripalli. Improving bounds on occluded vehicle states for use in safe motion planning. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 268–275. IEEE, 2020.

[178] Daniel Chi Kit Ngai and Nelson Hon Ching Yung. A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):509–522, 2011.

[179] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013.

[180] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6819–6828, 2018.

[181] Julia Nilsson, Mattias Brännström, Erik Coelingh, and Jonas Fredriksson. Lane change maneuvers for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1087–1096, 2016.

[182] NVIDIA. DRIVE Sim Powered by Omniverse — NVIDIA. [Online; accessed 2022-12-30].

[183] Simon O'Callaghan and Fabio Ramos. Continuous occupancy mapping with integral kernels. In *Proceedings of the AAAI conference on artificial intelligence*, volume 25, 2011.

[184] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373. IEEE, 2017.

[185] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

[186] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[187] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[188] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *International Conference on Robotics and Automation*, 2019.

[189] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang Jr. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.

[190] Quang-Hieu Pham, Pierre Sevestre, Ramanpreet Singh Pahwa, Huijing Zhan, Chun Ho Pang, Yuda Chen, Armin Mustafa, Vijay Chandrasekhar, and Jie Lin. A* 3d dataset: Towards autonomous driving in challenging environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2267–2273. IEEE, 2020.

[191] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021.

[192] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616. IEEE, 2014.

[193] Fabian Poggenhans, Jan-Hendrik Pauls, Johannes Janosovits, Stefan Orf, Maximilian Naumann, Florian Kuhnt, and Matthias Mayr. Lanelet2: A high-definition map framework for the future of automated driving. In *Proc. IEEE Intell. Trans. Syst. Conf.*, Hawaii, USA, November 2018.

[194] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[195] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[196] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[197] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[198] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[199] Craig Quiter. Deepdrive. [Online; accessed 2022-12-30].

[200] Fabio Ramos and Lionel Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, 2016.

[201] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[202] Eike Rehder, Jannik Quehl, and Christoph Stiller. Driving like a human: Imitation learning for path planning using convolutional neural networks. In *International Conference on Robotics and Automation Workshops*, pages 1–5, 2017.

[203] Lennart Reiher, Bastian Lampe, and Lutz Eckstein. A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE, 2020.

[204] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[205] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE, 2020.

[206] José Manuel Gaspar Sánchez, Truls Nyberg, Christian Pek, Jana Tumova, and Martin Törngren. Foresee the unseen: Sequential reasoning about hidden obstacles for safe driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 255–264. IEEE, 2022.

[207] Kenny Schlegel, Peter Weissig, and Peter Protzel. A blind-spot-aware optimization-based planner for safe robot navigation. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2021.

[208] Matthew Schwall, Tom Daniel, Trent Victor, Francesca Favaro, and Henning Hohnhold. Waymo public road safety performance data. *arXiv preprint arXiv:2011.00038*, 2020.

[209] Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip HS Torr. Urban 3d semantic modelling using stereo vision. In *2013 IEEE International Conference on robotics and Automation*, pages 580–585. IEEE, 2013.

[210] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[211] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[212] Omveer Sharma, Nirod C Sahoo, and Niladri B Puhan. Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey. *Engineering applications of artificial intelligence*, 101:104211, 2021.

[213] Amnon Shashua, Shai Shalev-Shwartz, and Shaked Shammah. Implementing the RSS Model on NHTSA Pre-Crash Scenarios. 2018.

[214] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016.

[215] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[216] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2(3), 2019.

[217] Xuejie Shi, Bin Kong, and Fei Zheng. A new lane detection method based on feature pattern. In *2009 2nd International Congress on Image and Signal Processing*, pages 1–5. IEEE, 2009.

[218] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[219] Siemens. Simcenter Prescan — Siemens Software. [Online; accessed 2022-12-30].

[220] Mechanical Simulation. CarSim Overview. [Online; accessed 2022-12-30].

[221] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[222] Sayanan Sivaraman and Mohan M Trivedi. Active learning for on-road vehicle detection: A comparative study. *Machine vision and applications*, 25(3):599–611, 2014.

[223] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014.

[224] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip HS Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC-British Machine Vision Conference*. BMVA, 2009.

[225] Hao Sun, Weiwen Deng, Sumin Zhang, Shanshan Wang, and Yutan Zhang. Trajectory planning for vehicle autonomous driving with uncertainties. In *Proceedings 2014 International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)*, pages 34–38. IEEE, 2014.

[226] Jie Sun and Xiaoping Gu. Device of clearing sensor automatically, February 25 2020. US Patent App. 29/657,576.

[227] Liting Sun, Wei Zhan, Ching-Yao Chan, and Masayoshi Tomizuka. Behavior planning of autonomous cars with social perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 207–213. IEEE, 2019.

[228] Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2111–2117. IEEE, 2018.

[229] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[230] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[231] Ömer Şahin Taş, Stefan Hörmann, Bernd Schäufele, and Florian Kuhnt. Automated vehicle system architecture with performance assessment. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

[232] Soo Siang Teoh and Thomas Bräunl. Symmetry-based monocular vehicle detection system. *Machine Vision and Applications*, 23(5):831–842, 2012.

[233] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[234] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[235] Ran Tian, Sisi Li, Nan Li, Ilya Kolmanovsky, Anouck Girard, and Yildiray Yildiz. Adaptive game-theoretic decision making for autonomous vehicle control at roundabouts. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 321–326. IEEE, 2018.

[236] Tomer Toledo, Haris N Koutsopoulos, and Moshe E Ben-Akiva. Modeling integrated lane-changing behavior. *Transportation Research Record*, 1857(1):30–38, 2003.

[237] Kailin Tong, Zlatan Ajanovic, and Georg Stettinger. Overview of tools supporting planning for automated driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[238] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[239] Manuel Trierweiler, Tobias Peterseim, and Cornelius Neumann. Automotive lidar pollution detection system based on total internal reflection techniques. In *Light-Emitting Devices, Materials, and Applications XXIV*, volume 11302, pages 135–144. SPIE, 2020.

[240] Ozan Unal, Luc Van Gool, and Dengxin Dai. Improving point cloud semantic segmentation by learning 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2950–2959, 2021.

[241] Michal Uřičář, Pavel Křížek, Ganesh Sistu, and Senthil Yogamani. Soilingnet: Soiling detection on automotive surround-view cameras. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 67–72. IEEE, 2019.

[242] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 25(8):425–466, 2008.

[243] Benoit Vanholme, Dominique Gruyer, Benoit Lusetti, Sebastien Glaser, and Said Mammar. Highly automated driving on highways based on legal safety. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):333–347, 2012.

[244] Jorge Vargas, Suleiman Alsweiss, Onur Toker, Rahul Razdan, and Joshua Santos. An overview of autonomous vehicles sensors and their vulnerability to weather conditions. *Sensors*, 21(16):5397, 2021.

[245] VI-Grade. ADAS driving simulator — VI-grade. [Online; accessed 2022-12-30].

[246] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[247] Matt Vitelli and Aran Nayebi. Carma: A deep reinforcement learning approach to autonomous driving. *Tech. rep. Stanford University, Tech. Rep.*, 2016.

[248] Felix Von Hundelshausen, Michael Himmelsbach, Falk Hecker, Andre Mueller, and Hans-Joachim Wuensche. Driving with tentacles: Integral structures for sensing and motion. *Journal of Field Robotics*, 25(9):640–673, 2008.

[249] Angtian Wang, Yihong Sun, Adam Kortylewski, and Alan L Yuille. Robust object detection under occlusion with context-aware compositionalnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12645–12654, 2020.

[250] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15. Rome, Italy, 2015.

[251] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020.

[252] Jinkun Wang and Brendan Englot. Fast, accurate gaussian process occupancy maps via test-data octrees and nested bayesian fusion. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1003–1010. IEEE, 2016.

[253] Junjie Wang, Qichao Zhang, Dongbin Zhao, and Yaran Chen. Lane change decision-making through deep reinforcement learning with rule-based constraints. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

[254] Lizi Wang, Hongkai Ye, Qianhao Wang, Yuman Gao, Chao Xu, and Fei Gao. Learning-based 3d occupancy prediction for autonomous navigation in occluded environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4509–4516. IEEE, 2020.

[255] Miao Wang, Tinosch Ganjineh, and Raúl Rojas. Action annotated trajectory generation for autonomous maneuvers on structured road networks. In *The 5th International conference on automation, robotics and applications*, pages 67–72. IEEE, 2011.

[256] Pin Wang, Ching-Yao Chan, and Arnaud de La Fortelle. A reinforcement learning based approach for automated lane change maneuvers. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1379–1384. IEEE, 2018.

[257] Sijia Wang, Kun Jiang, Shichao Xie, Yuanxin Zhong, Pengwei Guo, Qun Wu, and Diange Yang. A unified spatio-temporal description model of environment for intelligent vehicles. In *Proceedings of China SAE Congress 2019: Selected Papers*, pages 585–596. Springer Singapore, 2020.

[258] Xinpeng Wang, Huei Peng, Songan Zhang, and Kuan-Hui Lee. An interaction-aware evaluation method for highly automated vehicles. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 394–401. IEEE, 2021.

[259] Xinpeng Wang, Huei Peng, and Ding Zhao. Combining reachability analysis and importance sampling for accelerated evaluation of highly automated vehicles at pedestrian crossing. In *ASME 2019 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

[260] Xinpeng Wang, Songan Zhang, and Huei Peng. Comprehensive safety evaluation of highly automated vehicles at the roundabout scenario. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20873–20888, 2022.

[261] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[262] Ze Wang, Sihao Ding, Ying Li, Jonas Fenn, Sohini Roychowdhury, Andreas Wallin, Lane Martin, Scott Ryvola, Guillermo Sapiro, and Qiang Qiu. Cirrus: A long-range bi-pattern lidar dataset. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5744–5750. IEEE, 2021.

[263] James R Ward, Gabriel Agamennoni, Stewart Worrall, Asher Bender, and Eduardo Nebot. Extending time to collision for probabilistic reasoning in general traffic scenarios. *Transportation Research Part C: Emerging Technologies*, 51:66–82, 2015.

[264] Junqing Wei, Jarrod M Snider, Junsung Kim, John M Dolan, Raj Rajkumar, and Bakhtiar Litkouhi. Towards a viable autonomous driving research platform. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 763–770. IEEE, 2013.

[265] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6499–6508, 2020.

[266] Moritz Werling, Sören Kammel, Julius Ziegler, and Lutz Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, 2012.

[267] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010.

[268] Thomas Wiemann, Kai Lingemann, and Joachim Hertzberg. Optimizing triangle mesh reconstructions of planar environments. *IFAC-PapersOnLine*, 49(15):218–223, 2016.

[269] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[270] Kelvin Wong, Yanlei Gu, and Shunsuke Kamijo. Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine*, 13(1):91–106, 2020.

[271] Hanwool Woo, Yonghoon Ji, Hitoshi Kono, Yusuke Tamura, Yasuhide Kuroda, Takashi Sugano, Yasunori Yamamoto, Atsushi Yamashita, and Hajime Asama. Lane-change detection based on vehicle-trajectory prediction. *IEEE Robotics and Automation Letters*, 2(2):1109–1116, 2017.

[272] Kyle Hollins Wray, Bernard Lange, Arec Jamgochian, Stefan J Witwicki, Atsuhide Kobashi, Sachin Hagaribommanahalli, and David Ilstrup. Pomdps for safe visibility reasoning in autonomous vehicles. In *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*, pages 191–195. IEEE, 2021.

[273] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[274] Ning Wu, Weiwei Huang, Zhiwei Song, Xiaojun Wu, Qun Zhang, and Susu Yao. Adaptive dynamic preview control for autonomous vehicle trajectory following with ddp based path planner. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1012–1017. IEEE, 2015.

[275] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4(6):2, 2000.

[276] Shichao Xie, Diange Yang, Kun Jiang, and Yuanxin Zhong. Pixels and 3-d points alignment method for the fusion of camera and lidar data. *IEEE Transactions on Instrumentation and Measurement*, 68(10):3661–3676, 2018.

[277] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R Qi, and Dragomir Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15446–15456, 2021.

[278] Shaobing Xu and Huei Peng. Design, analysis, and experiments of preview path tracking control for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(1):48–58, 2019.

[279] Shaobing Xu, Huei Peng, Pingping Lu, Minghan Zhu, and Yifan Tang. Design and experiments of safeguard protected preview lane keeping control for autonomous vehicles. *IEEE Access*, 8:29944–29953, 2020.

[280] Shaobing Xu, Robert Zidek, Zhong Cao, Pingping Lu, Xinpeng Wang, Boqi Li, and Huei Peng. System and experiments of model-driven motion planning and control for autonomous vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[281] Jianru Xue, Jianwu Fang, Tao Li, Bohua Zhang, Pu Zhang, Zhen Ye, and Jian Dou. Blvd: Building a large-scale 5d semantics benchmark for autonomous driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6685–6691. IEEE, 2019.

[282] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[283] DianGe Yang, Kun Jiang, Ding Zhao, ChunLei Yu, Zhong Cao, ShiChao Xie, ZhongYang Xiao, XinYu Jiao, SiJia Wang, and Kai Zhang. Intelligent and connected vehicles: Current status and future perspectives. *Science China Technological Sciences*, 61(10):1446–1471, 2018.

[284] Diange Yang, Xinyu Jiao, Kun Jiang, and Zhong Cao. Driving space for autonomous vehicles. *Automotive Innovation*, 2(4):241–253, 2019.

[285] Shichao Yang, Yulan Huang, and Sebastian Scherer. Semantic 3d occupancy mapping through efficient high order crfs. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 590–597. IEEE, 2017.

[286] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020.

[287] Qiangqiang Yao, Ying Tian, Qun Wang, and Shengyuan Wang. Control strategies on path tracking for autonomous vehicle: State of the art and future challenges. *IEEE Access*, 8:161211–161222, 2020.

[288] Fei Ye, Xuxin Cheng, Pin Wang, Ching-Yao Chan, and Jiucai Zhang. Automated lane change strategy using proximal policy optimization-based deep reinforcement learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1746–1752. IEEE, 2020.

[289] Fei Ye, Shen Zhang, Pin Wang, and Ching-Yao Chan. A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 1073–1080. IEEE, 2021.

[290] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020.

[291] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[292] Senthil Yogamani, Ciaran Hughes, Jonathan Horgan, Ganesh Sistu, Sumanth Chennupati, Michal Uricar, Stefan Milz, Martin Simon, Karl Amende, Christian Witt, et al. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9307–9317. IEEE, 2019.

[293] Je Hong Yoo and Reza Langari. Stackelberg game based model of highway driving. In *Dynamic Systems and Control Conference*, volume 45295, pages 499–508. American Society of Mechanical Engineers, 2012.

[294] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. Occlusion-aware risk assessment for autonomous driving in urban environments. *IEEE Robotics and Automation Letters*, 4(2):2235–2241, 2019.

[295] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. Risk assessment and planning with bidirectional reachability for autonomous driving. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5363–5369. IEEE, 2020.

[296] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Occlusion-aware r-cnn: detecting pedestrians in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 637–653, 2018.

[297] Wenwei Zhang, Zhe Wang, and Chen Change Loy. Multi-modality cut and paste for 3d object detection. *arXiv preprint arXiv:2012.12741*, 2020.

[298] Xizheng Zhang and Xiaolin Zhu. Autonomous path tracking control of intelligent electric vehicles based on lane detection and optimal preview method. *Expert Systems with Applications*, 121:38–48, 2019.

[299] Zixu Zhang and Jaime F Fisac. Safe occlusion-aware autonomous driving via game-theoretic active perception. *arXiv preprint arXiv:2105.08169*, 2021.

[300] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[301] Jianhui Zhao, Xinyu Zhang, Ping Shi, and Yuchao Liu. Automatic driving control method based on time delay dynamic prediction. In *International Conference on Cognitive Systems and Signal Processing*, pages 443–453. Springer, 2016.

[302] Yuanxin Zhong, Zhong Cao, Minghan Zhu, Xinpeng Wang, Diange Yang, and Huei Peng. Clap: Cloud-and-learning-compatible autonomous driving platform. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1450–1456. IEEE, 2020.

[303] Yuanxin Zhong and Huei Peng. Real-time semantic 3d dense occupancy mapping with efficient free space representations. *arXiv preprint arXiv:2107.02981*, 2021.

[304] Yuanxin Zhong and Huei Peng. Driving with caution about fully occluded areas based on occupancy maps. In *2023 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2023.

[305] Yuanxin Zhong, Sijia Wang, Shichao Xie, Zhong Cao, Kun Jiang, and Diange Yang. 3d scene reconstruction with sparse lidar data and monocular image in single frame. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 11(07-11-01-0005):48–56, 2017.

[306] Yuanxin Zhong, Minghan Zhu, and Huei Peng. Uncertainty-aware voxel based 3d object detection and tracking with von-mises loss. *arXiv preprint arXiv:2011.02553*, 2020.

[307] Yuanxin Zhong, Minghan Zhu, and Huei Peng. Vin: Voxel-based implicit network for joint 3d object detection and segmentation for lidars. In *BMVC*, 2021.

[308] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[309] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020.

[310] Shiying Zhou, Yizhou Wang, Minghui Zheng, and Masayoshi Tomizuka. A hierarchical planning and control framework for structured highway driving. *IFAC-PapersOnLine*, 50(1):9101–9107, 2017.

[311] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[312] Yue Zhou, Edward Chung, Ashish Bhaskar, and Michael E Cholette. A state-constrained optimal control based trajectory planning strategy for cooperative freeway mainline facilitating and on-ramp merging maneuvers under congested traffic. *Transportation Research Part C: Emerging Technologies*, 109:321–342, 2019.

[313] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018.

[314] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[315] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. *arXiv preprint arXiv:2011.10033*, 2020.

[316] Robert AE Zidek and Ilya V Kolmanovsky. Optimal driving policies for autonomous vehicles based on stochastic drift counteraction. *IFAC-PapersOnLine*, 50(1):290–296, 2017.

[317] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[318] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G Keller, et al. Making bertha drive—an autonomous journey on a historic route. *IEEE Intelligent transportation systems magazine*, 6(2):8–20, 2014.

[319] Julius Ziegler and Christoph Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE, 2009.

[320] Wenhao Zong, Changzhu Zhang, Zhuping Wang, Jin Zhu, and Qijun Chen. Architecture design and implementation of an autonomous vehicle. *IEEE access*, 6:21956–21970, 2018.