Title
High Definition Map as An Infrastructure for Urban Autonomous Driving

Author
Zhou, Yiyang

Publication Date
2022

Peer reviewed|Thesis/dissertation

High Definition Map as An Infrastructure for Urban Autonomous Driving

by

Yiyang Zhou

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Jonathan R. Shewchuk
Professor Mark W. Mueller

Spring 2022

High Definition Map as An Infrastructure for Urban Autonomous Driving

Abstract

High Definition Map as An Infrastructure for Urban Autonomous Driving

by

Yiyang Zhou

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Technologies related to autonomous driving have been advancing rapidly for the past few years, and the community has already seen promising applications of self-driving vehicles in certain scenarios. However, urban driving environments, with their less structured roads, complicated traffic rules, and numerous visual occlusions, remain to be challenging. To assist the autonomous agents in such scenes, High Definition maps (HD maps) are proposed as an infrastructure for urban autonomous driving. A typical HD map usually has multiple layers of semantics including lane boundaries, drivable areas, and traffic rules. This prior knowledge of the scene is of great importance for downstream modules like perception, localization, prediction, and planning. Both in academia and in industry, nowadays, the HD map has become a fundamental module for autonomous driving.

With their loaded information, the HD maps, however, are not usually easy to obtain. Indeed, each module in the life cycle of an HD map imposes some challenges to researchers. The development of an HD map starts with the sensor setup on mobile mapping platforms and the collection of the mapping data. Typically, the mapping platform needs to be manually calibrated for the correct spatial-temporal relationship among sensors, and careful planning is required to prepare a mapping dataset. Then, the valuable data from these sensors are processed through HD map generation pipelines for future usage. Currently, the map construction requires intensive human labeling and post-processing, and few algorithms are robust enough to automatically map an arbitrary urban environment. Lastly, how to utilize a collection of HD maps is still a challenge for the mapping community. Due to limited onboard storage and computing capacity, loading a complete map would be infeasible, and end-users need an efficient submap query strategy in real-time.

Acknowledging the essential role of the HD map in urban autonomous driving and the difficulties in development, this dissertation discusses multiple perspectives in the complete life cycle of an HD map.

To begin with, this dissertation first discusses the mapping platform and the dataset for mapping applications in Part I. Chapter 2 focuses on the geometrical calibration and synchronization of the sensor suite on a mobile mapping platform. In this section, the com-

plementary LIDAR-camera configuration is discussed, and a semantic-based optimization algorithm is proposed to estimate both the geometric and the temporal relationship between these two sensor modalities. In Chapter 3, an exemplar mapping platform and an urban dataset are introduced. The design of the mapping vehicle considers complicated urban scenarios, and the dataset includes some of the most challenging city driving scenes. The dataset is open to the public to encourage research in the mapping field. With the mapping platform configured, the next question in the life cycle of an HD map is the routing problem. With more than one mapping vehicle, how to efficiently route a mapping fleet is discussed in Chapter 4. Here, a Model Predictive Control-based algorithm is proposed to accommodate traffic conditions and map updating problems.

Part II focuses on the algorithms related to the automatic generation of the HD map. Chapter 5 introduces a particle filter-based algorithm to efficiently explore the lanes in complicated urban situations. The algorithm specifically solves the merging, forking, and irregular lane cases on city roads. Chapter 6 moves more towards the intersections and discusses potential solutions with a multi-sensor setup. Here, the mapping problem is treated as the semantic segmentation in the Bird's Eye View frame. Network design comparisons are also provided to demonstrate a preferred strategy in cross-domain fusion tasks. Chapter 7 studies the potential of multitask learning for both static and dynamic objects on the road to exploit information in limited data. Built upon a single backbone, the proposed method compresses six tasks into one neural network, and the evaluation shows that the performance is comparable with single-task models.

In Part III, the management and the deployment of the HD map are discussed. Chapter 8 introduces a tile-based map management system to query and combine smaller HD maps for real-time application. The proposed framework leverages an RB tree data structure and uses a submap queue during vehicle operation to store only useful maps onboard.

This dissertation concludes with a summary of breakthroughs in the life cycle of the HD map development process and comments on the future directions of HD map-related research.

To my family,

in particular, my grandfather, a true engineer.

Contents

List of Figures

List of Tables

Acknowledgments

"If you're going to San Francisco, You're gonna meet some gentle people there." This song from Scott McKenzie was played when I traversed the continent to UC Berkeley with my father in the summer of 2018. Indeed, I met lots of amazing people here on this lovely little hill, and they made my Ph.D. experience in Berkeley a truly amazing 4-year journey.

In the first place, I would like to express my greatest appreciation and deepest gratitude to Dr. Masayoshi Tomizuka, whose knowledge, enthusiasm, and patience have guided me through my Ph.D. journey. He led me into the research world and taught me valuable life lessons beyond academia. We just celebrated his 76th birthday in the lab, and he is so energetic, revving just like his racing car. During the 4-month preparation of this dissertation, he offered me invaluable advice through numerous in-person meetings and correspondences. This work could not be completed without his generous support and patient guidance.

Special thanks to Dr. Jonathan Shewchuk and Dr. Mark Mueller, who also provided me with great insights into this world. Dr. Shewchuk opened the door of Machine Learning for me in the Spring of 2019 when I was a student in his CS289 class. The final project on sensor fusion inspired my following projects on perception and calibration. Dr. Mueller's ME233 lecture notes helped me to get a deeper understanding of the state estimation problem in controls, inspiring my work on the road module exploration.

I would also like to give a special acknowledgment to Dr. Wei Zhan. Wei's enthusiasm for autonomous driving has inspired so many lab members in the past few years. In particular, he guided me through different modules in autonomous driving and encouraged me to explore my potential in the field of my interest. His insights and leadership will bring more students and engineers towards a better future for autonomous driving.

Through Dr. Tomizuka and Wei's arrangement, I got the honor to work with some of the best researchers in this world! I enjoyed working with Mr. Yuichi Takeda from Nissan, Dr. Di Feng from Bosch, Dr. Weisong Wen from Hong Kong PolyU, and Dr. Di Wang from XJTU.

Furthermore, I would like to thank the members of the MSC lab for their invaluable friendship and camaraderie. I want to express my appreciation for Dr. Liting Sun, Dr. Zining Wang, Dr. Kiwoo Shin, Dr. Yujiao Cheng, Dr. Chen Tang, Dr. Saman Fahandezhsaadi, Dr. Jiachen Li, Dr. Yeping Hu, Dr. Zhuo Xu, Dr. Hengbo Ma, and "Big Sis" Dr. Jessica Leu for the inspiring discussions. I would also like to thank Changhao Wang, Lingfeng Sun, Huidong Mona Gao, Xinghao Zhu, and Ting Xu for supportive homework parties and encouraging conversations. Lastly, I would like to thank Akio Kodaira, Catherine Weaver, Chenran Li, Xiang Zhang, and other lab members who supported me over the past few years. When we were not doing research, the fishing trip with Zining, the ski trips with "Big Sis" and Catherine, and the lab retreat at Crater Lake are some of my best memories at Berkeley!

I would also like to mention Jiaming Zha and Haoyun Xu for the adventurous travels and late-night conversations during our time in California. Another thank goes to He Ren, with whom I passed my qualification exam and am expecting to receive my PhD degree! Her companionship has always been my source of happiness.

Last but not the least, I would like to express my love to my parents and family, whose endless support and encouragement have helped me to become who I am today. My grandfather, a true railroad engineer, has inspired me in numerous ways throughout the years. He always tells me that, as long as I try, there are always more solutions than problems. I would also like to thank my cousin Anqi, who takes care of the family when I am away from home.

Chapter 1

Introduction

1.1 Autonomous Driving and the HD Map

This is 2022. The tale of autonomous vehicles has been told for more than a decade, but most of the full autonomy promises are yet to be delivered. With technology breakthroughs in sensors, computations, and algorithms, the world has witnessed tremendous growth in the autonomous driving industry, and there are indeed some applications of autonomous cars deployed in certain scenarios. However, no one has successfully reached full autonomy in an arbitrary urban environment.

Different from more regulated scenarios like highways or parking lots, urban driving is hard. To begin with, traffic participants are more diversified in cities: walking pedestrians, runners, cyclists, and various sizes of vehicles are together sharing the road. More detrimental to the perception algorithms, these dynamic objects in the scene might occlude each other, creating some unavoidable detection misses. On the other hand, the static road structure in an urban scenario is more complicated as well. Figure 1.1 illustrates some typical roads in the San Francisco Bay Area with satellite images from [41]. Forking or merging lanes are common in cities. Other roads might not have any clear markings on their surfaces. Sometimes urban scenes even have complicated 6-way intersections or roundabouts. To handle these complexities and to alleviate the onboard sensing burden, some prior knowledge about the scene is needed.

To represent this prior knowledge, maps were utilized in the early days of autonomous driving. In the first DARPA urban challenge in 2007, various participating teams were generating their own versions of the route map to facilitate the driving process [101, 148]. These maps were created with LIDARs or cameras to generate an accurate 3D representation of the obstacles and the route itself. These maps showed rudimentary forms of the High Definition Maps (HD Maps) nowadays.

The concept of the HD map was more specifically defined in Mercedes-Benz's Bertha Drive Project in 2013 [82, 173]. More than a map with detailed 3D geometrical information, the map used in Bertha included semantics like lane topology and traffic rules. These

(a) Forking lanes (b) Lanes without markings (c) Complicated roundabout

Figure 1.1: Urban driving scene illustrations

semantics are particularly useful for urban driving tasks because the semantics pre-define how the vehicle should behave in a complicated environment. For example, by specifying the right-turning lanes in the map, the vehicle understands to only use that specific lane when planning a possible right turn.

Nowadays, HD maps carry even more information that is not directly visible to human eyes. Some companies have proposed to include prior information like the usual traffic patterns into the map, and other teams are adding real-time road conditions to assist real-time path planning [12].

As a summary, Figure 1.2 illustrates a commonly acknowledged definition of the HD map hierarchy. Starting with a base layer road network for human-level navigation, the HD map then incorporates accurate measurements of the surrounding environment in the geometric layer. The third layer contains semantic information like lane topology and traffic rules, and more advanced blocks carry prior traffic information or real-time condition updates.

The popularity of HD maps comes from their wide application in driver-less cars. Indeed, researchers in this field have demonstrated their significant roles in different modules in autonomous driving. In perception, for example, maps help the algorithms to identify potential drivable areas as semantic priors [162]. In localization modules, the map provides a globally accurate reference for point clouds or camera images to match with [86, 146]. Furthermore, in behavior prediction modules, traffic rules and lane semantics within the HD map give algorithms a strong hint of the target's future trajectory [33]. Today, even more applications with these HD maps are being developed.

Acknowledging the essential role of the HD map in autonomous driving, the next section dives deep into the development process.

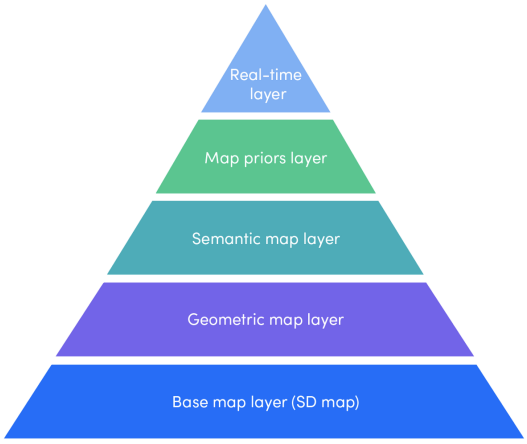


Figure 1.2: HD map hierarchy in autonomous driving

1.2 The Life Cycle of an HD Map and Challenges Thereof

The life of a paper map usually starts with a group of survey engineers in the field. The survey team is equipped with levelers, locators, tripods, etc. to accurately measure the geographical features in the environment. With raw data, specially trained cartographers decode the information and draw one copy of the map. Lastly, publishers or local libraries organize the maps and make them available to the public.

As an analogy of the traditional map-making process, an HD map also requires three parts: hardware and data, map generation algorithms, and map management. Modern HD map makers, just like their predecessors in ancient times, are facing challenges in these tasks. This section explains each mapping module in detail and further introduces the challenges faced by HD map researchers.

The Survey Team

The data source of the HD map comes from mobile mapping platforms (MMP) with various sensors. Here, the mobile platform could be a satellite, an aircraft, a car, or simply a pedestrian [142]. Nowadays, car-mounted mapping platforms are the most popular choices because the sensors are traveling on the same route as future autonomous vehicles. More importantly, the data collected during these travels could also be utilized for detection or prediction tasks.

Some typical sensor choices on an MMP include LIDARs, cameras, Inertial Measurement Units (IMU), Global Positioning Systems (GPS), On-board Diagnostic (OBD) sensors, etc. One of the most important steps in jointly utilizing these sensors is to calibrate them carefully.

Here, one needs to know both the spatial and the temporal relationship among these sensors, and such calibration is not trivial. Currently, researchers are calibrating the sensor suite with carefully prepared artificial rigs and meticulously designed synchronization algorithms [165]. More detrimentally is the irreproducible nature of such one-time calibration. If the sensor suite experiences vibration or deformation in an accident, the geometry calibration result is instantly voided. Thus, automatic calibration is critical for any onboard sensor suite [66].

With a fully functioning mapping platform, the next step is to collect a dataset for downstream tasks. Here, the mapper needs to configure a data logging protocol for the MMP and a database for storage. Furthermore, a mapping team also needs to consider the routing problem for its mapping fleet. In dynamic urban scenarios, the road condition changes every second, and it is important to consider various traffic patterns when routing these mapping vehicles.

When all the data are collected in an accurate and efficient manner, the next step for HD map development lies in the hand of the cartographic algorithm.

The Cartographic Algorithm

To make an HD map, one needs to first define the structure/model of the map. Take OpenStreetMap (OSM) as an example. OSM is an open-source online map database with course road-level information [109]. It constructs a map with a graph, where nodes are waypoints, and edges show the connection relationship. Accommodating the huge amount of information within an HD map, new map structures need to be proposed.

With an established map model, the next step is to encode each part of the model with useful information. Within the map, one needs to fill in semantics like reference waypoints, lane geometry, etc. Furthermore, it is also critical to include how lanes are connected at the intersection. Here, the aforementioned third layer of the HD map is fully explored. However, extracting the semantic information out of the raw sensor data is still largely menial, requiring a significant amount of human labeling [115]. A few commercial products have demonstrated the potential of automatic HD map-making in highway scenes [107], but fully automatic map generation in cities is still challenging.

After the map generation process, these valuable maps are stored on the mapping team's server. Now, how to make them available to each individual autonomous vehicle will be the next question.

The Atlas

A collection of maps is called an atlas. How to make the atlas available to all autonomous vehicles is the question to be answered at the distribution time. Considering the limited storage spaces and computing powers on autonomous vehicles, loading the whole atlas of a city directly onto the vehicle's hardware is often impossible. While some engineers are trying to compress the map to a smaller scale [155], a new methodology to query the map is needed. Instead of loading the complete map, a divide-and-conquer philosophy should be

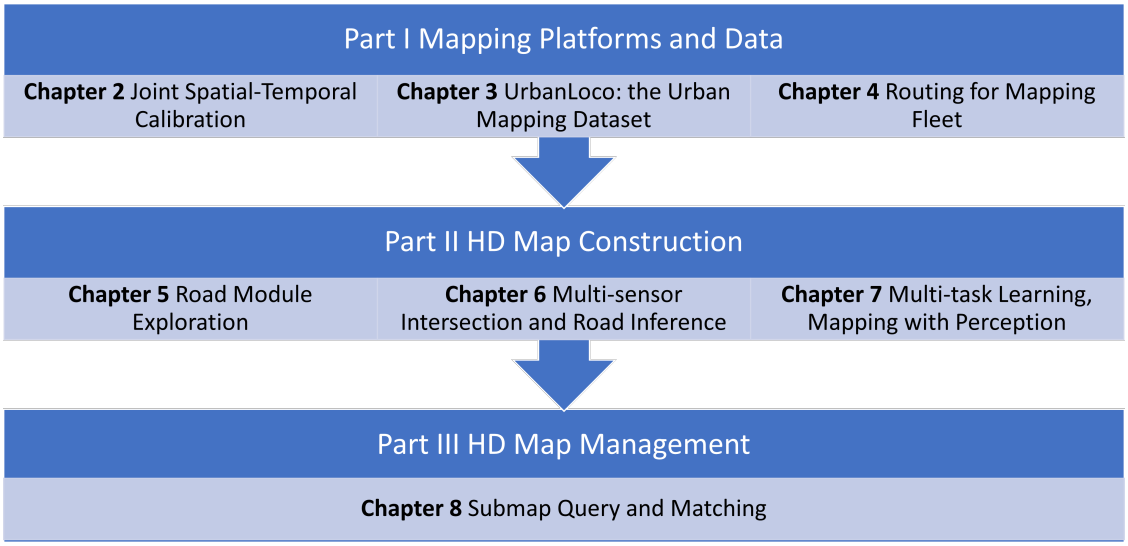


Figure 1.3: Dissertation outline

applied to map management. Here, one potential solution is to store all the HD maps on the cloud, and the onboard system only downloads the current and future segments when necessary [164]. However, more research into the atlas structure and the delivering strategy is needed for a fast query during real-time implementation.

Facing these challenges in each step of the HD map development process, researchers in the mapping domain needs a complete revisit of the HD map infrastructure.

1.3 The Dissertation Outline

Acknowledging the HD map's role as a critical infrastructure for urban autonomous driving, this dissertation is a revisit of each step in the complete HD map life cycle. As shown in Figure 1.3, this work sequentially visits each module mentioned in Section 1.2 and introduces related academic endeavors.

Part I Mapping Platforms and Data

As the first step of the mapping process, this dissertation starts with the sensors on the MMP and the data acquisition process. Here a joint spatial-temporal calibration method for LIDARs and cameras is first introduced as a demonstration of the hardware setup on a mapping vehicle. Then, a sample mapping dataset preparation process is demonstrated as a showcase. Lastly, a Model Predictive Control-based map routing solution is proposed for mapping fleet management.

Chapter 2 Joint Spatial-Temporal Calibration

Different sensor modalities describe a scene with different types of physical information, and multiple sensors together contribute to a more comprehensive representation of the environment. Camera and LIDAR, with complementary semantic and depth information, are the typical choices for detection tasks in complicated driving environments. For most LIDAR-camera fusion algorithms, however, the calibration of the sensor suite will greatly impact the performance. More specifically, these algorithms usually require an accurate geometric relationship among multiple sensors as the input, and they often assume that the contents from these sensors are captured at the same time. Preparing such sensor suites typically involves carefully designed calibration rigs and accurate synchronization mechanisms, which are relatively time-consuming. More critically, the offline calibration result would be voided if the sensor suite experiences any damages during deployment. In this chapter, a segmentation-based framework is proposed to jointly estimate the geometric and the temporal parameters in the calibration of a LIDAR-camera suite. A semantic segmentation mask is first applied to both sensor modalities, and the calibration parameters are optimized through pixel-wise bidirectional loss. Temporal parameters are estimated from visual odometry prediction between two consecutive frames. Since supervision is only performed at the segmentation level, no calibration label is needed within the framework. The proposed algorithm is tested on the KITTI dataset, and the result shows an accurate real-time calibration of both geometric and temporal parameters.

Chapter 3 UrbanLoco: the Urban Mapping Dataset

As introduced in previous sections, mapping and localization are critical modules of autonomous driving, and significant achievements have been reached in related fields. Beyond Global Navigation Satellite System (GNSS), research in point cloud registration, visual feature matching, and inertia navigation has greatly enhanced the accuracy and robustness of mapping and localization in different scenarios. However, highly urbanized scenes are still challenging: LIDAR- and camera-based methods perform poorly with numerous dynamic objects; the GNSS-based solutions experience signal loss and multipath problems; IMUs suffer from drifting. Unfortunately, current public datasets either do not adequately address this urban challenge or do not provide enough sensor information related to mapping and localization. Here we present UrbanLoco: a mapping/localization dataset collected in highly-urbanized environments with a full sensor-suite. The dataset includes 13 MMP trips collected in San Francisco and Hong Kong, covering a total length of over 40 kilometers. The UrbanLoco dataset includes a wide variety of urban terrains: urban canyons, bridges, tunnels, sharp turns, etc. More importantly, this dataset includes the complete information from LIDAR, cameras, IMU, and GNSS receivers. UrbanLoco is publicly available at `https://advdataset2019.wixsite.com/urbanloco`.

Chapter 4 Routing for A Mapping Fleet

To build large HD maps covering an entire city, a mapping team usually commands a fleet of vehicles to traverse different parts of the town to acquire accurate, comprehensive, and up-to-date mapping information of the urban environment. However, how to efficiently route each vehicle in the fleet remains to be a problem. This chapter specifically targets the mapping fleet routing problem in dynamic urban scenes. Here, a Model Predictive Control-based map routing solution MR.MPC is proposed. The method first abstracts a directed cyclic graph representation of the city from public mapping datasets and then constructs a linear model for the exploration task. The model is further extended to solve map updating problems. The proposed method is tested with both simulated and real-world environments, and the results demonstrate an efficient and robust exploration strategy.

Part II HD Map Construction

After the data is collected with calibrated sensors on mapping vehicles, the next step is to extract the mapping information. In Chapter 5, an urban map structure is firstly defined, and a road module exploration strategy is introduced. Chapter 6 extends the map exploration into intersections and proposes to use Bird's Eye View (BEV) segmented images as local representations. Chapter 7 combines static mapping tasks with dynamic object detection for multi-task learning.

Chapter 5 Road Module Exploration

As a crucial layer of the HD map, lane-level maps are particularly useful: they contain geometrical and topological information for both lanes and intersections. However, large-scale construction of HD maps is limited by tedious human labeling and high maintenance costs, especially for urban scenarios with complicated road structures and irregular markings. This chapter proposes an approach based on semantic particle filter to tackle the automatic lane-level mapping problem in urban scenes. The map skeleton is firstly structured as a directed cyclic graph from the online mapping database OpenStreetMap. The proposed method then performs semantic segmentation on 2D front-view images from ego vehicles and explores the lane semantics on a birds-eye-view domain with true topographical projection. Exploiting OpenStreetMap, we further infer the lane topology and the reference trajectory at intersections with the aforementioned lane semantics. The proposed algorithm has been tested in densely urbanized areas, demonstrating an accurate and robust reconstruction of the lane-level HD map.

Chapter 6 Multi-sensor Intersection and Road Inference

Either for an HD map construction or for an online perception of the static scene, it requires more than a simple semantic segmentation or lane tracing in the camera domain. Instead, it is essential to form a BEV representation of the 3D world with information gathered from

multiple sensors. In this chapter, we focus on intersection/lane geometry and topology in the BEV domain when the inputs are from multiple cameras. A local lane map representation is firstly proposed. Then, based on different sensor fusion methodologies, we design multiple variants of the BEV segmentation network to infer the local lane map. With experimental outcomes, a network with pre-fused BEV image input supervised in the BEV domain is concluded to be the preferred strategy for lane-level information extraction. The proposed method is tested in both simulated and real-world environments, showing the effectiveness of the chosen strategy in BEV sensor fusion tasks.

Chapter 7 Multi-task Learning, Mapping with Perception

Detecting dynamic objects and predicting static road information such as drivable areas and ground heights are crucial for safe autonomous driving. Previous works studied each perception task separately and lacked a collective quantitative analysis. In this chapter, we show that it is possible to perform all perception tasks via a simple and efficient multi-task network. The proposed network, LidarMTL, takes raw LIDAR point cloud as input and predicts six perception outputs for 3D object detection and road understanding. The network is based on an encoder-decoder architecture with 3D sparse convolution and deconvolution operations. Extensive experiments verify the proposed method with competitive accuracy compared to state-of-the-art object detectors and other task-specific networks.

Part III HD Map Management

With maps constructed, the next step in the HD map life cycle is for autonomous vehicles to efficiently use these maps for downstream tasks. In Chapter 8, an efficient submap query-stitching strategy is proposed for the real-time deployment of the HD map.

Chapter 8 Submap Query and Stitching

As demonstrated in previous sections, HD maps are extremely useful for downstream tasks in autonomous driving. However, for individual vehicles, how to efficiently get access to these pre-made maps is not trivial. Directly loading the complete map onto the onboard storage device is infeasible, because the HD maps could occupy a large amount of the memory space. In this chapter, the concept of a submap is introduced to dissect a large urban map into smaller tiles, and a stitching algorithm is proposed to fuse neighboring maps to guarantee local consistency. The patching algorithm is evaluated with Simultaneous Localization and Mapping tasks, showing an effective querying and matching operation in real-time.

Part I

Mapping Platform and Data

Chapter 2

Joint Spatial-Temporal Calibration

Sensors are essential tools for robotic systems to perceive the physical world. Naturally, the first step in the development of an HD map is to prepare a sensor suite that is suitable for the environment. Figure 2.1 shows a typical mapping vehicle that is used by the Mechanical Systems Control Lab at UC Berkeley [157]. On the top of the vehicle is a sensor suite that contains a LIDAR, six cameras, an Inertial Measurement Unit, and several GPS antennas. The sensor suite is designed to fully explore the complementary nature of these sensors. For example, LIDAR point clouds provide accurate yet sparse 3D measurements, and camera images yield dense semantics with no sense of depth. As a result, researchers would like to jointly use these sensors for a complete representation of the scene.

For most sensor fusion algorithms, however, additional information about the spatial-temporal relationships among these sensor modalities is needed. More specifically, algorithms usually require an accurate geometric relationship among multiple sensors to perform

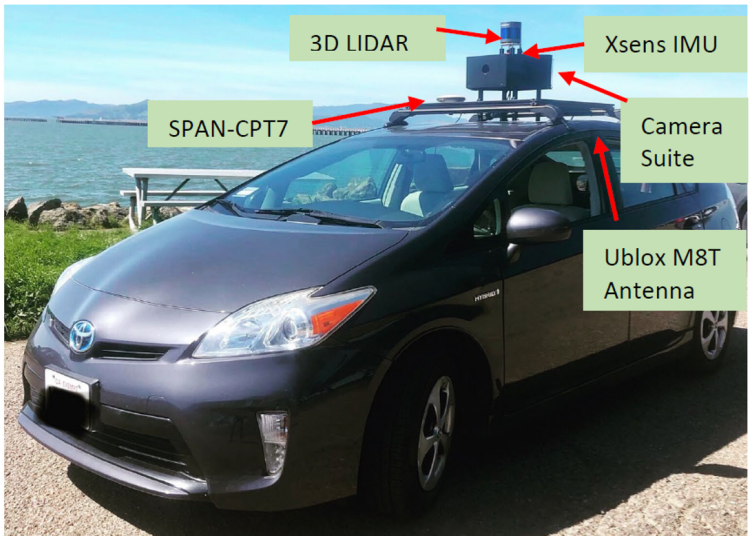


Figure 2.1: An example of the mapping sensor suite

coordinate transformations, and it is often assumed that the contents from these sensors are captured at the same time. Thus, a complete calibration of the sensor suite is needed.

Promoted by such demand, this chapter focuses on the joint spatial-temporal calibration of a LIDAR-camera sensor suite. Here, a semantic segmentation-based method is proposed to minimize the newly designed bi-directional loss in the image domain. Temporal parameters are estimated from visual odometry prediction between two consecutive frames. Since supervision is only performed at the segmentation level, no calibration label is needed within the framework. The method is implemented on the KITTI [38] dataset, and the results show a robust and accurate calibration of the LIDAR-camera sensor suite[1].

2.1 Introduction

Robots nowadays are usually equipped with multiple types of sensors to ensure a comprehensive perception of the scene. Different sensors on a robot provide observations with their unique physical characteristics. For mobile robots in particular, cameras and LIDARs are among the most popular choices in perception tasks. In outdoor scenes, camera images provide semantically rich representation in dense 2D formats without any depth data, and LIDARs offer accurate yet sparse 3D measurements as point clouds.

The complementary nature of these two types of sensors inspires the designs of fusion algorithms for detection tasks. In practice, the fusion algorithms are proved to out-perform single-modal methods with a significant margin [165]. More importantly, since multiple sensors are located differently on the mapping vehicle, a complete sensor suite can often overcome potential occlusion problems in a single sensor setup.

One of the fundamental elements in sensor fusion algorithms is calibration [165]. To begin with, the accurate geometric relationship between two sensor modalities is required for most fusion algorithms [171, 139, 152, 135, 117]. More specifically, three translation and three rotation parameters (six degrees of freedom or 6-DOF) need to be provided as inputs. However, acquiring such geometrical relationships is challenging. Human labeling, artificial rigs, and manual measurements are often involved in some earlier attempts, and such processes are usually time-consuming. More detrimentally is the irreproducible nature of the one-time calibration: whenever the vehicle experiences vibration or collision, the geometric data from manual calibration is voided. Thus, automatically calibrating the geometry is critical for any onboard sensor suite. In recent years, a series of learning-based methods start to predict the extrinsic parameters of the sensor suite. These methods are usually data hunger, requiring pre-labeled data as supervision to predict the parameters in the target sensor setting [128, 55, 85]. However, the ground truth calibration data is still hard to obtain and none of these works reported the generalization capacity on other datasets. Some other methods leverage the semantic outcome of each sensor modality and directly regress the extrinsic matrices without supervision. This work follows on SemAlign [83] for finer geometric calibration.

[1]This chapter includes materials from the author's previous work [66]

Beyond the geometric parameter estimation, the temporal relationship among sensors also requires researchers' attention. More specifically, how to synchronize each sensor remains to be a long-neglected problem in the autonomous driving research domain. Algorithms usually assume naively that the content in each sensor frame is acquired at precisely the same time, but this assumption rarely holds. For example, in the Argoverse dataset [14], the synchronization is performed by simply associating the closest timestamp between LIDARs and cameras. Some other works [157] perform synchronization at the data acquisition level, but the static nature of a one-time delay compensation could not address the changes in the real-world deployment. Thus, dynamically calibrating the time delay among sensors is critical, yet almost unexplored.

This chapter proposes a joint spatial-temporal calibration framework between LIDARs and cameras on an autonomous driving platform. The input of the proposed framework are sequences of camera and LIDAR frames. Here, each sensor modality is processed through an arbitrary semantic segmentation network, which one can choose based on the available training data. Secondly, the segmented LIDAR point cloud is projected onto the semantic image, where a newly designed bi-directional alignment loss is calculated for geometrical parameter regression. Not limited to point-to-pixel loss, we downsample the semantic pixel for pixel-to-point loss as well. To estimate the time delay between sensor modalities, we estimate the visual odometry from two consecutive images and predict a shifted point cloud for matching.

The joint spatial-temporal calibration framework is tested on the KITTI dataset for 150 frames, and the proposed method can robustly regress geometrical and temporal relationships among LIDARs and cameras. Ablation studies are also included in this chapter to demonstrate the bi-directional loss superiority and the strong temporal estimation capacity.

The major contributions of the work included in this chapter are:

- A joint spatial-temporal calibration algorithm is proposed for LIDAR-camera sensor suite;
- A bi-directional loss is designed for more robust performance in geometrical parameter regression;
- A time parameter is coupled with visual odometry to estimate the temporal delay among sensors.

Geometric Calibration

Rig-based Calibration

The earliest attempts in geometric sensor calibration start with artificial targets, also known as rigs. Such objects are often of regular shapes or specific patterns, making them easy for simple detection algorithms.

In Single-Shot Calibration, Geiger et al. propose to use checkerboards as targets for the calibration task between cameras and range sensors [39]. The patterns on the checkerboard are easy to identify for cameras, and the board themselves are easily detectable planes for range sensors. A downstream optimization process is then carried out for the alignment. Following a similar pipeline, other methods use different rigs or features for such calibration tasks [65, 69, 71]. However, constructing an artificial rig could consume a significant amount of time, and such a one-time calibration result is often not repeatable. For example, if the sensor suite is changed geometrically in accidents, a new round of rig calibration is required.

Feature-based Calibration

Departing from the calibration board, some other methods explore the possibilities of finding various features in the real world for calibration.

Exploring the geometric features in the environment inspires the authors of [59] and [100] to use edges and lines in indoor scenes to estimate the 6-DOF of a sensor suite. Tamas et al., on the other hand, use the plane feature in outdoor environments [140]. However, these features are not universally available in all scenes, and advanced detection/correspondence algorithms are often needed. Another school of thought is to utilize the color channels in images and the intensity channels in LIDARs to maximize the mutual information for calibration purposes [111]. However, depending on the viewing angle, the intensity return from the LIDAR could be drastically different from the image outcomes [145].

End-to-end Calibration

More recently, some learning-based methods predict the extrinsic parameters of the sensor suite with raw sensor inputs [128, 55, 85]. To train these prediction models, researchers usually need to acquire a large dataset with ground-truth calibration data. More importantly, none of the aforementioned works evaluate the models' generalization capacity on other datasets.

Last year, SemAlign was proposed as a semantic segmentation-based extrinsic calibration method [83]. The model utilizes off-the-shelf segmentation networks for LIDARs and cameras, and it optimizes the geometric calibration parameters for minimum Semantic Alignment loss. Such a method is favorable because the generalization capacity lies within the semantic segmentation modules, which are well explored-domains in computer vision. However, the single-direction loss function in this work is not robust enough for complicated scenes.

Temporal Calibration

Sensor synchronization is often overlooked in driving-related tasks [165], and most algorithms often naively assume that the data provided is already synchronized. However, such assumptions rarely hold. To synchronize the sensors, researchers designed both offline and

online mechanisms for synchronization. Here we introduce a few existing attempts related to temporal calibration.

Offline Synchronization

In offline designs, the synchronization is performed before the data acquisition. Similar to the triggering mechanism in multi-cameras setups in the early days of photography, a naive attempt in sensor synchronization is to use hardware triggers [15] like a fix-frequency impulse signal. Some other datasets prepare software triggers with compensated triggering time [157], but the manual calibration of the time delay is usually time-consuming. More recently, the IEEE1588 Precision Time Protocol is also introduced in system-wide synchronization applications.

Online Synchronization

Online synchronization happens during or after the data acquisition process. A typical example is the Argoverse dataset[14], where each camera image is matched to the closest LIDAR scan, while the physical sensor themselves are not exactly synchronized. Most research work in this domain estimates the time delay between sensors while they capture real-world data. VINS calibrates the time delay between the high-frequency Inertia Measurement Unit (IMU) and a camera by optimizing the trajectory consistency between these two sensors [122].

Joint Calibration

Calibration through Simultaneously Localization and Mapping (SLAM) is a popular method for the joint calibration task. Through SLAM algorithms, each sensor modality provides a unique odometry that is both spatial and temporal dependent. In [112] and [143], the authors utilize such odometry estimation to find the best spatial-temporal match between the two generated trajectories. However, the aforementioned methods leverage the performance of the SLAM algorithms over the whole traveling trajectory, and SLAM's drifting effect could easily deteriorate the performance of the downstream calibration task. In our proposed work, we optimize the extrinsic and the temporal parameter iteratively, avoiding error accumulation along the traveling path. Furthermore, Persic et al. propose to use tracking results from both sensor modalities to solve for geometric and temporal calibration parameters [116]. Similarly, the calibration performance is also limited by the accumulated error in tracked objects.

2.2 Full Calibration of LIDAR-Camera Sensor Suite

The workflow of the proposed calibration method is shown in Figure 2.2. The calibration process consists of the static spatial parameter calibration module for the spatial initial guess and the joint spatial-temporal parameter calibration module for duo-parameter estimation.

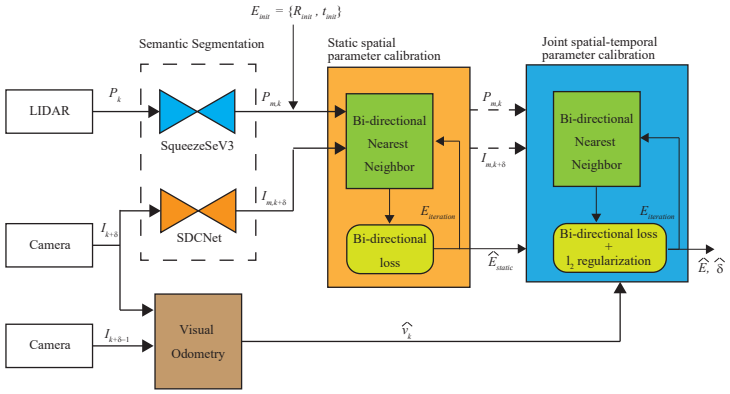


Figure 2.2: Workflow of the proposed calibration method

The inputs of the proposed algorithm are one point cloud scan P_k, and two sequential RGB images $\{I_{k+\delta}, I_{k+\delta-1}\}$. The goal of the algorithm is to estimate a 6-DOF $E = \{R, \boldsymbol{t}\}$ of the geometric relationship and the time delay δ between P_k and $I_{k+\delta}$.

To do this, we first process P_k and $I_{k+\delta}$ through arbitrary semantic segmentation algorithm to obtain semantic mask $P_{m,k}$, $I_{m,k+\delta}$. Then with initial extrinsic guess $E_{init} = \{R_{init}, \boldsymbol{t}_{init}\}$ from either rough measurement or sampling and the known intrinsic matrix K, we project the LIDAR point cloud onto the camera image plane. By finding the nearest neighbor both from point to pixel and from pixel to point, we calculate the euclidean distance between them to form the bi-directional loss function of the optimization algorithm.

The first optimization iteration (the static spatial parameter calibration module) will be carried out at the frame when the ego vehicle's velocity is almost 0. The static spatial parameter calibration gives the initial estimation of the rotation and translation $\hat{E}_{static} = \{\hat{R}_{static}, \hat{\boldsymbol{t}}_{static}\}$. This estimation will be used as the initial guess and the regularization reference for the joint spatial-temporal parameter calibration. We include the static iteration to avoid local minimum during the joint calibration processes.

Secondly, for dynamic scenes, we estimate the temporal information between $I_{k+\delta}$ and $I_{k+\delta-1}$ from the visual odometry which will predict the velocity $\hat{\boldsymbol{v}}_k$ between two camera frames. Here, the translation shift between P_k and $I_{k+\delta}$ can be represented as $\boldsymbol{t}_{\delta,k} = \hat{\boldsymbol{v}}_k \cdot \delta$. We use this $\hat{\boldsymbol{v}}_k$ as part of the optimization and estimate both the time delay $\hat{\delta}$, and $\hat{E} = \{\hat{R}, \hat{\boldsymbol{t}}\}$.

Semantic Segmentation

With off-the-shelf semantic segmentation modules, the proposed method would generalize to any dataset with semantic labels. In this chapter, we use SqueezeSegV3[158] and SDCnet[123] trained on KITTI dataset for point cloud and image semantic segmentation, respectively. Considering the frequent presence of vehicles in the urban environment, in this work we only use the car class for semantic segmentation. Applying these semantic segmentation modules to the input, we get semantic masks $P_{m,k}$, $I_{m,k+\delta}$.

Point Cloud Projection

In order to compute the semantic loss, we first project the semantic mask of a point $p_{i,m,k} \in P_{m,k}$ onto the 2D image plane. Based on the classic camera model [87], we can achieve the projection as

$$\begin{bmatrix} px_{i,m,k} \\ py_{i,m,k} \\ pz_{i,m,k} \end{bmatrix} = K[R\boldsymbol{p}_{i,m,k} + \boldsymbol{t}], \tag{2.1}$$

$$\bar{\boldsymbol{p}}_{i,m,k} = \begin{bmatrix} pu_{i,m,k} \\ \\ pv_{i,m,k} \end{bmatrix} = \frac{1}{pz_{i,m,k}} \begin{bmatrix} px_{i,m,k} \\ \\ py_{i,m,k} \end{bmatrix}, \tag{2.2}$$

where $pu_{i,m,k}$ and $pv_{i,m,k}$ are the image coordinates of the projected point.

Bi-directional loss

Let $\bar{\boldsymbol{p}}_{1,m,k} \cdots \bar{\boldsymbol{p}}_{n_p,m,k}$ be a set of projected LIDAR points that are within the camera's field of view. Now for the projected point $\bar{\boldsymbol{p}}_{i,m,k}$, let $\boldsymbol{q}_{j,m,k+\delta} \in I_{m,k+\delta}$ be the nearest neighbour pixel of the same class. Then, the single-direction point-to-pixel (point-to-image) semantic alignment loss on frame k can be computed as

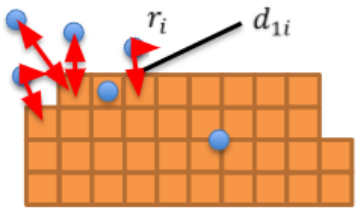
$$L_{p2i,k} = \sum_i ||\bar{\boldsymbol{p}}_{i,m,k}(R, t, \boldsymbol{p}_{i,m,k}) - \boldsymbol{q}_{j,m,k+\delta}||_2^2. \tag{2.3}$$

Here, the loss is computed per projected point. Figure 2.3a illustrates the point-to-pixel calculation process. As shown in [83], by minimizing this loss function, we can make the projected point cloud well overlapped with the pixels that have same semantic label. Thus, minimizing this loss function could lead us to the correct estimation for $\hat{E}_{static} = \{\hat{R}_{static}, \hat{\boldsymbol{t}}_{static}\}$.

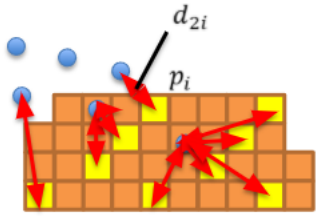
However, when the initial guess for the extrinsic matrix is significantly different from the ground truth, the nearest neighbor matching does not necessarily give us the appropriately matching result for most of the pairs, and the information of some important pixels will be abandoned. Thus, minimizing the single-directional loss would fall into the inappropriate local minimum.

To avoid such loss of information, we propose a bi-directional loss that utilizes the pixel-to-point (image-to-point) nearest neighbor matching as well (Figure 2.3b).

Considering the fact that one image has too many pixels to match in real time, we downsample the pixels for the pixel-to-point matching. Let $\{\bar{\boldsymbol{q}}_{1,m,k+\delta} \cdots \bar{\boldsymbol{q}}_{n_i,m,k+\delta}\} \subset I_{m,k+\delta}$ be a set of the downsampled pixels. Now for the pixel $\bar{\boldsymbol{q}}_{i,m,k+\delta}$, $\bar{\boldsymbol{p}}_{j,m,k} \in P_{m,k}$ is the nearest neighbor projected point. Then, the pixel-to-point semantic alignment loss on frame k can be computed as



(a) Point-cloud to Pixels



(b) Pixels to Point-cloud

Figure 2.3: Bi-directional projection demonstration

$$L_{i2p,k} = \sum_i ||\bar{\boldsymbol{q}}_{i,m,k+\delta} - \bar{\boldsymbol{p}}_{j,m,k}(R,t)||_2^2. \tag{2.4}$$

Here, the loss is computed per sampled pixel. Then, the bi-directional semantic alignment loss at the l-th iteration can be represented as

$$L_{bi,k,l} = L_{p2i,k,l} + w_l \cdot \frac{n_p}{n_i} \cdot L_{i2p,k,l}, \tag{2.5}$$

where $\frac{n_p}{n_i}$ is the normalization term and w_l is the weight at the optimization iteration number l. With smaller w_l, the optimizer tends to align the projected points within the image mask to minimize $L_{p2i,k,l}$. With larger w_l for minimizing $L_{i2p,k,l}.$, the optimizer tends to have the image mask well included in the projected point cluster. Thus, shifting the value of w_l during optimization iterations can avoid getting stuck in local minimum, and the optimization solution would lead us to a better nearest neighbor matching for the next iteration. Thus, optimizing the bi-directional loss function would yield a more refined guess $\hat{E}_{static} = \{\hat{R}_{static}, \hat{\boldsymbol{t}}_{static}\}$ for the joint calibration.

Joint Spatial-Temporal calibration

To estimate the temporal parameter, we extract the velocity $\hat{\boldsymbol{v}}_k$ between two sequential RGB images $\{I_{k+\delta}, I_{k+\delta-1}\}$ using visual odometry. The visual odometry used in this chapter is based on the sparse optical flow for FAST feature tracking [84], and Nister's 5-point algorithm with RANSAC [106] for essential matrix estimation.

With a moving ego vehicle and an asynchronized sensor suite, the projected point cloud obtained from Equation 2.1 will never match with the corresponding pixels, even with the ground truth geometric calibration parameters. To compensate for this time delay, we need to modify the projection equation as

$$\begin{bmatrix} px_{i,m,k,\delta} \\ py_{i,m,k,\delta} \\ pz_{i,m,k,\delta} \end{bmatrix} = K[\hat{R}_{static}\boldsymbol{p}_{i,m,k} + \hat{\boldsymbol{t}}_{static} + \hat{\boldsymbol{v}}_k \cdot \hat{\delta}], \tag{2.6}$$

$$\bar{\boldsymbol{p}}_{i,m,k,\hat{\delta}} = \begin{bmatrix} pu_{i,m,k,\hat{\delta}} \\ \\ pv_{i,m,k,\hat{\delta}} \end{bmatrix} = \frac{1}{pz_{i,m,k,\hat{\delta}}} \begin{bmatrix} px_{i,m,k,\hat{\delta}} \\ \\ py_{i,m,k,\hat{\delta}} \end{bmatrix}. \tag{2.7}$$

Here, $pu_{i,m,k,\hat{\delta}}$ and $pv_{i,m,k,\hat{\delta}}$ are the image coordinate of the projected point compensated with $\hat{\delta}$ and $\hat{\boldsymbol{v}}_k$. Therefore, we can estimate both spatial and temporal parameters by minimizing the modified bi-directional loss at iteration l:

$$L_{bi,k,\hat{\delta},l} = L_{p2i,k,\hat{\delta},l} + w_l \cdot \frac{n_p}{n_i} \cdot L_{i2p,k,\hat{\delta},l} + \beta, \tag{2.8}$$

$$\beta = \lambda_1 ||\hat{t} - \hat{t}_{static}||_2^2 + \lambda_2 ||\hat{R}\hat{R}_{static}^{-1}||_2^2. \tag{2.9}$$

Here, β is the regularization term bringing the estimation closer to initial guesses. λ_1 and λ_2 are the regularization coefficients for translation and rotation respectively.

2.3 Implementation and Experiment

In this section, we report our experiment with the proposed algorithm and related works. After introducing the implementation details, we report our algorithm performance on the joint LIDAR-camera calibration. Furthermore, we show the experiment result of each calibration module under various noise levels in both geometrical and temporal sense. In the end, we include an ablation study on the loss formulation.

Implementation Details

Each driving sequence in KITTI [38] includes RGB images, LIDAR point clouds, and accurate extrinsic/intrinsic calibration parameters. We treat intrinsic calibration parameters as

given and assume the provided extrinsic parameters as the ground truth. We use Sequence 00 (4541 frames) for evaluation on both static and dynamic calibration.

For static calibration, we use $w_l = 20$ for the first 20 optimization iterations, $w_l = 1$ for the next 30 iterations, and $w_l = 0.02$ for the last 10 iterations. The iterative optimization has 60 iterations in total. For the joint calibration part, we use constant weight ratio $w = 5$ for 20 iterations in total. The regularization coefficient we use are $\lambda_1 = 10^6$ and $\lambda_2 = 10^9$. The downsampling rate of pixels for the pixel-to-point matching is 2%.

We use quaternion angle difference (QAD) and the average euler angle difference (AEAD) [83] to evaluate the rotational error between estimated rotations and the ground truth rotation. Both angle differences are computed as

$$QAD = 2 \arccos(|p \cdot q|), \tag{2.10}$$

$$AEAD = (R_{error} + P_{error} + Y_{error})/3, \tag{2.11}$$

where p and q are ground truth rotation and estimated rotation respectively. R_{error}, P_{error}, Y_{error}, denote the absolute Euler angle difference of roll, pitch, and yaw.

ATD (Average Translation Difference) is used to evaluate the absolute translation error. The ATD is calculated as

$$ATD = (x_{error} + y_{error} + z_{error})/3, \tag{2.12}$$

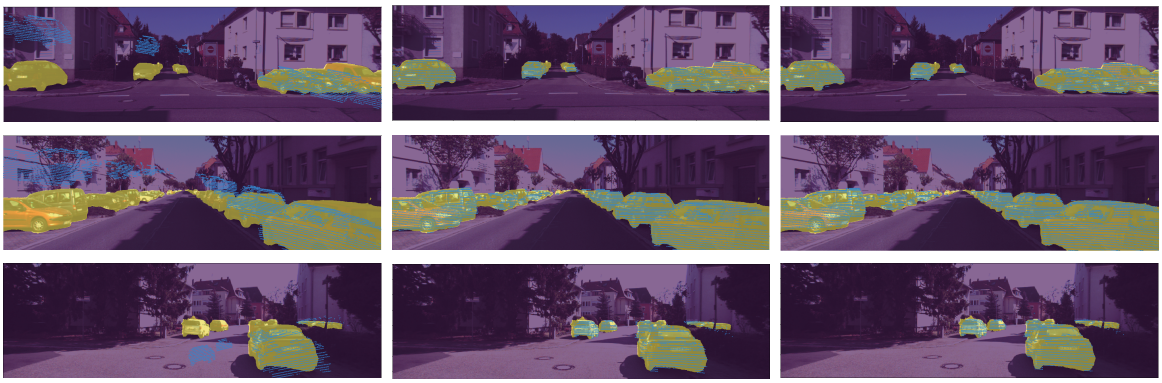
where x_{error}, y_{error}, and z_{error}, represent the absolute translation error along the x, y, and z-axis.

Joint Spatial-Temporal Calibration

In this experiment, it is assumed that the LIDAR and the camera are not synchronized, meaning that there is a certain amount of time delay for each sensor's data acquisition trigger. To exacerbate this delay using the KITTI odometry dataset, we intentionally use RGB image data acquired one frame ahead of the LIDAR point cloud. In the KITTI odometry dataset, both the RGB image and the point cloud are captured every 100 ms, meaning that the data acquisition frequency of both the camera and the LIDAR is 10 Hz. Therefore, the ground truth of the time delay in this simulation is around 100 ms. The initial guesses of translation for each x, y, z-axis are sampled from the uniform distribution [-10 cm, 10 cm] away from the ground truth. The rotation guesses for roll, pitch, yaw are sampled from a uniform distribution [-10 deg., 10 deg.] away from the ground truth.

First, we use Frame 547 to 552 where the car is not moving for the static calibration part. The estimated spatial calibration result is then fed to the joint calibration algorithm to estimate the temporal parameter.

We sample 50 frames for joint calibration evaluation. A visualization of the experimental results could be seen in Figure 2.4. The blue points are the segmented LIDAR point clouds



(a) Initial guess (b) After SST-Calib (c) Ground truth

Figure 2.4: Calibration results on the asynchronized KITTI odometry dataset

Table 2.1: Joint spatial-temporal calibration results

	Delay δ [ms]	Average delay error [ms]	ATD [cm]		QAD [deg.]		AEAD [deg.]	
			Mean	Median	Mean	Median	Mean	Median
Taylor et al.[143]	100	10.0	5.27	-	-	-	0.34	-
Park et al.[112]	10	3.5	20.00	-	-	-	0.88	-
SSTCalib (ours)	100	3.4	7.45	5.53	0.67	0.66	0.32	0.32

projected with the corresponding extrinsic matrix. The yellow masks are the segmentation result for vehicles in the image. The quantitative results are shown in Table 2.1. Our proposed method estimates the time delay as 96.6 ms, only 3.4 ms different from the ground truth. While appropriately estimating the temporal parameters, the spatial parameters estimation also achieves the same level of performance compared with the state-of-the-art spatial calibration methods.

Robustness Study: Calibration under Noises

The robustness of our proposed method using the bi-directional loss is demonstrated in this section. We sample 150 frames from sequence 00 with random translation and rotation initial guesses. The initial guess noise for the translation is uniformly distributed within [-10 cm, 10 cm], and the initial guess noise for the rotation is uniformly distributed within [-10 deg., 10 deg.] and [-20 deg., 20 deg.]. In Table 2.2, we observe that our proposed method has the similar level of spatial parameters estimation capability without any costly pre-process. For both rotational noise distribution, our method reports better AEAD results compared to the SemAlign, which samples 5000 random transformations that may lead to better initial guesses. More importantly, when the noise level doubles, our proposed algorithm maintains a similar level of performance.

We also test the algorithm performance under different time delay intervals. Table 2.3

Table 2.2: Static spatial calibration results

	Pre-process	Initial rotation	ATD [cm]		QAD [deg.]		AEAD [deg.]	
			Mean	Median	Mean	Median	Mean	Median
SemAlign[83]	Sampling 5000 E_{init} for initialization	[-10, 10]	-	-	1.14	0.46	0.62	0.23
		[-20, 20]	-	-	2.59	0.49	1.49	0.24
SSTCalib (Ours)	None	[-10, 10]	18.9	12.8	1.28	0.81	0.60	0.38
		[-20, 20]	20.2	20.0	1.53	1.19	0.69	0.59

Table 2.3: Impact of the time delay magnitude

Time delay [ms]	Time delay average error [ms]	ATD [cm]		QAD [deg.]		AEAD [deg.]	
		Mean	Median	Mean	Median	Mean	Median
100	3.4	7.4	5.5	0.67	0.66	0.32	0.32
200	13.5	6.5	6.8	0.68	0.69	0.34	0.34
300	23.3	4.6	4.9	0.7	0.67	0.34	0.33

Table 2.4: Comparison between single- and bi-direction loss, an ablation study

	Initial rotation	ATD [cm]		QAD [deg.]		AEAD [deg.]		Failure rate [%]
		Mean	Median	Mean	Median	Mean	Median	
Single-direction	[-10, 10]	42.7	22.4	3.67	1.62	1.63	0.79	16.2
Bi-direction	[-10, 10]	18.9	12.8	1.28	0.81	0.60	0.38	8.8

shows a result of our experiments with time interval of 100, 200, and 300 milliseconds. The proposed algorithm shows a successful estimation of both geometric and temporal parameters.

Ablation Study: Bi-directional Loss

As a contribution of the proposed algorithm, we argue that the bi-directional semantic loss is superior to the single-directional loss. The semantic calibration algorithm using single-direction loss is similar to SemAlign [83], omitting the random pre-sampling process. We report the calibration result of these two loss functions in Table 2.4. Here, all error metrics of the single-directional loss are approximately twice as large as the ones of the bi-direction loss. More importantly, the single-directional loss formulation fails more frequently at the optimization phase, meaning that the point-to-pixel loss alone is not robust for complicated scenes.

2.4 Chapter Summary

This chapter starts the HD Map development journey at its very first step: the sensor suite calibration. More specifically, the complementary LIDAR-camera suite is studied for complete spatial-temporal calibration.

The proposed calibration method aims to optimize both geometric and temporal parameters with an accentuation on the generalization capacity. Here, the data from both sensor modalities are firstly fed through arbitrary semantic segmentation networks, and a bi-directional loss is calculated on the projected semantic image. To incorporate the temporal elements, visual odometry is estimated to compensate for the asynchronized trigger between sensor modalities. The proposed method is evaluated on the KITTI dataset, the experiment result and related ablation studies demonstrate an accurate and robust calibration of the sensor suite.

With well-calibrated sensors, the next step in the HD map development is to prepare a full sensor-suite dataset for urban autonomous driving.

Chapter 3

Urbanloco: the Urban Mapping Dataset

Data is the foundation for most artificial intelligence systems nowadays. For mapping and localization applications, in particular, collecting data is one of the first few steps in the development process.

However, constructing a dataset is not usually easy. First, a complete sensor suite is usually needed for mapping purposes. Beyond Global Navigation Satellite System (GNSS), researchers in the mapping field also work on point cloud registration, visual feature matching, and inertia navigation. Thus, sensors like LIDARs, cameras and Inertial Measurement Units (IMU) are needed on the mapping platform. Furthermore, to push the limit of the mapping algorithms, it is important for the dataset to include some of the most challenging scenes. Highly urbanized areas fall well within this challenging zone: LIDAR- and camera-based methods perform poorly with numerous dynamic objects; the GNSS-based solutions experience signal loss and multipath problems; the IMUs suffer from drifting. Researchers in both academia and industry are eager to see challenging public datasets with complete sensor suites.

In this chapter, a sample mapping dataset UrbanLoco is presented. With Simultaneously Localization and Mapping (SLAM) applications in mind, this dataset is constructed with multiple sensor modalities including cameras, LIDARs, IMUs, and the GPS. The data is collected in densely urbanized areas like San Francisco and Hong Kong to showcase the challenging city environments. A few state-of-the-art algorithms are benchmarked on the dataset, and the result shows that urban SLAM is still an unsolved problem for the research community. Currently, UrbanLoco is open to the public for research and education purposes [1,2].

[1] `https://advdataset2019.wixsite.com/urbanloco`

[2] This chapter includes materials from the author's previously published work [157]

Figure 3.1: An overview of the UrbanLoco dataset

3.1 Introduction

Publicly available datasets have been hugely influential in autonomous driving research, both in academia and in industries. In the past few years, numerous datasets were published to advance different aspects of autonomous driving. One of the pioneers in academia is the KITTI dataset [37], which covers most topics in current research and launches various benchmarks for evaluation. Recently, a few companies, including Lyft [51], nuTonomy [9], Waymo [154], and Argo AI [14], publish their datasets for detection and prediction. However, few of these public datasets address the urban mapping and localization problem, which is yet to be solved with a cost-effective sensor combination.

The need for urban datasets comes from the challenging nature of the urban scenes. Due to high-rise structures and numerous dynamic objects, mapping and localization in a densely populated area are hard, and simultaneous localization and mapping (SLAM)[43] in a city is even harder. Figure 3.1 shows an overview of the city landscape in our dataset. The traditional Global Navigation Satellite System (GNSS) based solution fails in urban canyons due to limited satellite visibility and multipath problems. For more recent LIDAR- or vision-based approaches, the dynamic objects (vehicles, pedestrians, cyclists) may cause inaccurate point registration. While the Inertial Measurement Units are less affected by urban environments, they suffer significantly from the drifting effect over time.

With the aforementioned challenges, however, most of the current public datasets are not specifically targeted at the mapping and localization tasks in city applications. As densely populated scenes are unavoidable for autonomous driving, it is urgent to provide the public with a corresponding dataset. Here we present the UrbanLoco: a full sensor suite dataset for mapping and localization in densely populated landscapes. The dataset includes information from 4 essential sensor modalities: LIDAR, camera, IMU, and GNSS. The data is collected in the populous districts in Hong Kong and San Francisco.

The major contributions of the work included in this chapter are:

- We are releasing the first large scale full sensor suite dataset focusing on urban mapping and localization challenges;
- The dataset includes over 40 kilometers of travel, covering various driving scenarios including urban canyons, bridges, hills, tunnels, etc. There are also numerous dynamic objects in the scene;
- The dataset provides information from a full sensor-suite: one LIDAR, six cameras with 360 degrees views (one camera in Hong Kong), one IMU, and one GNSS. The sensor-suite is carefully calibrated with calibration logs available online;
- We define the urbanization measure for localization in city landscapes and evaluate the urbanization rate on current mapping/localization datasets;
- The dataset is publicly available through the project website, and related APIs are also available for users' convenience.

Autonomous Driving Datasets Review

Autonomous Driving Datasets for Mapping and Localization

Mapping/localization-focused datasets and benchmarks have been heavily used by current researchers. KITTI [37] odometry is a popular benchmark collected in the German town of Karlsruhe. Most of the trajectories are long enough ($\geq$ 500 m) to test outdoor localization algorithms. A total of 20 trips have been the popular test field for the latest algorithms. Indeed, the KITTI odometry benchmark lead board is still updated nowadays. Unfortunately, the KITTI dataset does not pose enough challenges for mapping and localization: the data is collected in rural areas with light traffic and relatively low-lying structures. Furthermore, KITTI only provides front-view cameras and LIDAR information, lacking the critical GNSS and IMU outputs. Oxford RobotCar [88] is another popular dataset that specializes in mapping and localization. Unlike KITTI, the Oxford RobotCar dataset offers extra information from IMU and GPS measurements, which provides more possibilities for testings of various algorithm designs. However, the data is collected in the less-urbanized Oxford, not adequately addressing the urban mapping and localization problems. Table 3.1 shows a comparison of the aforementioned two datasets with our dataset. We would further quantify the urbanization rate specifically for localization in the next section.

Autonomous Driving Datasets for Urban Scenes

Until recently, it is rare to see autonomous driving datasets in highly-urbanized scenarios. In early 2019, the Boston/Singapore-based NuScenes dataset [9] was published. Similar to our dataset, the NuScenes data is collected among dense traffics and high-rise structures. More importantly, the NuScenes dataset also incorporates visual information and IMU. However, since the NuScenes dataset is majorly designed for perception and tracking, the data is

segmented into scenes of 20 seconds in length. Thus, it is hard to acquire a sufficiently long trajectory for mapping and localization purposes. Later in 2019, Lyft [51], Waymo [154], and Argo AI [14] published their datasets that were targeted at detection and prediction tasks as well. Thus, they are similarly segmented into discrete pieces. Furthermore, these datasets do not include IMU and GPS information. Table 3.1 shows a comparison of different public datasets for autonomous driving.

Table 3.1: Datasets comparison

Dataset	Location	Urbanization rate	Distance (Max path)	LIDAR	Camera	IMU	GNSS
KITTI[37]	Karslruhe	Low	1km	✔	front	✔	✘
RobotCar[88]	Oxford	Low	1km	✔	front	✔	✘
KAIST [56]	Seoul	High	11.42km	Tilted	front	✔	✔
NuScenes[9]	Singapore, Boston	High	20s	✔	360	✔	✔
Waymo[154]	SF Bay Area	High	20s	✔	360	✘	✘
Lyft[51]	San Francisco	High	20s	✔	360	✘	✘
Argoverse[14]	Miami, Pittsburgh	Mid	30s	✔	360	✘	✘
Ours	San Francisco Hong Kong	High	13.8 km	✔	360	✔	✔

Geometric Mapping and Localization Algorithms

GNSS Based Localization

GNSS is widely used as a convenient tool for localization tasks. Equipped with a GNSS receiver, the ego vehicle could easily navigate in regions where GNSS receptions and satellite visibility are satisfactory. However, such requirements are rarely fulfilled in urban scenarios. Indeed, GNSS manufacturers are incorporating IMU with GNSS receivers to continuously navigate when the satellites are lost insight. However, such equipment is usually expensive, and could not be used commercially for mass application.

Vision-based Mapping and Localization

Stereo cameras are used to extract spatial information about a scene [105, 90], but it is hard for a monocular camera to solve the general motion. To complement the lack of depth information for monocular cameras, RGB-D cameras are utilized successfully for SLAMs in different scenes [54, 49, 64]. However, the depth perception span is very limited compared to LIDARs. Another sensor fusion method for monocular navigation is the Visual Inertial Navigation System (VINS-MONO) [121], which tightly couples a monocular camera along with an IMU for localization. A nonlinear optimization algorithm is applied to incorporate the loss in IMU measurement as well as the loss in visual feature matching. This specific algorithm is evaluated in Section 3.4.

Laser-based Mapping and Localization

Laser odometry is another significant branch of SLAM research. Point cloud registration algorithms are the cornerstones for laser-based methods: ICP and its multiple variations [7, 168, 129] are among the earlier attempts in odometry estimation. Other algorithms like the Normal Distribution Transformation (NDT) [89] and the Mix-Norm Matching (MiNoM) [151] also improve the quality of pure laser odometry estimation. The LIDAR Odometry and Mapping method (LOAM) [167] extracts surface and corner features in the scan to determine distances in a grid-based voxel map. In this chapter, we present the evaluation of the performances of LOAM and NDT on our dataset.

3.2 The Urbanization Measure

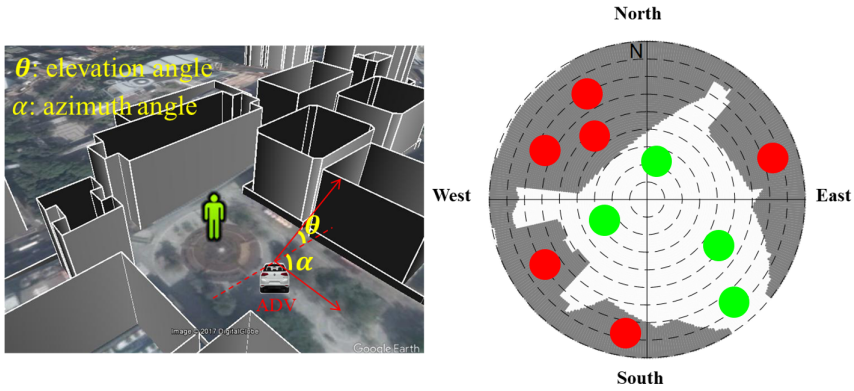
In this section, we introduce an urbanization measure for localization challenges in different scenes. This metric is further used to quantitatively compare our dataset with other publicly available ones.

While GNSS is an intuitive and cost-effective solution for localization challenges, its performance deteriorates in dense urban areas. One major reason for the poor performance is the high-rise structures in the urban landscapes: skyscrapers in a city could block satellite signals, resulting in a limited number of visible satellites for positioning. Moreover, the GNSS signal can be reflected by building surfaces in urban canyons, introducing an extra traveling path for the GNSS time measurement. The reflected GNSS signal would induce the multipath effect and non-line-of-sight (NLOS) reception. More importantly, high-rise structures are often associated with dense population and heavy traffics in a city. Thus, it is intuitive to quantify the urbanization rate for localization problems based on the building structures near the ego vehicle.

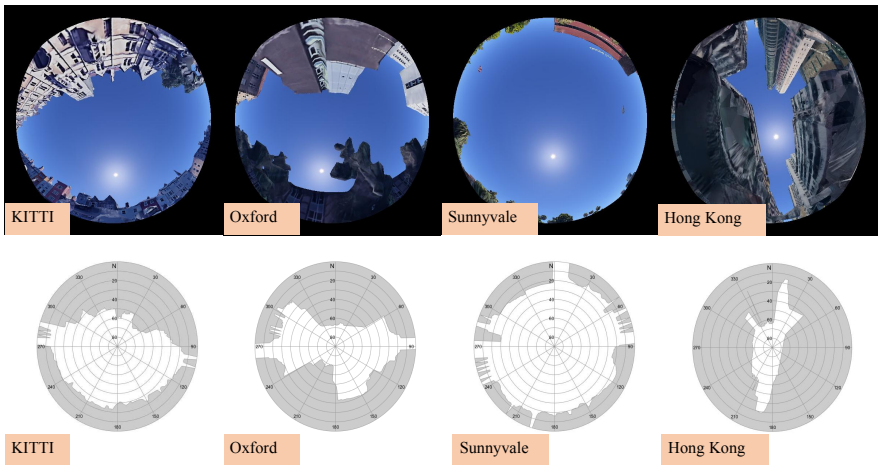
To define the urbanization rate of a specific location, a straightforward approach is to use Skymask, a polar plot of the structures' silhouette. Built upon a Skyplot [93], the Skymask includes the building structures as a mask for satellites. As shown in Figure 3.2a, the gray section denotes the sky-view blocked by the structures, and the white section denotes the clear overhead sky. One way to generate a Skymask is to employ the 3D building models which are widely used in the GNSS field [52, 53, 44]. Another method is to utilize the fisheye camera on the top of the vehicle (shown in Figure 3.3a) and algorithms like normalized cut [131] for image segmentation.

To quantitatively analyze a Skymask, we further define two parameters: the mean mask elevation angle μ_{MEA}, and the mask elevation angle standard deviation σ^2_{MEA}. The definitions are:

$$\mu_{MEA} = \frac{\sum_{\alpha=1}^{N} \theta_\alpha}{N}, \tag{3.1}$$



(a) Urbanization Measure Definition



(b) Skymask Comparison

Figure 3.2: Skymask illustration

$$\sigma^2_{MEA} = \sqrt{\frac{\sum(\theta_\alpha - \mu_{MEA})^2}{N-1}}, \tag{3.2}$$

where θ_α represents the elevation angle at a given azimuth angle α. θ_α here is highly correlated to the building heights. N denotes the number of equally spaced azimuth angles from the Skymask. We usually use N=360, meaning that the azimuth angle has a resolution of 1 degree.

When the ego vehicle is in a dense urban area, the Skymask is usually dominated by high-rise structures, resulting in a large μ_{MEA}, and a relatively small σ^2_{MEA}. In rural areas, on the other hand, both μ_{MEA} and σ^2_{MEA} would be relatively small. In places with mixed high-rise and low-lying buildings, σ^2_{MEA} would be of a relatively large value.

With the aforementioned urbanization measure, we evaluated the urbanization rate for current mapping/localization dataset, and the result is shown in Table 3.2 and Figure 3.2b.

The Skymask is generated by 3D building models in these areas, and we developed Figure 3.2b through post-processing. Readers may notice that none of the three datasets match the level of urbanization in Hong Kong.

Table 3.2: Quantified urbanization rate

Dataset name	Urbanization Rate	
	μ_{MEA} (degree)	σ^2_{MEA}
KITTI [37]	32.8	21.4^2
Oxford RoboCar [88]	42.3	16.3^2
Waymo (Sunnyvale) [154]	15.2	7.9^2
Ours	60.9	15.9^2

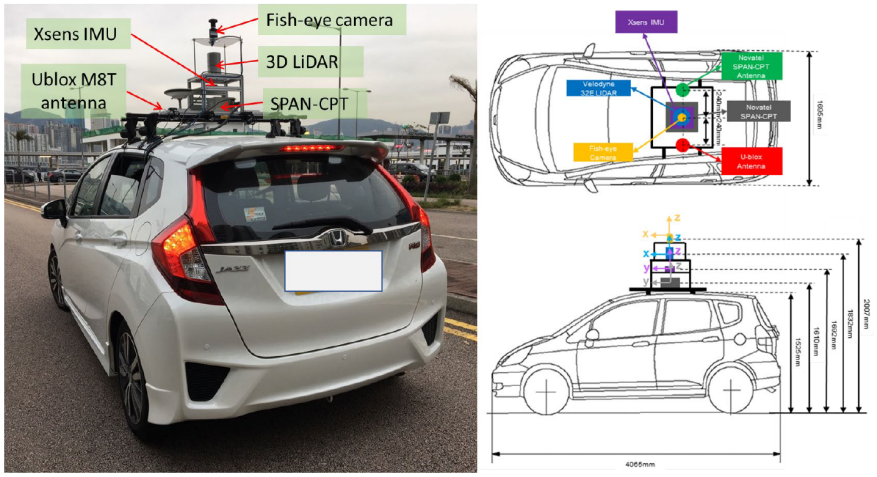
3.3 The UrbanLoco Dataset

Compared with the current public datasets for autonomous driving, our presented dataset covers more challenging urban trajectories with a full-sensor suite. The data is collected in densely populated areas in Hong Kong and San Francisco. These two cities have drastically different landscapes, driving behaviors (directions), architectures, and infrastructures. The trajectories cover urban canyons, tunnels, bridges, hills, sharp maneuvers, and other challenging scenes for the aforementioned mapping and localization solutions. More importantly, the scenes are filled with pedestrians, vehicles, trolleys, and cyclists. The sensors used are one LIDAR, six cameras (San Francisco), one fisheye camera (Hong Kong), one IMU, and one GNSS receiver. The ground truth for both cities is given by the Novatel SPAN-CPT, a navigation system that incorporates Real-time Kinematic (RTK) corrected GNSS signals and IMU measurements. In this section, we will explain our data collection platform and calibration in detail.

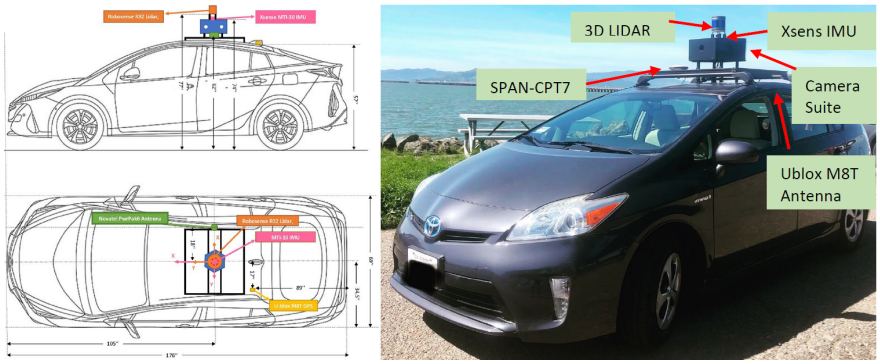
The platform for data collection in Hong Kong is a Honda Fit. The corresponding illustration and calibration coordinates are shown in Figure 3.3a. All the localization-related sensors are equipped in a compact sensor kit on the top of the vehicle:

- **LIDAR** Velodyne HDL 32E, 360 Horizontal Field of View (FOV), -30 +10 vertical FOV, 80 meters in range, 10 Hz;
- **Camera** Grasshopper3 5.0 MP (GS3-U3-51S5C-C), fisheye lens Fujinon FE185C057HA-1, 185 HFOV, 185 V-FOV, 10 Hz;
- **IMU** Xsens Mti 10, 100 Hz;
- **GNSS** Ublox M8T, GPS/BeiDou, 1Hz;

We use Ublox M8T to collect raw GNSS signals for future research on GNSS positioning: e.g., correcting the NLOS measurements [156]. A sky-pointing fisheye camera is also applied



(a) Hong Kong data collection vehicle



(b) San Francisco data collection vehicle

Figure 3.3: Data collection platforms

to capture the sky view. With the camera, it is possible to quantify satellite visibility based on the sky view image to monitor and improve the quality of GNSS positioning [63, 2]. The 3D LIDAR is employed to scan the environment for HD Map generation and SLAM [43] purposes. Lastly, the Xsens IMU is employed to collect raw acceleration and orientation measurements at high frequency.

The platform for data collection in California is a Toyota Prius as illustrated in Figure 3.3b. Slightly different from the Hong Kong platform, the following sensors are used:

- **LIDAR** RS-LIDAR-32, 360 Horizontal Field of View (FOV), -25 +15 vertical FOV, 80 meters in range, 10 Hz;
- **Camera** Six FLIR Blackfly S USB3, 2048*1536, 10 degree overlap on each side, 10Hz;
- **IMU** Xsens Mti 10, 100 Hz;
- **GNSS** Ublox M8T, GPS/GLONASS, 1Hz

The additional six 360-degree view cameras are synchronized and calibrated with the LIDAR. These two kinds of sensors are synchronized via triggering the front camera when the rotating LIDAR beam passes the front centerline (software trigger). Following cameras are triggered sequentially at a fraction of the running LIDAR frequency. To compensate for the possible delay in the triggering signal, the triggering time is adjusted to align LIDAR and cameras. Thus the specific section of the LIDAR scan and the corresponding camera image contain information at the same clock time. The intrinsic matrices of the cameras are calibrated through the kalibr toolbox [32], and the extrinsic calibration matrices are solved through Autoware [61].

The ground truth used in this dataset is provided by Novatel SPAN-CPT, a GNSS-IMU navigation system. The device is widely used for accurate localization assignments on mobile platforms. For the GNSS receiver, the received signal is corrected from the RTK signal sent from local public base stations. When the satellite visibility is satisfactory, the ground truth error is within 2 cm. In cases of a complete loss of the GNSS signal, the device is still able to output a continuous trajectory based on IMU measurements. The calibration certificate dictates that the error after 10s of GNSS blackout is within 12 cm.

3.4 Dataset Benchmark

A few state-of-the-art SLAM algorithms are evaluated with our dataset. For the tested open-source LIDAR and camera-based methods, the results are less satisfactory in the highly urbanized scenarios. A quantitative evaluation is given in Table 3.3, and a collection of 2D trajectory plots are shown in Figure 3.4. The dynamic objects in the moving scene largely contribute to the failures of SLAM algorithms, as most of these algorithms use static feature points to estimate the pose of the vehicle. Map $CA\_20190928173350$, collected during the rush hour on one of the busiest streets in San Francisco, is the most distorted map produced with current SLAM algorithms.

Laser-based Methods

The tested LIDAR-based methods are LOAM [167] (ranking No.2 on KITTI [37] at the time of the publication), and NDT [89] (a similar algorithm ranked No.24 on KITTI [37] at the time of the publication). For LOAM, the reported average translation error on KITTI is less than 5.7m/km, and the rotation error is 0.0013 degree/m. However, when retesting the algorithm with the open-source LOAM algorithm by Leonid Laboshin [72], the translation performance deteriorates significantly. For cases filled with dynamic objects (Figure 3.4e and 3.4f), the performance drops to above 10m/km in the horizontal directions. In cases of drastic altitude changes (Figure 3.4b, 3.4c and 3.4d), the translation performance further decreases to over 20m/km. As for rotation, the rotation performance decreases when the vehicle experiences sharp maneuvers (Figure 3.4c and 3.4d).

For NDT [89] family solutions, the best algorithm on KITTI reaches 8.9m/km in translation error and 0.003 degree/m in rotation. In our evaluation, we apply the package prepared by Kenji Koide [67] and fine-tune the parameters three times for the best performance. Furthermore, a half real-time playback rate is used to guarantee solution convergence. However, the result is less satisfactory: in the longest testing route (5.9km, Figure 3.4a), the translation error is more than 100 meters overall. It is also observed that the algorithm performed poorly on altitude estimation.

Vision-based Methods

As for visual odometer estimation, we use the open-source VINS-MONO [121] algorithm developed by Qin et al. VINS-MONO tightly couples visual odometry with IMU estimation to output an optimized localization result. As pointed out in VINS-MONO, the algorithm out-performs most existing visual odometry methods. During the experiment, we notice that the visual odometry is very sensitive to changes in light conditions: the algorithm fails when entering-exiting a tunnel. After using a half-real time playback rate, we generate the continuous trajectory successfully. For performance, while the algorithm slightly outperforms the LIDAR-based method in translation, the error in roll/pitch/yaw angle estimation is worse. The performance further deteriorates when the path is filled with sharp maneuvers (Figure 3.4d). Since VINS-MONO takes time for initialization, a rigid body transformation [136] is applied to the constructed map before evaluation.

From the aforementioned evaluation and analysis, it is safe to say that our dataset in Hong Kong and San Francisco addresses urban driving challenges adequately. Indeed, the dataset is a valuable test field for future urban-focused localization solutions.

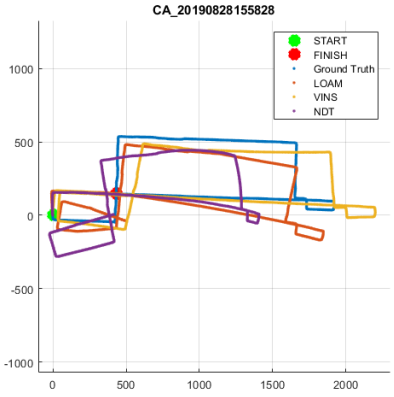
3.5 Chapter Summary

In this chapter, a challenging full sensor suite dataset for autonomous vehicle mapping and localization is presented. Our data collection platform contains various sensors for mapping and localization purposes: LIDAR, cameras, IMU, and GNSS receivers. Compared with current datasets, the UrbanLoco dataset contains trajectories from more challenging scenes, and the exemplar state-of-the-art algorithms performed poorly on the dataset. An urbanization measure is also proposed to further quantify the challenges in different scenarios.

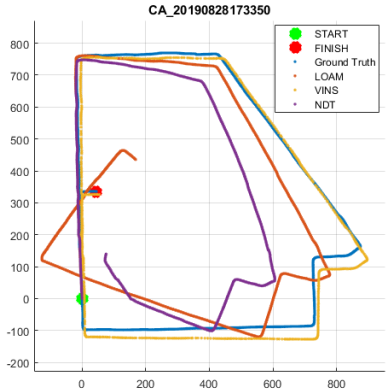
For the UrbanLoco dataset, the data is collected with one single mobile mapping platform. However, in industrial applications, a mapping team is usually equipped with a fleet of mapping vehicles. How to efficiently route each vehicle for more better data collection will be the topic of the next chapter.

Table 3.3: Map evaluation results

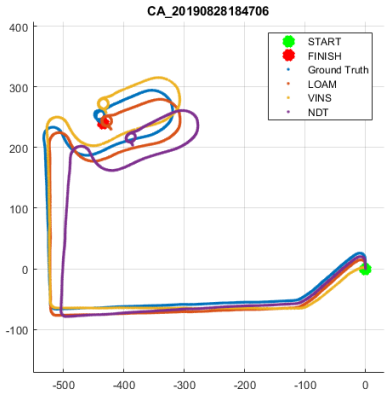
Data Rosbag	Description	Length (km)	Method	Mean Translation Error (m)			Mean Rotation Error (deg)		
				X	Y	Z	Z	Y	X
CA_20190828155828	Traffic, Structures	5.9	LOAM	36.91	95.06	149.97	10.15	0.97	0.50
			VINS	42.68	42.68	23.24	6.04	0.69	2.05
			NDT	184.05	94.01	337.91	3.68	4.43	5.89
CA_20190828173350	Traffic, Hills	3.2	LOAM	81.74	85.69	80.91	20.149	6.11	2.65
			VINS	26.745	34.756	14.83	8.12	0.81	2.47
			NDT	129.27	71.13	253.76	10.10	0.76	11.07
CA_20190828184706	Hills	1.8	LOAM	32.90	25.01	19.83	0.42	0.97	0.43
			VINS	32.52	31.47	26.89	2.65	0.89	1.28
			NDT	40.35	24.82	77.74	0.31	3.46	5.02
CA_20190828190411	Hills, Maneuvers	1.0	LOAM	29.53	21.97	15.50	12.29	1.58	2.85
			VINS	22.27	17.57	23.84	11.32	4.12	4.31
			NDT	29.51	16.78	14.62	9.04	2.08	0.42
HK_20190426101600	Structures, Traffic	0.8	LOAM	10.56	9.73	0.36	1.23	0.05	0.03
			NDT	6.14	7.77	2.01	7.23	0.43	0.19
HK_20190426100200	Structures, Traffic	0.7	LOAM	19.65	16.15	0.88	1.64	0.03	0.68
			NDT	6.63	8.28	2.10	7.80	0.32	0.90



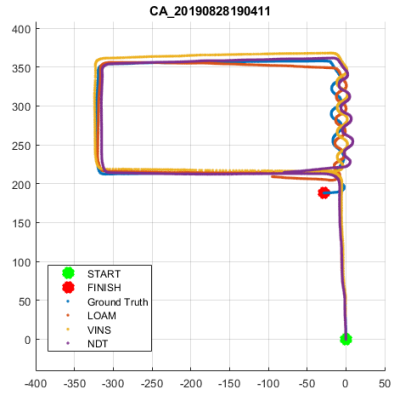
(a) CA_20190828155828



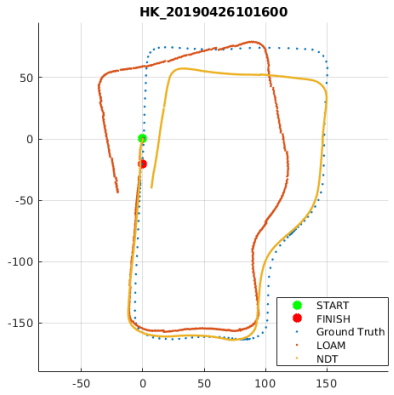
(b) CA_20190828173350



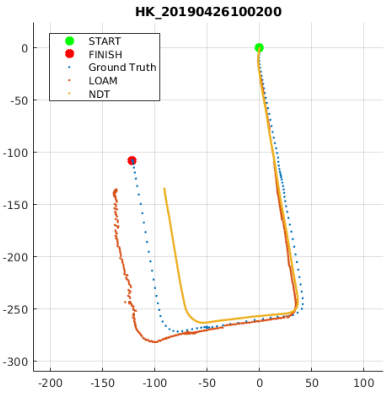
(c) CA_20190828184706



(d) CA_20190828190411



(e) HK_20190426100200



(f) HK_20190426101600

Figure 3.4: Map evaluation results

Chapter 4

Routing for A Mapping Fleet

In Chapters 2 and 3, sensor calibration and data acquisition processes are studied for single-vehicle setups. However, most mapping teams are equipped with multiple mapping vehicles for more efficient data collection and map updates. Typically, each vehicle explores a specific section of a city to avoid repeated visits. Here, the natural question is: how to route these vehicles in an ever-changing urban environment?

A classical mathematical problem, routing has been studied by researchers for centuries. Just like instructing knights to travel through the Seven Bridges of Königsberg, here we are directing vehicles to traverse a city. However, the dynamic nature of the urban environments makes it hard to solve the problem in a classical graph theory way. First, the urban traffic conditions are changing during the mapping process, making it a dynamic problem. Secondly, maps expire the moment they are created, requiring frequent revisits to the same location.

Observing the characteristics of the urban routing problem, a Model Predictive Control (MPC)-based solution is discussed as a potential strategy for the map routing problem. In the problem formulation, the urban map is first treated as a Directed Cyclic Graph. We then construct a linear model for the exploration task. The model is further extended to solve real-time routing and map updating problems. We have experimented with the MPC-based solution on simulated and real-world maps, and the proposed framework demonstrates an efficient exploration of cities[1].

4.1 Introduction

As the autonomous driving industry gradually shifts its interest to urban driving tasks, it is critical for driver-less vehicles to be equipped with accurate, comprehensive, and up-to-date maps of various city scenes. HD maps, with their detailed and vectorized representation of the environment, have already demonstrated their quintessential role in urban driving [3, 33, 146, 162].

[1]This chapter includes materials from the author's previous project [170]

The great demand for maps has boosted the growth of numerous mapping companies and their mapping fleets. However, how to route the mapping vehicles in a complicated urban scene is still a challenge. Firstly, the urban environment is ever-changing: constructions, accidents, traffic congestion, etc. could easily make an easy drive extremely time-consuming. Secondly, an HD map is outdated immediately after its creation. How to keep the entire map relatively up to date remains to be another challenge. Traditional solutions to the traveling salesman problem (TSP) and the vehicle routing problem (VRP) largely assume a static environment, which is not suitable for the mapping vehicle routing problem nowadays.

In this chapter, we propose a Model Predictive Control (MPC)-based algorithm to solve the mapping vehicle routing problem. In our proposed algorithm Map Routing with MPC (MR.MPC), we first extracted the urban road network from the open-source map platform OpenStreetMap [109]. Following [171], we represent the map as a Directed Cyclic Graph (DCG) stored as a two-dimensional matrix. The MPC problem is defined by vehicle state, map state, vehicle control, and corresponding linear dynamics. We further design a robust cost function with time decay to encourage vehicle exploration. With the proposed formulation, the map routing problem could be efficiently solved by a Mixed Integer Linear Programming (MILP) solver. Lastly, we propose to extend this formulation to map updating problems as well.

The designed algorithm is tested in simulated environments with simple road structures and complicated real-world scenes in Downtown Berkeley. The proposed algorithm demonstrates an efficient exploration strategy within the dynamic environment.

In this work, our major contributions are:

- An MPC-based routing solution is proposed for mapping fleets in a highly dynamic urban environment;
- The routing problem is formulated as a linear system for an efficient solution with MILP solvers.
- A potential alternative solution is provided for TSP and VRP from the control point of view.

The Vehicle Routing Problem

The identified routing challenge in the urban mapping problem resembles those seen in the TSP and the VRP. For the TSP, one agent is seeking an optimized route in a graph, while in the VRP there are multiple agents for optimization. Both of these problems are proved to be NP-hard [34].

Two schools of thought dominate the VRP solution: the optimization-based method, and the learning-based method. To start with, the optimization-based method is seeking better algorithmic solutions for the NP problem, often with better representations [22] or extensions [48]. Such methods are usually exhaustive and optimal, but they take a significant amount of time for execution.

More recently, the rise of neural networks has encouraged researchers to solve the VRP in a learning fashion. To start with, the VRP problem could be modeled with reinforcement learning techniques, treating each vehicle as an agent and proposing policies at the graph node. Typical works utilize Value Iteration Networks for Q value predictions [74, 137]. Some other methods, observing the graphic nature of the VRP problem setup, utilize Graph Neural Networks to explore the edge/node relationships [75]. With improved computational time, the learning-based method, however, could not always provide an optimal solution. More importantly, the learning-based methods require a large amount of data or simulation to work with.

Observing the elegant representation of traditional optimization-based methods and the iterative nature of the learning-based methods, we propose to model the mapping vehicle routing problem with MPC. In general, we formulate the long-horizon problem with iterative steps from the solution of MPC, and each state has a value estimation from the cost function. For each time step, however, we are still solving an optimization problem with agents in graphs. Furthermore, an MPC formulation of the VRP problem makes it possible for us to handle dynamic environments and map updates.

4.2 An MPC Formulation of the Routing Problem

Map Preliminaries

Following the formulation in [171], we represent a city road network with a DCG. As shown in the Figure 4.1, the city road map M is consisted of vertices $\{V_i\}$ and edges $\{E_{ij}\}$. Each vertex V_i represents an intersection, and the edge E_{ij} represents a directional road from vertex V_i to vertex V_j. Here, we specifically use a directional representation for the existence of one-way roads in the urban environment.

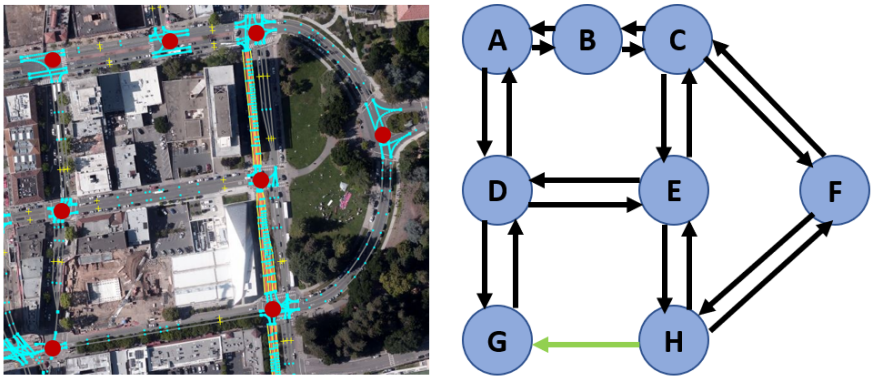


Figure 4.1: Map hierarchy as a directed cyclic graph

A graph, in data structure languages, could be represented in various ways. In this problem formulation, considering the linearity of the solution, we chose to use a 2-dimensional

matrix M with float number entries to represent the DCG. For a map with N vertices, the matrix M will be an N by N matrix, with each M_{ij} representing the cost to go from intersection V_i to intersection V_j. The cost could be defined according to the mapping teams' interests. A typical choice would be the length of the road. In this work, we firstly use the length of the road extracted from OSM as the cost. We then add a traffic congestion factor for certain edges to simulate the real-world condition of urban driving. M is a sparse matrix with all the non-connected roads represented by infinity cost.

We also design a binary matrix M_{bool} to represent the connection logic of the map for easier MPC formulation. An entry with 1 at edge E_{ij} means that the road from V_i to V_j is connected.

MPC parameter and discretization

To start with, we define the following parameters of the problem. The mapping fleet has a total of K exploration agents, each of which represents a mapping vehicle C_k. Following the aforementioned map definition, we have a N vertices map. Lastly, the horizon of the MPC exploration is T.

Different from the real world traversing with physical locations and time-flow in seconds or minutes, in the Map Routing task we are only seeking specific actions at the vertex of the map M. That is to say, the algorithm only needs to make a decision when the vehicle is or will be at an intersection, ignoring the deterministic traveling along the current edge. Thus, we design the MPC model with a different timeline as compared with the physical world time-flow. In the MPC problem, the timeline t_{mpc} is defined by discrete steps. Between t_{mpc} and $t_{mpc} + 1$, an arbitrary agent C_k is able to travel one whole edge E_{ij} as long as the edge has a bounded cost.

States

The first part of the state is the fleet location state $[X_{fleet}(t_{mpc})]$ of a dimension of N by K at each times step t_{mpc}. The column k of the $[X_{fleet}(t_{mpc})]$ matrix is a one-hot encoded vector representing vehicle C_k's location at t_{mpc}. An entry of 1 at $[X_{fleet}]_{nk}$ means that the vehicle C_k is currently at the vertex V_n. At the beginning of the exploration, assuming that all vehicles depart from the depot at the zeroth vertex, the matrix $[X_{fleet}]$ will have a row of ones at the top and zeros for other entries.

The second part of the state is the map exploration state $[X_{exp}(t_{mpc})]$, an N by N binary matrix at each timestamp t_{mpc}. Entry $[X_{exp}(t_{mpc})]_{ij}$ is a binary indicator of the exploration status from vertex V_i to vertex V_j, where 1 means the edge is not explored, and 0 means that the edge has been explored or not connected. At the beginning of the exploration, $[X_{exp}(0)]$ would be the same as M_{bool}.

Controls

To linearize the system for faster solution time, we expand the controls to a 3-dimensional matrix $[U(t_{mpc})]$ at each time step t_{mpc}. $[U(t_{mpc})]$ is of shape N by N by K, where the $ijk-th$ entry represents that the vehicle C_k choose to travel from vertex V_i to V_j. Intuitively, the matrix $[U(t_{mpc})]$ is a sparse matrix.

As a side note for the choice of a 3-dimensional representation of the control, we also implement a 2-dimensional control version with the same size as the $[X_{fleet}(t_{mpc})]$. However, such a formulation would lead to a nonlinear dynamic system. We execute the two formulations with GLPK [91] and Bonmin [6], and conclude that the nonlinearized formulation would take $10\times$ more time to solve.

Dynamics

The first dynamic rule in Equation 4.1 reflects the change in the $[X_{fleet}(t_{mpc})]$ with given $[U(t_{mpc})]$.

$$X_{fleet}(t_{mpc}+1)[j,k] = X_{fleet}(t_{mpc})[j,k] + \sum_{i=1}^{N} U(t_{mpc})[i,j,k,t] - U(t_{mpc})[j,i,k,t] \tag{4.1}$$

As expressed above, the interchanged indexes and signs in the summation term handle the arrival and departure conditions at the vertex. When no actions are taken, which is highly unlikely, the state indicator will also stay.

The second dynamic rule in Equation 4.2 demonstrates the change in $[X_{exp}(t_{mpc})]$ with respect to $[U(t_{mpc})]$.

$$X_{exp}(t_{mpc}+1)[i,j] >= X_{exp}(t_{mpc})[i,j] - \sum_{k=1}^{K} U(t_{mpc})[i,j,k,t] \tag{4.2}$$

The summation term collects the trajectories from all vehicles corresponding to the edge of interest. Since multiple vehicles could traverse on the same edge when control options are limited, the right-hand side of the equation could be negative. As a result, we chose to use inequality during implementation. The cost function mentioned in the following sections would enforce the $[X_{exp}(t_{mpc})]$ to take the value of zero when there is indeed an action involved on the specific edge.

Constraints

Other than the dynamics in the previous section constraining the MPC problem, there are also static constraints at each time step.

The first two constraints are related to the initial conditions of X_{fleet} and X_{exp}. All vehicle locations are assumed to be at the depot, encoded by the term $X_0[i,k]$. The initial exploration record X_{exp} will be the same as the binary map $M_{bool}[i,j]$.

$$X_{fleet}(0)[i,k] = X_0[i,k] \tag{4.3}$$

$$X_{exp}(0)[i,j] = M_{bool}[i,j] \tag{4.4}$$

The third constraint is to ensure that each action taken by the controller is feasible, meaning the next target node is reachable.

$$U(t_{mpc})[i,j,k] <= M_{bool}[i,j] \tag{4.5}$$

The fourth constraint is to guarantee that at each time step, each vehicle only takes one or no action when the next target is feasible. The agent will never choose an infeasible action. Considering the binary nature of the $U(t_{mpc})$, we could formulate this rule as an inequality.

$$\sum_{i=0}^{N}\sum_{j=0}^{N} U(t_{mpc})[i,j,k] <= 1 \tag{4.6}$$

The fifth constraint is to enforce that at each time step, each vehicle departs from where it is located.

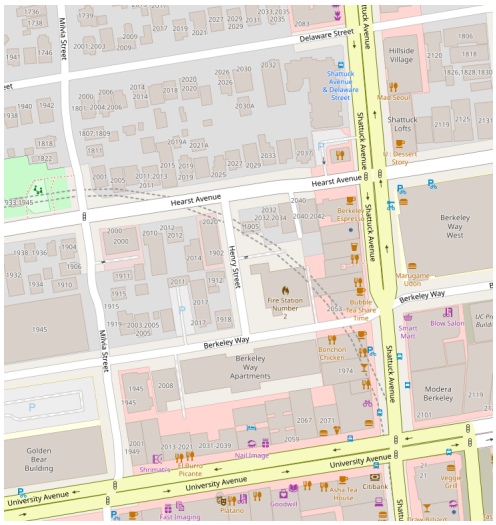
$$U(t_{mpc})[i,j,k] <= X_{fleet}(t_{mpc})[i,k] \tag{4.7}$$

Cost Function

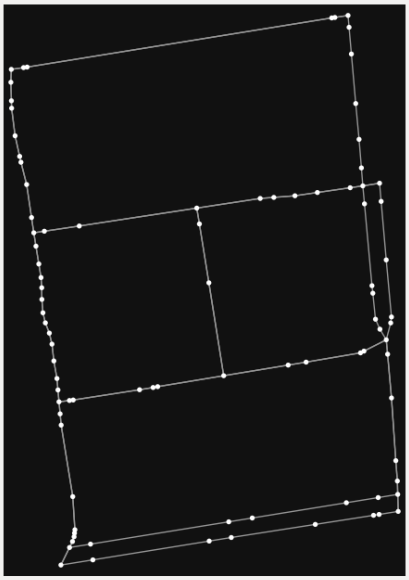
As mentioned in the introduction, the cost function involves an exponential decay parameter γ. Here we follow a similar discount design in the Markov Decision Process to encourage earlier exploration. The cost function is trying to minimize the sum of exploration matrix cost at every time step.

$$\min \sum_{t=0}^{T}\sum_{i=0}^{N}\sum_{j=0}^{N} \gamma(t_{mpc}) X_{exp}(t_{mpc})[i,j] M[i,j] \tag{4.8}$$

With changing road conditions like traffic congestion or construction, the M matrix would be updated with a higher cost at each different MPC step. For the map updating case, one could redefine the map cost as a dynamically changing matrix at each real-world time step. The cost will reduce abruptly after the exploration, but it will increase exponentially to an upper bound as time elapses.



(a) Downtown Berkeley map for simulation



(b) Pixels to Point-cloud

Figure 4.2: Bi-directional projection demonstration

4.3 Experiments and Discussions

Simulator Setup and Data

For simulation purposes, we design a continuous world environment to test our MR.MPC algorithm. In contrast to the discrete MPC model, the real-world model is based on the clock time, in seconds. Without the limitation of the linearized system, states $X_{fleet,sim}(t)$ and $X_{exp,sim}(t)$ are more expressive. In the simulation, $X_{fleet,sim}(t)$ is a list of length K, filled with dictionary data structure storing the vehicle C_k's information. For each vehicle C_k, we are interested in the current edge location (from V_i to V_j), the remaining edge cost on the road, and a discrete control queue. Assuming that the cost in M is the traveling distance between two nodes, the remaining cost is reduced at a rate of 10 meters per second as the continuous system dynamic. Such a rate equals a vehicle traveling at around 22 miles per hour. The first-in-first-out (FIFO) control queue is the output command from MPC, which will give a control command once the vehicle's remaining edge cost is smaller than a threshold.

We implement the simulator environment in *Python*, and the MPC solver is built with *Pyomo* [8] and *GLPK* [91]. We also implement the MPC solver with *Casadi* and *Bonmin* [6] as a comparison. We test the MR.MPC algorithm with simulated toy case and the real world map in Downtown Berkeley.

Figure 4.3: Simulation process on the Berkeley map

Results

Restricted by the project scope, this experimental section focuses on the static and dynamic map only, ignoring the map updating problem. We first design a toy case consisting of five nodes in an A shape with two single-directional roads radiating out from the center node. The optimal solution in three time steps should yield a cost of zero at the terminal state. We use this case as a sanity check for the proposed algorithm.

From the Berkely OSM map shown in Figure 4.2a, we extract a DCG map model for evaluation (Figure 4.2b) in real world. The simulation result is shown in Figure 4.3, where the blue and green dots are fleet vehicles in the scene. Another factory map is also acquired and processed for the experiment purpose.

Table 4.1 shows the solution cost in the static setting. We choose a random explorer as a baseline for comparison. It could be seen that the MR.MPC algorithm could efficiently explore the map in a limited amount of time, while the random agent wastes the effort.

Table 4.1: Static solution result

Maps	Linear model	Baseline
Toy case	0	1094
Downtown Berkeley	7931	11594
Factory	15950	39855

We further deploy the algorithm with a dynamic map update. We randomly change a number of edges to infeasible during the running process to simulate road blockages and compared the cost with the static method in Table 4.2. If the vehicles choose the original exploration route, the exploration cost associated with the congestion will be much higher

than the dynamically re-planned strategy. we only carry these experiments on larger Berkeley maps, because the toy case does not have enough vertices for an update.

Table 4.2: Dynamic solution result

Maps	Dynamic update	Static exploration
Downtown Berkeley	437	11594
Factory	8501	39855

Discussion

As previously mentioned in the experiment setup section, we compare the proposed linear case with the *Casadi* implemented nonlinear case. The solution time could be seen in Table 4.3. It is clear that the *Pyomo* model and *GLPK* could be solved in a shorter time. However, it is worth noticing that the linear model has a $O(N^2)$ complexity associated with the control, which could suffer from the Curse of Dimensionality [5] when a larger map is given.

Table 4.3: Linearization effect on MPC solver time (s)

Maps	Linear model (Pyomo+GLPK)	Nonlinear model (Casadi+Bonmin)
Toy case	0.02	0.09
Downtown Berkeley	0.7	6.5
Factory	0.45	41.32

4.4 Chapter Summary

In this chapter, an MPC-based map routing framework MR.MPC for urban map routing problem is proposed. The introduced framework can dynamically explore the complicated urban road network, taking both traffic conditions and road structure into consideration. Experimented with both simulated and real-world data, MR.MPC demonstrated efficient routing solutions in dynamic urban scenes.

Chapter 4 here concludes the discussion on mapping platforms and data. As a review for Part I, we start with sensor setups on mobile mapping platforms in Chapter 2, where a joint spatial-temporal calibration method is proposed. Built upon well-prepared sensor suites, Chapter 3 showcases UrbanLoco, an urban mapping dataset. In Chapter 4, we try to solve the mapping fleet routing problem with an MPC formulation.

In the life cycle of an HD map, finishing the aforementioned steps leaves us with datasets containing accurate raw sensor measurements of the urban environment. The next step in the HD map development will be to draw the map with the hard-won data.

Part II

HD Map Construction

Chapter 5

Road Module Exploration

In traditional paper maps development, after raw measurements are collected by the survey team, cartographers will take these data and draw a representation of the world onto a piece of paper. These cartographers are specially trained to decode the numbers to meaningful semantic representations. For example, a series of elevation readings will be transformed into contour lines on the map.

For HD maps development, one also needs these specially trained "cartographers". Nowadays human labelers are employed to mark out the semantic components in an HD map. For example, lane boundaries, curbs, and stop lines are marked out by human beings. However, considering the exhaustive details in an HD map and the massive scale of the urban environment, manual labeling is not a feasible solution as the autonomous driving market grows. Indeed, HD map researchers are working towards replacing human cartographers with cartographic algorithms to automatically construct an HD map.

This chapter discusses an automatic HD map construction algorithm designed specifically for the road modules: the algorithm studies the lanes between two intersections. In urban scenes, lanes are more complicated with forking, merging, and irregular shapes. The work included in this chapter proposes an approach based on the semantic particle filter to tackle these scenes. The map skeleton is first structured as a directed cyclic graph from the online mapping database OpenStreetMap [109]. Our proposed method then performs semantic segmentation on 2D front-view images from ego vehicles and explores the lane semantics on a birds-eye-view domain with true topographical projection. Exploiting OpenStreetMap, we further infer lane topology and reference trajectory at intersections to verify outcomes of the lane inference. The proposed algorithm has been tested in densely urbanized areas, and the results demonstrate an accurate and robust reconstruction of the lane-level HD map[1].

[1]This chapter includes materials from the author's previously published work [171]

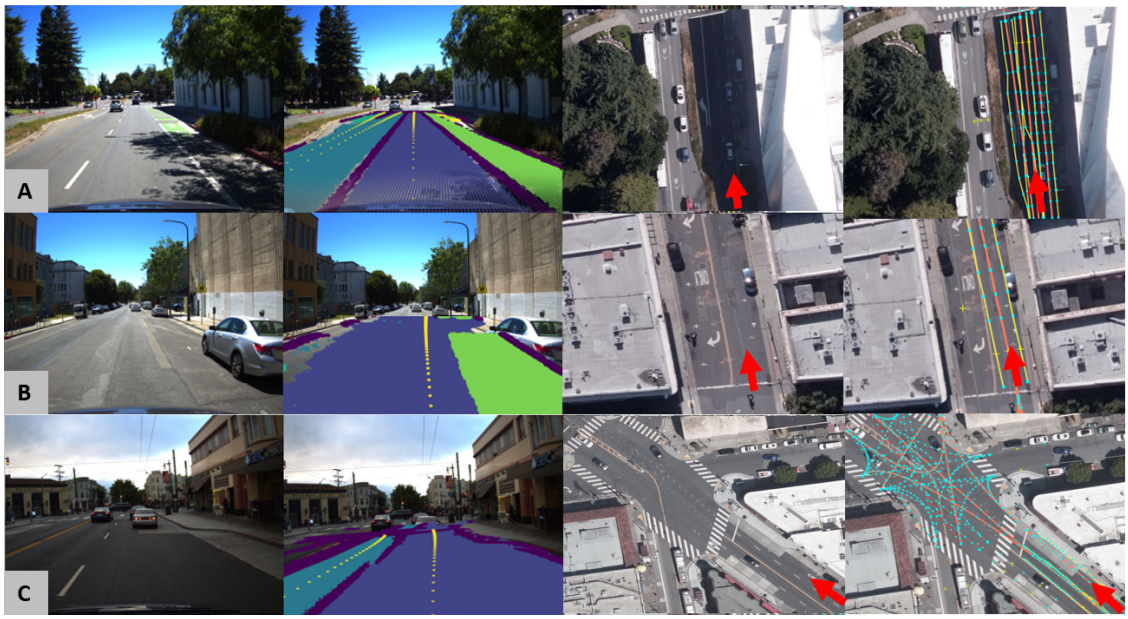


Figure 5.1: Roads in an urban environment

5.1 Introduction

High definition (HD) maps have become a crucial component for full autonomy in a variety of complex scenarios. Encoded with accurate and comprehensive information about the static environment, HD maps can significantly facilitate perception, localization, prediction, and planning [130]. HD maps contain multiple layers of information abstractions, and lane-level information plays the quintessential role in many applications. Embedded with lane geometries, road semantics, and connection topology, the lane layer can be utilized for defining a potential region of interest (ROI) for some key modules in autonomous driving, including but not limited to object detection [162], regulating the lateral location of the ego vehicle [86, 146], and predicting the behavior of other vehicles [33].

With numerous alluring applications, these lane-level HD maps, however, do not scale easily. Many of these HD maps are restricted to small-scale environments due to the high costs of manual labeling and maintenance [115]. Recently, a few commercial products have been launched to automatically map highways [107], where lanes are structured and markings are clear. In urban scenes, however, the roads are much more complicated. As shown in Figure 5.1, urban roads may have complicated forking, potholes, broken markings, or even no markings at all. Furthermore, a city road also carries irregularities such as parking zones, bus curbs, and bike lanes. As a result, automatic lane-level mapping in urban areas remains an open and challenging problem.

It is worth noticing that the lane-level HD map construction problem is different from geometry tasks like lane detection or trajectory inference. HD map requires a semantic and topological understanding beyond the instance level, meaning that the map should contain

logic connections with geometric information. As compared with simple lane boundary regression, HD map constructors further infer the merging/forking relationship between lanes; as compared with trajectory inference, HD maps further address the lane topological relationship.

Previous works have regarded lane detection and intersection trajectory generation as separate problems, and most of these works only target the geometric understanding of the scenes. In this work, we start with lanes and try to understand intersections jointly, providing topological understanding beyond simple geometric representations.

In this chapter, we firstly define the HD map representation as a directed cyclic graph (DCG) for easy data storage and query. We then propose the semantic particle filter to automatically generate an urban lane-level HD map with a front view camera and an optional LIDAR sensor. The proposed method contains two major components: a semantic segmentation network for scene understanding, and a sequential Monte Carlo lane tracing module over bird's-eye-view (BEV). We further utilize lane exploration with OpenStreetMap (OSM) [109] to showcase the potential in intersection inference. The whole pipeline only requires one single execution per road direction for a complete reconstruction of the lane-level details including the lane boundaries, reference trajectories, lane splitting information, and road topology. Lastly, we represent our generated lane map in a differentiable format for downstream modules. We test the proposed method in densely urbanized areas such as San Francisco and Downtown Berkeley from the UrbanLoco dataset [157], and some exemplar mapping results can be seen in Figure 5.1. The experiment covers areas with lane merging/splitting, missing/broken lane markings, complicated intersections, and irregular road shapes such as bus curbs and parking areas. The results demonstrate an accurate and robust construction of the lane-level HD maps.

The major contributions of the work included in this chapter are:

- We propose to combine semantic scene understanding with Monte-Carlo sequential exploration for accurate and robust HD map construction in urban scenes.
- Geometrical representation and topological relationships are inferred for urban lanes.
- OSM is exploited as a coarse prior map to construct a directed cyclic graph representation of the urban road structure.
- The proposed algorithm is tested with a public dataset collected from densely urbanized areas and validated the robustness and accuracy.

Extracting mapping semantics from raw sensor data has been a popular topic in the HD map research community. We will start with a brief introduction of a few prior works in this field.

Lane detection: markings

Since the lanes are defined by markings painted on the road surface, a natural way for lane-level map construction is initiated from the lane marking detection perspective. Nieto et al. use a step-row filter [103] for lane marking detection and apply the Rao-Blackwellized Particle Filter for lane tracing [104]. In [76], Li et al. use Convolutional Neural Network and Recurrent Neural Network for lane marking detection on highways. More recently, Garnett et al. [35] and Guo et al. [46] further use the 3D-LaneNet framework not only to classify the lane in an image but also to predict its location in 3D.

These approaches achieve high-quality marking detection on highway or suburban roads where the shape of the roads is simple and can be approximated by a polynomial-like function. However, it is hard to implement the aforementioned methods in urban scenarios with complicated road structures, broken/missing lane markings, and frequent road splits. Furthermore, the methods mentioned in [35] and [46] only work for scenarios with mild changes in 3D slope, disregarding the abrupt changes in road topography.

Lane detection: drivable areas

Some other methods focus on drivable areas for lane detection. Meyer et al. designed a neural network for ego and neighboring lane detection [95]. Kunze et al. create a scene graph from semantic segmentation to generate a detailed scene representation of the drivable areas and all road signage [70]. On the other hand, Roddick et al. use a pyramid projection network to extract the drivable area as well as other vehicles [124]. Neither of these methods is extended to the HD mapping domain, allowing a third vehicle to make full use of the detection results.

Intersection lane inference

Understanding the intersection structure is an indispensable technique for autonomous driving and HD Map generation. Thus a number of studies have been conducted to extract invisible lanes and connecting topology at intersections.

Trajectories of other vehicles have been commonly used for intersection exploration. For example, in [36] [58], the authors use vehicle trajectories acquired from on-board sensors such as stereo cameras or LIDARs. Later, Meyer et al. use simulated vehicle trajectories and employed a Markov chain Monte Carlo sampling to reconstruct the lane topology and geometry at both real and simulated intersections [97, 96]. In [125, 20], the authors leverage collected GPS data loaded on fleet vehicles. These methods have the potential to estimate the lane-level structure of intersection, but they are data-hungry, and the performance heavily relies on the quality of vehicle trajectories, which themselves are non-trivial to acquire.

More recently, directly predicting road connections at intersections from camera images became another popular direction for lane inference. The work from Nvidia formulates the inference problem as classification and chose the best trajectories from a real trajectory pool with cross-entropy loss [117]. And Paz et al. predict the trajectory from a Gated Recurrent

Unit on a BEV semantic map [114]. However, both methods focus on the ego vehicle lane and only predict one trajectory as a reference for the ego vehicle, not considering all visible lanes at an intersection.

HD map generation

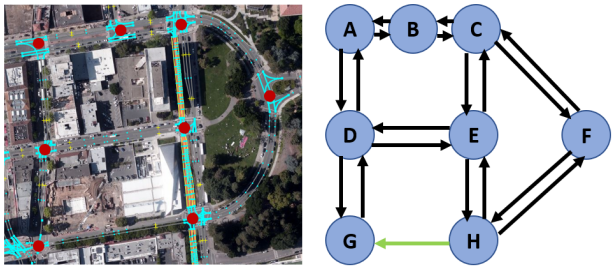
Large-scale map generation has long been considered a Simultaneous Localization and Mapping (SLAM) problem [164]. However, instead of semantics, common SLAM algorithms focus more on odometry estimation. Recently, in [115] and [25], the authors use semantics in the image domain and fuse semantic results with LIDARs to generate a semantic representation of the scene, not exactly an HD map. In terms of lane-level map generation, a common approach is to accumulate extracted road environments along with localization results. Authors of [45] and [57] utilize LIDAR to extract road environments, and leverage the OSM as a map prior. In [57], fork and merge of the lanes are recognized by particle filter-based lane marking tracking. Homayoundar et al. on the other hand, use a directed acyclic graph (DAG) and treat each detected lane marking as a node to decide future actions for connection, initiation, or termination [50]. However, these methods focus on road segments ignoring intersections, and they do not define the start and endpoint of the lanes. Thus, connections between lanes are not studied in these works. With the help of aerial images, authors of [94] study both road sections and intersections. However, the method is limited by the availability and resolution of aerial images.

While the aforementioned efforts in lane detection and map generation have significant contributions to the community, they are limited to either simple road structures or single road segments. Our work targets specifically the complicated urban driving scenes and considers both lane segments and intersection inference. Furthermore, we consider the topographical deformation of the road surface for more robust and accurate lane detection.

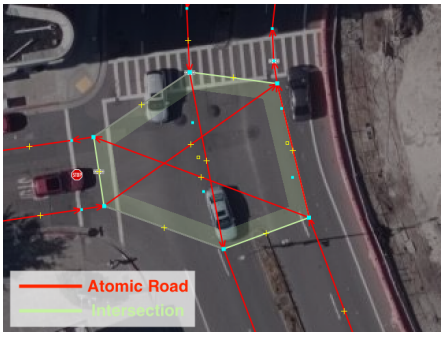
5.2 Road Network Abstraction

We depart from a topological point of view for this cartography task. A city road network consists of individual road segments and connecting intersections. As shown in Figure 5.2a, an urban lane-level HD Map M could be abstracted into a directional cyclic graph representation with each edge E_{ij} representing a directional road from an intersection node I_i to another intersection node I_j . We deliberately choose to have an edge for each direction due to the ubiquity of one-way roads and physically separated two-way roads in urban scenes (as shown in Figure 5.1). Previous works focused on either the edge E_{ij} or the node I_i for geometrical information extraction, but we study the HD map M containing both roads and intersections

$$M := \{E_{i\cdot}, I_i\}. \tag{5.1}$$



(a) DCG representation of the map M



(b) Intersection ROI

Figure 5.2: Map hierarchy as a directed cyclic graph

To acquire such a high-level map skeleton, we extract the coarse road-level topological information from OSM [109]. Disregarding the direction of travel for most roads, the OSM defines a road as a series of nodes connected together under a road instance. For special two-way roads with solid barriers as the median, OSM would have one road instance for each direction. Furthermore, the OSM defines an intersection as a connection node of two or more road instances. Although represented by a single node, the actual intersection is a geometrical region where multiple roads/lanes intersect. Utilizing such a definition, we predict the intersection ROI (shown in Fig 5.2b) with a polygon formed by the closest road node to the intersection node, leaving the rest of the road nodes as part of the atomic road.

With the atomic road E_{ij}^0 and intersection patch I_i^0 defined as the skeleton M^0 of our map, we can match the moving data collection vehicle to an atomic road or an intersection in the road network. Different from the goal variables M, E_{ij} and I_i, M^0, E_{ij}^0 and I_i^0 contains neither geometric nor topological information of the lanes. Now we proceed to the proposed methodology for automatic map generation.

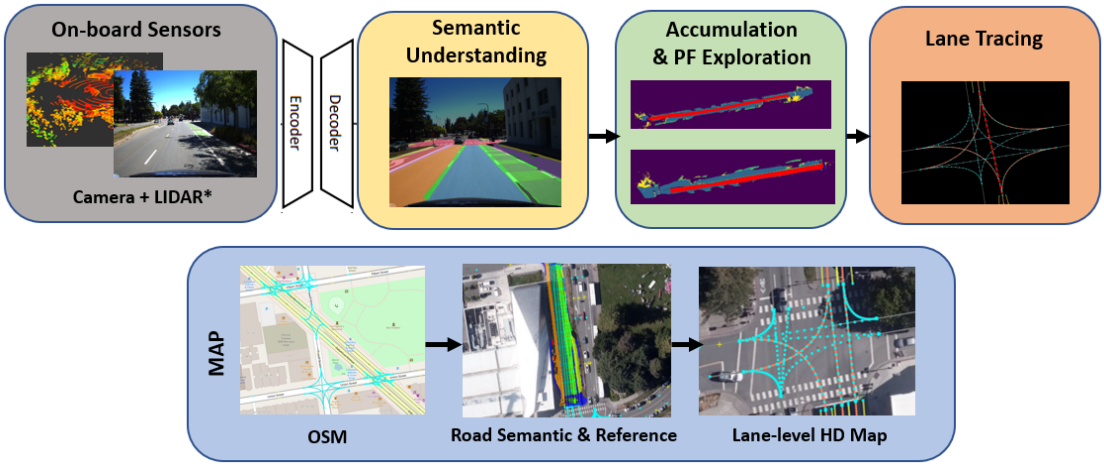


Figure 5.3: Particle filter exploration framework

5.3 Particle Filter Exploration

Problem formulation

To automatically generate the lane-level map from the aforementioned map skeleton M^0, our proposed approach utilizes an onboard front-view camera C_t and an optional (represented by *) LIDAR L_t^* at time t. We also require the synchronized global vehicle poses RT_t, which come either from external sensors or SLAM algorithms. Representing our framework as a function $F(\cdot)$, the goal map shown in Equation 5.1 could be further described by

$$M := \{E_{i\cdot}, I_i\} = F(C_t, L_t^*, RT_t, M^0). \tag{5.2}$$

The goal for atomic roads E_{ij} estimation is to infer lane k's centerline $\{L_k\}_K$ and its left-right boundaries $\{B_{k,left}, B_{k,right}\}_K$, where K is the number of lanes in this atomic road. We deliberately go beyond a lane width as an asymmetrical representation of lane boundary due to irregularities in the drivable areas of a lane (examples are in Figure 5.1). Here, L_k could be represented either as a continuous trajectory function or as a collection of waypoints, while discrete points $B_{k,left}, B_{k,right}$ would form an enclosed drivable area of the lane. As shown in Figure 5.3, we would start with semantic segmentation ($S(C_t)$) of the camera image, and then use particles to explore the lane over the BEV domain, which is accumulated from $S(C_t)$, L_t^*, RT_t, and M^0. To be more specific, we are looking for:

$$E_{ij} = \{L_k, B_{k,left}, B_{k,right}\}_K = G_1(S(C_t), L_t^*, RT_t, M^0). \tag{5.3}$$

The goal for intersection I_i estimation is to infer the topological relationship and geometrical reference trajectory between $E_{\cdot i}$ and $E_{i\cdot}$ as a Bezier curve $\mathbf{B}(E_{\cdot i}^k, E_{i\cdot}^l)$, where k and l denotes specific lanes in atomic roads. When two lanes are not topologically connected, the

Figure 5.4: Example results of semantic understanding module

function output is set to be null. Here, we are utilizing the result in the previously defined E_{ij} and the skeleton map M^0 for lane tracing at the intersection. Again, we study:

$$I_i = \{\mathbf{B}(E^k_{\cdot i}, E^l_{i\cdot})\}_{K\times L} = G_2(E_{i\cdot}, M^0) \tag{5.4}$$

In corresponding maps shown at the bottom of Figure 5.3, we start with a coarse map skeleton M^0, go through the road semantics E_{ij} and reference trajectory I_i generation, and end with a lane-level HD map M.

Semantic understanding of the scene

Drivable areas and lanes are defined by the road markings painted on road surfaces, but lane markings in urban areas are more complicated: there are numerous lane splitting, frequent stop lines, and broken/missing markings.

Therefore, instead of only extracting lane markings from camera images, we consider this problem as semantic segmentation and infer both drivable areas and lane markings from camera images. Built upon a DeepLab-v3+ [16] structure, we are particularly interested in ego lanes, 2 neighboring lanes on each side of the ego lane, dashed lines, solid lines, crosswalks, road curbs, and stop lines. As shown in Figure 5.4, the input of the network is an image C_t, and we predict the aforementioned 10 classes. These semantic instances form the foundation of our understanding of the current scene. Considering both lanes and lane markings is especially efficient for urban scenes as the lane markings might be missing, and some drivable areas may be confused with parking zones.

BEV accumulation and atomic road structure estimation

After the road semantics are extracted from the camera images, the segmented images ($S(C_t)$) are projected onto the ground plane in map coordinates. These projected BEV images are then accumulated together for a semantic representation of the atomic road. For the scope of this semantic mapping problem, we assume that the pose of the ego vehicle RT_t is given from auxiliary sensors like inertia-GPS navigation systems or SLAM algorithms like [164].

For the BEV projection, most previous works assume a flat ground as the road model. However, such estimation is over-simplified in city scenes. For urban areas, undulating

(a) Without LIDAR correction (b) With LIDAR correction

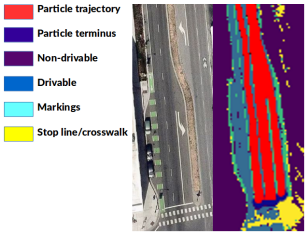
Figure 5.5: BEV projection comparison with LIDAR

road surfaces, either purposefully designed for drainage or accidentally caused by lack of maintenance, are common. Figure 5.5a shows that by simply assuming a flat surface of the road, the BEV projection could be distorted. We propose to optionally use a synchronized LIDAR scan L_t^* as the ground topography, and then project the image onto a topography mesh. To generate the ground mesh from sparse point clouds, we process the synchronized LIDAR point cloud through a Delaunay Triangulation with each point as a vertex on the mesh. After LIDAR correction, the BEV projection could be seen in Figure 5.5b, where the distorted cross-walk markings are rectified. To further demonstrate the improvement of using the true ground topography, we include an ablation study of the map quality with/without the LIDAR correction in the experiment section.

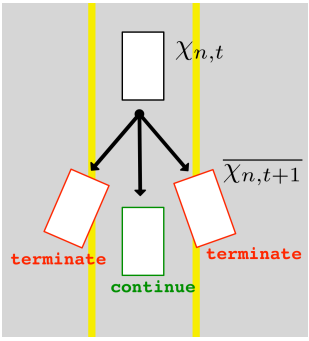
With a semantic atomic road map shown as the background of 5.6a, we now study the possible vehicle trajectories in such scenarios. To tackle the complicated, mostly irregular, driving scenes, we propose to use a Monte Carlo exploration strategy: the particle filter. Each particle represents a moving vehicle of an average sedan size with three state parameters: the BEV location and the yaw angle: $\chi_{n,t} = [x, y, \phi]_{n,t}^T$. The details of this exploration strategy are shown in Algorithm 1. With the ego vehicle starting at one end of the atomic road with RT_0, we generate a strip of N particles $\{\chi_{n,t=0}\}_N$ perpendicular to the driving direction in RT_0. Each particle $\chi_{n,t}$ then proceeds along the driving direction in the atomic lane map shown in Figure 5.6a. Here, we are simulating the actual driving of the car with speed v_m and yaw-rate ω_m uniformly sampled from noisy linear and angular velocity distribution V_m, Ω_m. The dynamic update function is

$$\overline{\chi_{n,t+1}} = \chi_{n,t} + \begin{bmatrix} \cos(\phi_{n,t} + \omega_m) * v_m \\ \sin(\phi_{n,t} + \omega_m) * v_m \\ \omega_m \end{bmatrix} \Delta t \tag{5.5}$$

Each predicted particle $\overline{\chi_{n,t+1}}$ will be re-weighted for it's overlapping ratio with the lane boundaries (Figure 5.6b), and the weight is saved as $w_{n,t+1}$. Particles will be terminated at I_j. Through the sequential Monte Carlo process, we could simulate vehicle behaviors at lane splitting scenes. For unstructured roads too narrow to fit two lanes, the particle filter also



(a) Particle history plotted next to the satellite map



(b) Prediction and evaluation step of particle filter

Figure 5.6: Particle exploration illustration

demonstrated a preferred driving trajectory of the vehicle (Figure 5.1).

For particles traveled to the end of the atomic road $\{\chi_{n,T}\}$, we first cluster these particles with the geometrical clustering algorithm DBSCAN [26] to determine the resulting number of lanes. Then we perform numerical regression to find the best representation of these lanes. After experimenting with different regression models, we conclude that the best fit would be an optimized piece-wise linear regression smoothed with a natural spline. By minimizing the sum-of-squared regression loss shown in Equation 5.6, we first determine the optimal breakpoints b_i for the piece-wise regression function $f(x_n, b_i)$, and then use the natural spline to smooth the connection for a differentiable curve representation. Such a differentiable form of reference trajectory is particularly valuable for further path-planning and prediction tasks [33].

$$L_k = f(x_n, \underset{b_i}{argmin}(\sum_{t=0}^{T}\sum_{n=1}^{N}(y_n - f(x_n, b_i))^2)) \tag{5.6}$$

Sampling waypoints for each regressed lane L_k, we estimate the lane width by probing towards the latitudinal direction of the lane. As a result, we would get accurate lane width $B_{k,left}, B_{k,right}$ at each specific sampled location. Together we form the atomic road map E_{ij} introduced in Equation 5.3 with reference trajectories L_k and their lane boundaries $B_{k,left}, B_{k,right}$. It is worth noticing that such representation could be easily transferred to a LaneLet2 [118] format for further tasks.

Intersection lane tracing

As previously introduced in Equation 5.4, the road connection topology and reference trajectories inferences inside an intersection I_i come from the OSM skeleton M^0 and the lanes generated on neighboring atomic roads $E_{\cdot i}$ and $E_{i\cdot}$. Since the OSM defines the road connecting topology, we transfer this relationship to the lane level with general traffic rules: i.e.

Algorithm 1: Lane Exploration Particle Filter

Result: Returns the regressed lane on atomic roads on E_{ij}
initialize N particles $\{\chi_{n,t=0}\}_N$ with RT_0 and size;
while $\chi_{n,t}$ *not at* I_j **do**
  $\{\overline{\chi_{n,t+1}}\}_N$, $\{w_n\}_N = \emptyset$;
  for n *in 1 to* N **do**
    sample $\overline{\chi_{n,t+1}}$ from Equation 5.5 with V_n and Ω_n;
    w_n = evaluate $\overline{\chi_{n,t+1}}$;
    $\{\overline{\chi_{n,t+1}}\}_N \leftarrow \overline{\chi_{n,t+1}}$; $\{w_n\}_N \leftarrow w_n$;
  end
  for n *in 1:N* **do**
    draw i from $\{w_n\}$;
    $\{\chi_{t+1}\} \leftarrow \chi_{i,t+1}$;
    save particle history;
  end
end
cluster $\{\chi_{n,T}\}$ in C with DBSCAN;
for c_k *in* C **do**
  $L_k = f(x_k, b_k | x_k \in c_k)$
  $B_{k,left}, B_{k,right} \leftarrow$ Longitude explore L_k
end

left lanes in $E_{\cdot i}$ would connect to left lanes in $E_{i\cdot}$; and the algorithm is demonstrated in Algorithm 2. Without explicit supervision, the topological inference would reach over 90% precision in urban areas. Based on the topology, reference trajectories are then regressed with an optimized second-order Bezier curve shown in Equation 5.7. We optimize Equation 5.7 for smoothness over the variable $\alpha \in [0, 1]$. The Bezier curve's head and tail correspond to the lane's location $L_{k_1,-1}$ and $L_{k_2,0}$ as shown in Figure 5.7, and we used intersecting point of lines extending along the direction from the head and tail nodes as control point $\mathbf{P}$ for the curve.

Here we only use this intuitive intersection exploration strategy to demonstrate the effectiveness of the road exploration module. More discussion on the intersection prediction will be carried out in Chapter 6.

$$\mathbf{B}(E_{ji}^{k_1}, E_{il}^{k_2}, \alpha) = (1-\alpha)^2 L_{k_1,-1} + 2\alpha(1-\alpha)\mathbf{P} + \alpha^2 L_{k_2,0} \tag{5.7}$$

Algorithm 2: Lane inference at an intersection

Data: E_{ji}, E_{il}: set of incoming and outgoing atomic roads to intersection I_i^0
Result: returns set of lanes I_i for intersection I_i^0
connections = ϕ
for E_{in} *in* E_{ji} **do**
  for E_{out} *in* E_{il}^0 **do**
    $N_{in} = num\_lanes(E_{in})$
    $N_{out} = num\_lanes(E_{out})$
    for k *in 1 to* $min(N_{in}, N_{out})$ **do**
      $connections = connections \bigcup \{(E_{in}^k, E_{out}^k)\}$
    end
  end
end
$I_i = \phi$
for $\{(E_{in}^k, E_{out}^k)\}$ *in connections* **do**
  $\mathbf{B}_k$ = Bezier curve from Equation 5.7
  $I_i = I_i \bigcup \mathbf{B}_k$
end
return I_i

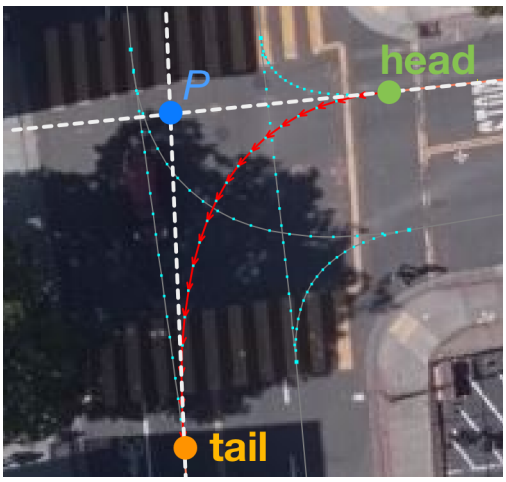


Figure 5.7: Lane inference at an intersection

Approaches	Location name	Urbanization rate	Lane density ($1/km^2$)	Sensor input	Reported Results			
					RMS Error (m)	mIOU	Precision	Recall
Mattyus et al. [94]	Karlsruhe	Low	103.5	Aerial+Cam	0.57	0.55	0.86	0.60
Meyer et al. [95]	Frankfurt	High	– –	Stereo	– –	0.58	– –	– –
Paz et al. [115]	UCSD	Medium	95.1	LIDAR+Cam	– –	0.71	0.78	– –
Joshi et al. [57]	King, MI	Low	29.2	LIDAR	0.06	– –	– –	– –
Elhousni et al. [25]	Not Known	Medium	– –	LIDAR+Cam	0.30	– –	– –	– –
Ours	Berkeley	Dense	360.9	LIDAR+Cam	0.24	0.79	0.84	0.73
				Cam	0.25	0.73	0.64	0.56
	San Francisco	Dense	259.8	LIDAR+Cam	0.33	0.76	0.63	0.63

Table 5.1: Atomic road mapping evaluation

Approaches	Location Name	Urbanization Rate	Lane density ($1/km^2$)	Sensor Input	Evaluation Metrics		
					RMS Error (m)	Precision	Recall
Geiger et al [36]	Karlsruhe	Medium	103.5	Stereo	3.00	––	0.92
Joshi et al [58]	King, MI	Low	29.2	LIDAR	0.5	––	0.96
Meyer et al [97]	Karlsruhe	103.5	Low	Simulation	0.27	––	0.85
Roeth et al [125]	Not Known	Mdeium	––	Fleet DGPS	5.2	––	––
Ours	Berkeley	Dense	360.9	LIDAR+Cam	0.24	0.91	0.80
		Dense		Cam	0.30	0.90	0.81
	San Francisco	Dense	259.8	LIDAR+Cam	0.46	0.93	0.90

Table 5.2: Intersection inference evaluation

5.4 Experiments

Experimental Setup

To validate the proposed method, we have created lane-level HD maps for 3.5 KM routes in 47 blocks of the San Francisco Bay Area. The data is from a public mapping dataset [157] with a front-view camera, LIDAR, and ground-truth ego-vehicle pose. To evaluate the atomic road and intersection reconstruction IOU, we manually label the drivable areas on Bing [98], and we perform a global rectification [136] for alignment as the map and the ground truth are generated in different coordinate systems. To quantify the trajectory deviation, we use the trajectory from another driver passing through the same scenes on a different day.

For the semantic understanding module, We adopt DeepLab-v3+ [16]. We pre-train the module with the Cityscapes dataset [23] and fine-tune with the BDD100K dataset [166]. Since the BDD100K dataset only defines ego-lane and neighboring lanes for drivable areas, we extend this label to ego-lane, left lane, second left lane, right lane, and second right lane. We use SDG as an optimizer with a learning rate of 0.01, momentum 0.9, and batch size 4. Fine-tuning is done for 250,000 steps. Images are resized to 512×512 on the semantic segmentation process and restored to their original size on BEV projection.

For BEV accumulation, we use the ground-truth localization provided in [157]. In the particle filter exploration, we deploy 500 particles, each with a length of 3m and a width of 1.5m. The velocity (m/s) is drawn uniformly from $U(0.9, 1.1)$, and the yaw angle (radians) is uniformly drawn from $U(-0.2, 0.2)$. The maximum cluster distance for DBSCAN is set to be 1m, which is around half of a typical lane in urban scenes.

Lane-level map in Berkeley and San Francisco

A qualitative atomic road module mapping result is shown in Figure 5.1: the proposed method could successfully map lane-splitting (A), un-structured roads (B), and complicated intersections (C). For quantitative studies on lane-level HD maps, we do not find a consensus in academia for map quality evaluation. However, we use all popular metrics in previous works to demonstrate the effectiveness of our algorithm: the root-mean-square (RMS) error between the reference trajectory and ground truth, the mean intersection of union (mIOU) index of the proposed lane boundary, and the precision-recall values. All the evaluations are performed on a per-lane basis. To study the detection rate of the proposed approach, we define a successful detection as one having over 0.7 mean IOU or less than 0.2m of RMS. A quantitative study of the proposed methods could be seen in Table 5.1.

Gauged at 0.7 mIOU or 0.2m RMS, our proposed method has a mean RMS of 24cm for lane center trajectory estimation, and an average IOU of 0.79 for lane boundary estimation in Downtown Berkeley. With the same threshold, we are able to achieve 0.84 in precision and 0.73 in recall. In North Beach, San Francisco, our approach has an RMS of 0.33m and mIOU of 0.76 when gauged at 0.7 IOU with 0.63 precision and 0.63 recall.

To the best of our knowledge, most previous endeavors in related fields use private data for evaluation, and we could not test our proposed approach on their dataset. More critically, most mapping-related algorithms and parameters are close-sourced, making it impossible to re-evaluate on our dataset. Thus, Table 5.1 lists the performance of other proposed methods [57, 25, 115, 94, 95] in similar matrices. We also list the urbanization rate at each location for a qualitative review of the difficulties of mapping these areas. To support the urbanization ratings, we further calculated the lane density index defined as $\#oflanes/km^2$. Higher density means that more lanes are compacted in a unit area, making the lane structure more complicated.

For the invisible topology and trajectory inference in intersections, we evaluate our performance by the topological relationship precision-recall index as well as the inference trajectory RMS error. Quantitative results for the intersection could be seen in Table 5.2. The previsision and recall are gauged at 0.1m RMSE with correct topology prediction. Compared with a series of other methods in less urbanized areas, our algorithm is still able to discover the potential topological relationship at intersections with a lower RMS error.

Ablation study: true road topography

To study the effect of the road topography on our mapping system, we compare the map generated with true road topography and the map generated with a plane assumption of the ground. A quantitative comparison is shown in Table 5.1 and 5.2. It is clear that with the LIDAR corrected topography, we have a significant improvement in atomic road detection precision and recall score. Also, the estimated drivable area is closer to our labeled ground truth.

5.5 Chapter Summary

Chapter 5 discusses the road exploration problem in the urban HD map construction process. To begin with, an OSM-based city map skeleton is introduced, and a DCG is constructed to represent the urban road network. Next, the semantic particle filter-inspired method is proposed to explore each lane in the urban road setting. The proposed method is tested on busy roads in San Francisco and Berkeley, and the result shows a robust and accurate reconstruction of the urban road map.

The work included in this chapter majorly focuses on the road module exploration, only studying the intersection topology and geometry with a naive approach. In the next chapter, we will look closely at the intersection, and see how the HD map is constructed with multiple sensors.

Chapter 6

Multi-sensor Intersection and Road Inference

According to the Federal Highway Administration, more than 50 percent of fatal and injury crashes occur at or near intersections [13]. The danger at the intersections comes from drivers' misjudgments, occlusions, etc. Naturally, intersections become the center of HD map research.

Furthermore, to acquire a comprehensive understanding of complicated scenes like intersections, autonomous agents need multiple sensors. Thus, how to leverage a multi-sensor setup efficiently for mapping purposes is another problem faced by mapping researchers.

Acknowledging the challenges at intersections and the demand for multi-sensor setups, here we design a novel sensor fusion mapping framework with the intersection in mind.

For an HD map construction at the intersection, it requires more than a simple semantic segmentation or lane tracing in the camera domain to accomplish these tasks. Instead, it is essential to form a Bird's Eye View (BEV) representation of the 3D world with information gathered from multiple sensors. In this chapter, we focus on lane geometry and topology in the BEV domain when the inputs are from multiple cameras. We firstly proposed to use a local lane map as a representation for the static scene. Then, based on different sensor fusion methodologies, we design multiple variants of the BEV segmentation network to infer the local lane map. With experimental data, we conclude that a network with pre-fused BEV image input supervised in the BEV domain is the preferred strategy for lane-level information extraction. The proposed method is tested in both simulated and real-world environments, and the results show an accurate and robust segmentation in the BEV domain[1].

6.1 Introduction

To achieve fully autonomous driving, it is critical for the perception module to understand roads and lanes. These static features serve as a foundation for scene understanding, and

[1]This chapter includes materials from the author's previous work [139]

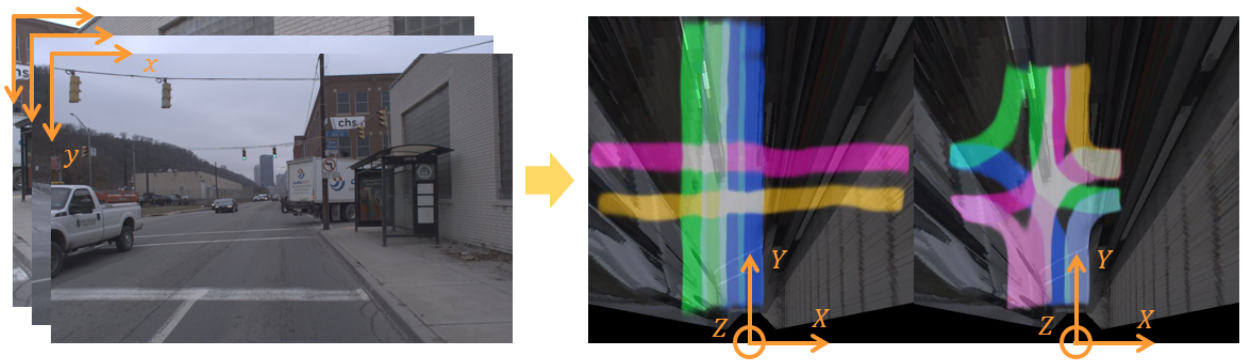


Figure 6.1: Mapping a scene between multiple cameras' frames to a BEV frame

they are also important for downstream tasks. For the past few years, researchers have demonstrated the key role of lane structures in perception [162], localization [3], and path planning [33]. In particular, researchers are looking for geometrical and topological attributes of the lanes. For example, the lane geometry contains instances like the drivable areas or centerlines, and the lane topology contains information on connection logic.

For either online lane perception or offline High Definition (HD) map generation, there exist two challenges in the camera-based lane exploration task: scene representation and coordinate transformation.

Firstly, what kind of information would suffice the aforementioned needs in scene understanding remains an open question. In the past, researchers on this topic have given various answers: derivable areas [92] [124], lane graphs [70], reference paths [117], etc. Considering the demand for downstream tasks, in this chapter, we propose a more informational representation of the scene with both geometric and topological information: a local lane map in the Bird's Eye View (BEV) as shown in Figure 6.1.

With the representation debate going on for detection tasks, the word "BEV" here triggers a harder challenge: coordinate transformation. Typical perception works generate labels in either a coordinate-agnostic frame (pixel) or a camera's ego frame [117]. However, in the static perception problem for automated vehicles, perception outputs should always be done in the BEV frame (Fig. 6.1) for the information to be useful in downstream tasks [117, 92, 124]. More critically, perception modules sometimes need multiple cameras to cover a larger field of view (FOV), thus involving the problem of sensor fusion. To tackle these challenges, we propose to use a pre-fused BEV image as direct supervision for a convolutional neural network to predict our desired class labels in the BEV domain. We also design different variations of the proposed network to further study the optimal time and input combination in the sensor fusion frameworks. Furthermore, we compare our proposed method with other state-of-the-art networks on both synthetic and real-world datasets, and the experiments show that the proposed method would accomplish an accurate and robust construction of the static scene.

In this work, we study **what** is the desired output for the mapping problem, **when** is the

prime time to fuse sensor inputs, and **how** to design a network for such a task. Our major contributions are:

- We propose a representation in local lane map of the static scene around ego vehicle and a method to predict the map from images captured by multiple cameras;
- Multiple variants of the proposed network are designed to study the performance of different sensor fusion strategies;
- The proposed network is tested against other related works on both synthetic and real-world datasets, and the results show a robust and accurate construction of the BEV lane map.

Static Scene Representation

To formulate the static scene understanding problem, it is necessary to first define the desired output. In general, such output should contain both geometric and topological information for downstream tasks.

Some works treat the road network as a graphical model for investigation. Kunze et al propose to use a lane graph to represent a road network [70], where lanes and signage are fit into a graph for logical understanding. However, such a method does not analyze the lane geometry. More recently, Wang et al model an intersection explicitly with 38 parameters and try to learn these parameters from one camera image [153]. As demonstrated in their paper, however, this method only handles simple road structures. Some other works are trying to employ a model-free method for the road network. Roddick et al [124] and Mani et al [92] use a drivable area segmentation result as a local map without topological connection. Disregarding a dense representation, Lift Splat and Shoot from Nvidia [117] as well as Paz's TridentNet [114] choose to use sampled reference trajectory as the representation. Such networks usually generate only one trajectory for the ego vehicle rather than a map for the entire environment. Lastly, Li et al [77] and Tesla [60] propose to detect only lane markings and curbs on the road surface, but it is unclear how to incorporate this information into downstream modules.

Observing the limited capacity of the model-based method and the ambiguity in the current model-free methods, we propose to use a road model consisting of directional lanes and their individual segmentation to represent the static environment.

Cross coordinate image segmentation

Different from traditional perception problems like classification [127] and semantic segmentation [23] where all predictions are made in the image or the camera frame, a useful scene understanding for the autonomous vehicle should be in the BEV frame [60]. Moreover, due to the limited FOV of a single camera, information from multiple cameras needs to be merged at wider scenes such as intersections. Mathematically, the pin-hole camera model

and epipolar constraints define such sensor fusion and coordinate transformation: one could re-project the semantic image onto the road surface through each camera's intrinsic and extrinsic matrices. Intuitive as it may seem, however, such a method assumes 1. an accurately calibrated camera rig and 2. a flat road surface. As a result, simple re-projection is rarely seen in real-world applications.

Many studies are tackling the transformation problem with multiple sensors. BevSeg [102] proposes to use both depth and semantic networks as bootstrap before the sensor fusion. A following encoder-decoder network predicts a BEV segmentation map from the fused images. A similar work from Pan et al [110] uses depth image as an additional input with similar semantic bootstrap. However, in most cases, methods with bootstrap layers in between could not be trained end-to-end. Nvidia's Lift Splat Shoot [117] embeds a depth estimation module inside the network and merges each image into the BEV frame for a comprehensive semantic segmentation. More recently, Tesla demonstrates using a variation of the transformer network [150] to query labels in the BEV domain for better segmentation [60].

Some other methods focus more on the coordinate transformation. Roddick et al [124] uses a pyramid occupancy network (PON) to dissect different parts of the image for BEV projection, and an extrinsic matrix transformation is then applied to the segmentation results. MonoLayout [92] encodes information to latent variable and predicts road layout utilizing the power of the General Adversarial Network [40]. Van Gool's team takes temporal images from a single camera and transfers its coordinate to the BEV frame inside the network [11]. However, as shown in our experiment section, such methods usually lack the accuracy on the output BEV layer. In this work, we propose to use direct supervision of pre-fused BEV maps in a semantic segmentation backbone to complete the sensor fusion and coordinate transformation task.

6.2 Representation of the Map

Observing the limitations of the aforementioned representations, we propose to use a local lane map Y_B as a mapping output to estimate the geometry and topology of lanes surrounding the ego-vehicle.

$$Y_B = \{\mathbf{y}_{B,l} | l \in L\} \tag{6.1}$$

The local lane map Y_{BEV} is a rasterized image per lane class l (Equation 6.1) in the BEV frame as shown in Fig. 6.1. Here, we focus on 20 classes, which could be broken down as follows. In the road model, there are 4 general traveling directions from the ego vehicle's perspective: the ego direction, the opposite direction, and two perpendicular directions. For each of these directions, we further assume another set of 5 possible lane structures: 3 straight lanes and 2 turning lanes (left and right). In Figure 6.1, different color pixels refers to the different segmentation result for these classes. Since there can be overlapping

semantics on roads, the local lane map is not one-hot encoded, and two or more classes are allowed to be assigned to a single pixel.

With the local lane map representation, we solve the global mapping problem in [117, 114] and the ambiguous interpretation problem in [60, 77].

6.3 Study of Fusion Strategies

Given the local lane map representation, the mapping problem can thus be regarded as a semantic segmentation task. However, segmentation between multiple coordinate systems remains as to be a challenge. In this section, we start with our proposed methods and then introduce other variations. Along with the experiment results, we later clarify how the performance varies depending on when to project to BEV, which domain to use for supervision, and how to integrate multiple camera information. Diagrams of these variants are shown in Fig 6.2.

(a) Proposed Method-BEV Input: The proposed method first projects all camera images to the BEV frame. Then, this BEV image is fed to the semantic segmentation network as shown in Fig. 6.2a and outputs a local lane map in the BEV frame. Although the input image quality is reduced by BEV projection, the network is expected to acquire knowledge of the final BEV domain, because the input and the output are sharing the same coordinate system.

The local lane map estimation discussed in this chapter ignores occlusion and treats it as if it were transparent. Thus, we assume that a road surface is a plane that is uniformly spread out under the vehicle, relying on the network for geometrical correction. With this assumption, a pixel (w_c, h_c) in the camera frame corresponds to a point $(x_B, y_B, 0)$ in the BEV frame, and can be obtained using the pinhole camera geometry in Equation 6.2, where K_c and P_c denote intrinsic and extrinsic matrix respectively. d is the depth from the camera.

$$d\,[w_c, h_c, 1]^T = K_c P_c\,[x_B, y_B, 0, 1]^T \tag{6.2}$$

For semantic segmentation, we utilize DeepLabv3+ [16] with a modified EfficientNet-B6 [141] backbone in order to align with other network variants. For the loss function selection in this network, we choose to use channel-wise binary loss shown in Equation 6.3, where y_l is the ground truth, and $\hat{y}_l$ is a prediction from the network. Compared with our selection, a cross-entropy loss is more common in segmentation tasks, and it assumes each pixel is assigned only to one class. However, such a setting is not appropriate for the local lane map since a single pixel may be assigned to multiple classes.

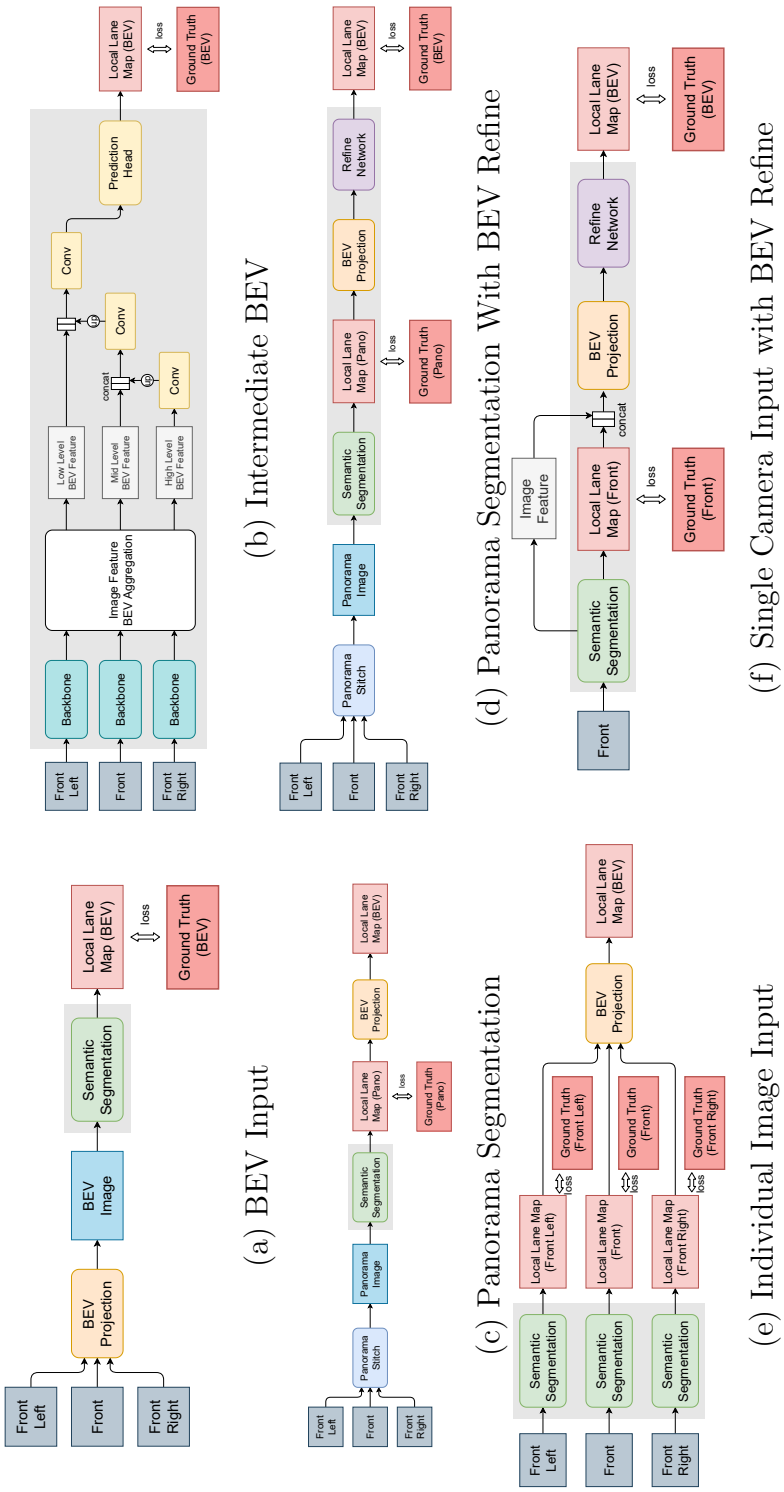


Figure 6.2: Diagrams of network variant designs

$$\mathcal{L} = \frac{1}{|L|} \sum_{l \in L} y_l \cdot \log\left(\hat{y}_l\right) + \left(1 - y_l\right) \cdot \log\left(1 - \hat{y}_l\right) \tag{6.3}$$

In the following variants, unless otherwise noted, the BEV projection, network construction, and loss function follow the same manner as mentioned here.

(b) Intermediate BEV: This model takes in individual camera images and outputs a local lane map in the BEV frame. Camera images are first fed into the shared backbone, and the network extracts image features at multiple resolutions. Then, each feature is projected from the camera domain to the BEV domain and is aggregated by the Image Feature Aggregation Module (Figure 6.3). The BEV features are then up-sampled by applying a convolution block with skipped connections. Then, the prediction head estimates the local lane map in the BEV frame. We adopt EfficientNet-B6 as the backbone, and the convolution block consists of 3 convolutional layers (3×3) with batch normalization.

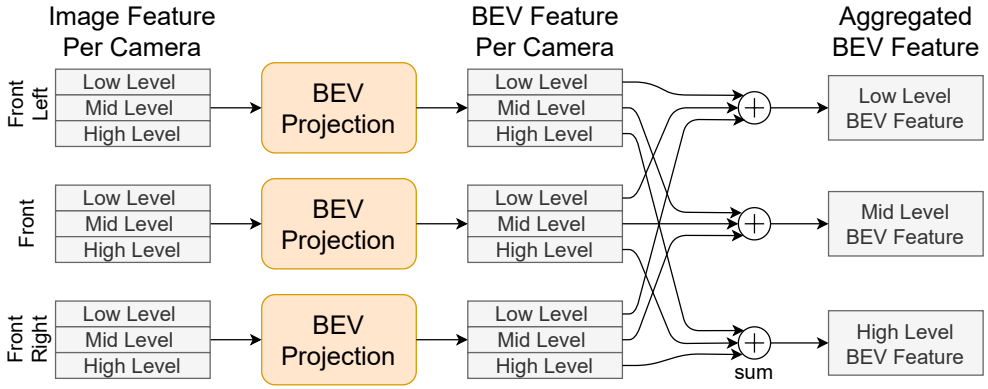


Figure 6.3: Diagram of the image feature BEV aggregation module

(c) Panorama Input: Another early fusion strategy is to use a panorama image to cover a larger FOV. This variant first stitches images into a panorama image, which is then fed into the semantic segmentation network. The output of the network is a local lane map in the panorama domain Y_P, and we acquire the BEV domain local lane map Y_B through Eq. 6.2. For panorama stitching, a method described in [138] is used with a spherical coordinate.

(d) Panorama With BEV Refine: This variant is based on (c) with an extra refinement network (Fig. 6.4) in the BEV domain. The refinement network is designed to compensate for the distortions caused by the BEV projection. Since the lane region in the local lane map has few high-frequency signals, the refinement network consists of an Encoder-Decoder instead of a skip-connection architecture. Our motivation is that the final BEV projection in (c) may reduce the accuracy of Y_B, which can be compensated

by the refinement layer in this modified design. This variant supervises in both the panorama and the BEV domain, and the total loss is a weighted sum of the loss of each domain.

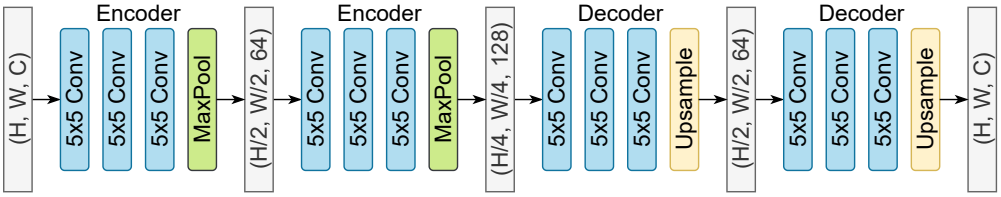


Figure 6.4: Network architecture of the BEV refinement network

(e) Individual Image Input: In addition to the multi-camera input, we also design a network with individual images as a baseline for comparison. Each camera image is fed into the semantic segmentation network individually for a prediction in the camera domain. The outputs are then transferred to the BEV domain through Equation 6.2. This variant shares the same network structure with (c).

(f) Single Camera Input with BEV Refinement: In this variant, we add the aforementioned refinement network to the single image network (e). This design can be regarded as a simplified version of Van Gool's method [11] with a new backbone design. Here, vehicles prediction and temporal accumulation in the original method are ignored for a fair comparison. We follow [11] to add concatenated camera features at BEV projection.

6.4 Experiments

For the experiments of the proposed variants and related works, we use both a synthetic dataset generated from the CARLA simulator and a real-world dataset Argoverse.

CARLA [24] is a popular open-sourced simulator for autonomous driving. In this chapter, we adopt CARLA to simulate an ideal environment with no vehicles or pedestrians. Three cameras with 60 degrees horizontal FOV are mounted 2 meters above the center of the vehicle at a 60-degree interval (as shown in Figure 6.5). From all possible intersections in each world (Town01 to Town05) in CARLA, the data is generated around the intersection at 2-meter intervals. Ground truth labels are generated from the HD Map and ego vehicle localization. For labeling consistency, the data of roundabout and highway interchanges are excluded.

Argoverse dataset [14] is a pubic dataset that contains HD Map and perception data. We follow the original train/validation splits of the dataset, and the ground truth label is generated by projecting lane information from the HD Map. Input and output image size is set to 568×568 to fit the backbone network. The resolution of the local lane map is set at 0.1m.

Methods	Fusion Timing	CARLA			Argoverse		
		Straight	Intersection	Average	Straight	Intersection	Average
MonoLayout [92]	NA	0.190	0.423	0.412	0.180	0.302	0.294
PON [124]	NA	0.711	0.520	0.634	0.659	0.450	0.575
Lift Splat Shoot [117]	mid	0.404	0.391	0.399	0.307	0.248	0.255
(a) BEV Input	early	**0.848**	**0.710**	**0.793**	0.665	**0.525**	**0.609**
(b) Intermediate BEV	mid	0.766	0.625	0.709	**0.668**	0.514	0.607
(c) Panorama Input	early	0.772	0.551	0.683	0.605	0.448	0.541
(d) Panorama with BEV Refine	early	0.794	0.586	0.711	0.651	0.514	0.580
(e) Individual Image Input	Late	0.687	0.459	0.596	0.534	0.366	0.467
(f) Single Camera Input with BEV Refine	NA	0.721	0.501	0.633	0.633	0.459	0.563

Table 6.1: Evaluation result for CARLA and Argoverse

For evaluation metric, we adopt Intersection over Union (IOU) following prior works [124, 171, 92]. Since the output of each class is a probability, the IOU is calculated using 0.5 as the threshold. We evaluate the performance with the average IOU of the straight lane classes, the average IOU of intersection turn classes, and the overall mean IOU.

We train all networks on NVIDIA TITAN X with PyTorch [113]. All training parameters are kept the same across the networks: we train for 50 epochs with a batch size of 2 and a learning rate of 0.001. The loss weight ratio between the BEV refine layer and the panoramic/single image layer is 40:1. Scores are smoothed using an exponential moving average with a window size of 4 among the last 10 epochs, and the best epoch is chosen for evaluation.

We also evaluate the performance of other state of the art networks: Lift Splat Shoot [117], PON [124], and MonoLayout [92]. Quantitative evaluation results are shown in Table 6.1, and visualization results of the inference are shown in Figure 6.5, and 6.6.

As a discussion, we first study the impact of the input format. Compared with methods using panoramic images (c) or vanilla RGB images (variant (b), (e), Monolayout, PON, LSS), BEV image (a) input has the best performance in terms of the IOU metric. Furthermore, when adding BEV as a refinement step in the network, as shown in the comparisons of (d) to (c) and (f) to (e) in Table 6.1, BEV can boost the performance of the original network. Such improvement in performance comes from juxtaposing input and prediction in the same coordinate system. As illustrated in Figure 6.8, even though the input BEV image of method (a) is distorted due to imperfect calibration, the network learns implicit knowledge of the road structure: e.g. roads are drawn straight, and intersections are perpendicular. The bonus performance of training under the same coordinate could also be observed in camera domains: the variants without inference in the BEV domain (variant (c) and (f)) show a high IOU score in their own output domains. The average IOU for the Argoverse dataset of variant (c) is 0.638 in the panorama domain, which is higher than the highest BEV domain score of variant (a). An inference result is shown in Figure 6.7.

To study the effect of the fusion timing, we have classified the timing of sensor fusion into three categories: early, mid and late. Early fusion represents methods where the input of the network is a fused-up image, regardless of the fusion domain. Direct BEV input and panorama images are typical early fusion methods. Mid-network fusions refer to methods that combine sensor information in the middle of the network ([117] and (b)). Lastly, there are late fusions where sensor fusion is outside the network after a prediction is made on images. As shown in Table 6.1, it is clear that an early fusion strategy will greatly enhance the network performance. Such improvement is likely caused by the errors in the intrinsic/extrinsic matrix, where the network could absorb and learn noisy input images for a more regularized output.

Lastly, we study the effect of bootstrapping in network design. In the past few years, a typical method of constructing cross-coordinate networks is through bootstrapping, meaning that there would be an intermediate layer of output with physical meanings. For example, Lift Splat Shoot [117] bootstraps on a depth image, and monolayout [92] bootstraps on static and dynamic detection. However, as compared with bootstrap-free methods (variants (a)-(f))

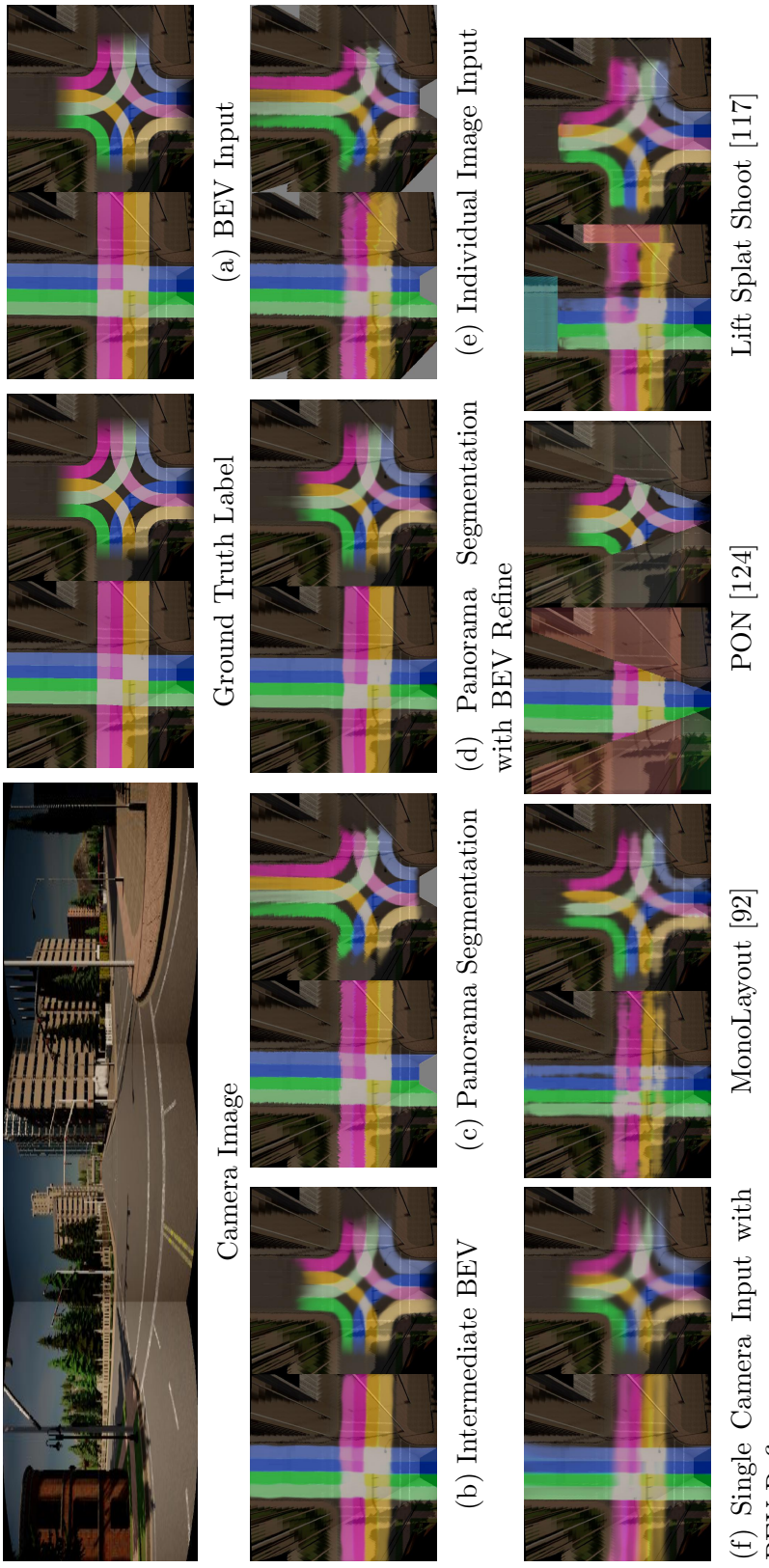


Figure 6.5: Prediction results on the CARLA dataset.

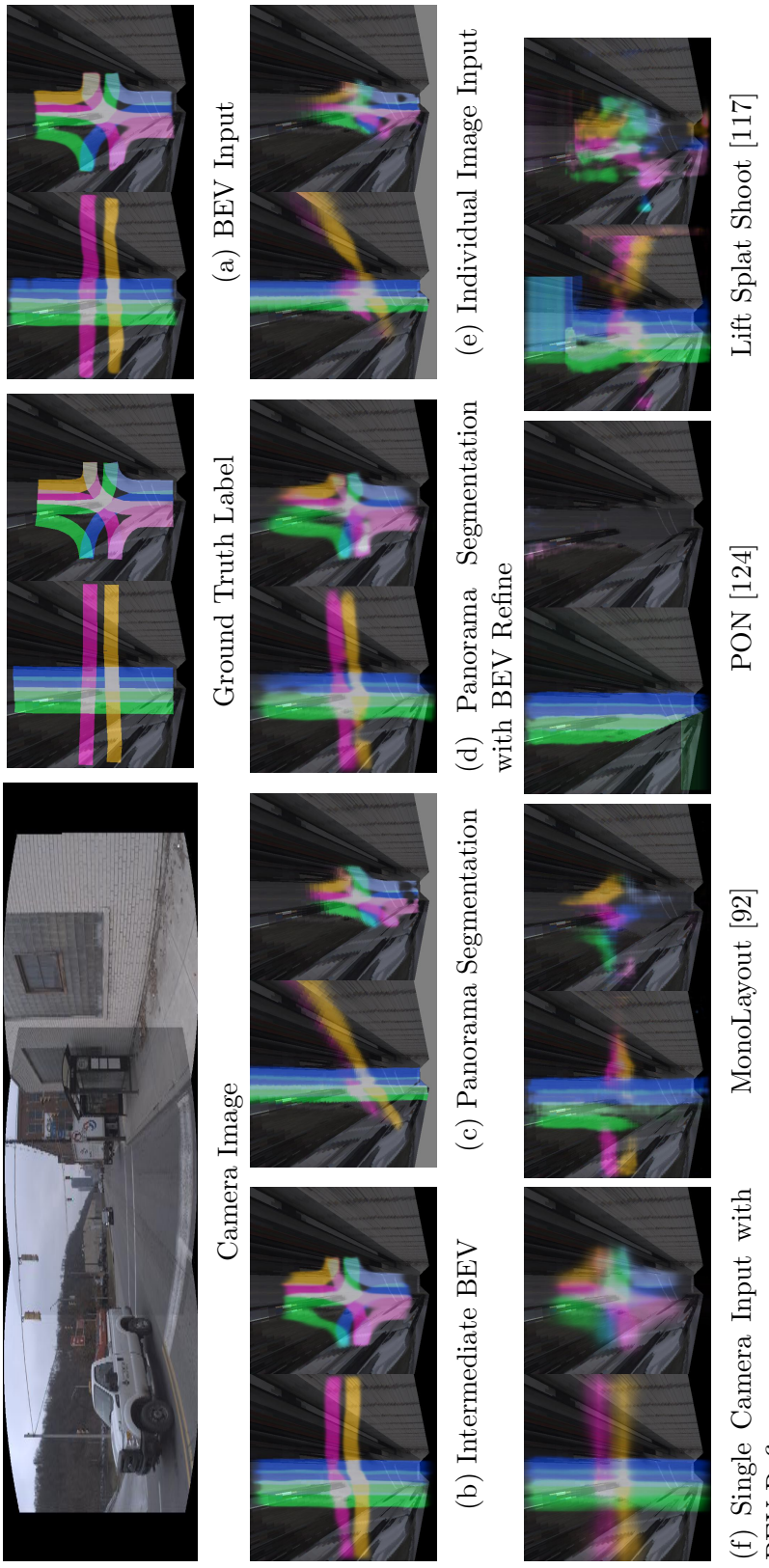
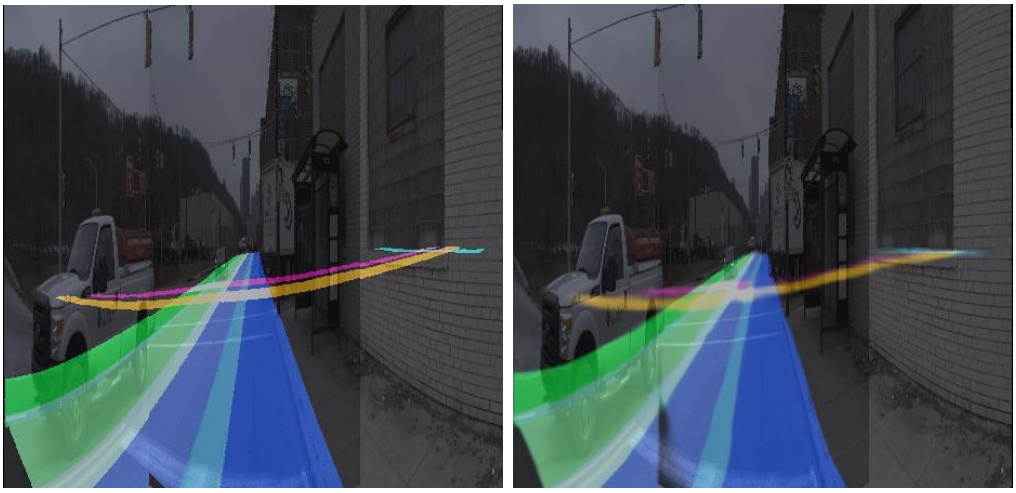
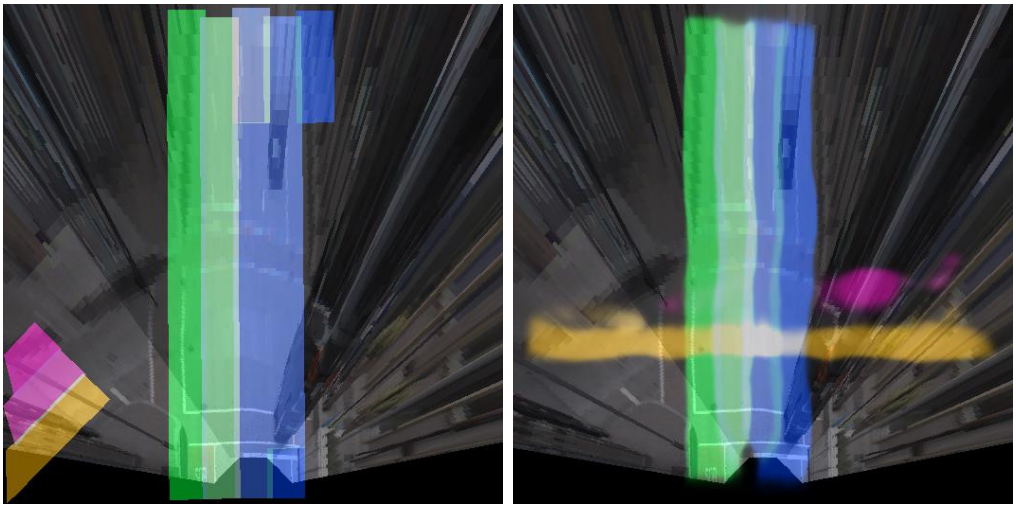


Figure 6.6: Prediction results on the Argoverse dataset



Ground Truth Inference

Figure 6.7: Loss function domain's impact on performance



Ground Truth Inference

Figure 6.8: Road structure learned in the BEV domain

and PON, bootstrapping decreases the performance on static road elements. In particular, Lift Splat Shoot and (b) Intermediate BEV resembles in network structure without specific depth supervision, but (b) out-performs LSS in the IOU metric.

Since the noisy extrinsic matrix will greatly change the result of the cross-view prediction, we extend our experiment with an ablation study with noises added to the extrinsic at testing time. We add uniform translation noise within 5 cm and a uniform rotational noise within 1 degree and the impact on performance could be seen in Table 6.2. By adding the BEV supervision, the performance of (d) is more resistant to noises, even surpassing the single image benchmark in (e).

Methods	orig. mIOU	Impact
Lift Splat Shoot [117]	0.399	-3.2%
(a) BEV Input	0.793	-10.19%
(b) Intermediate BEV	0.709	-7.53 %
(c) Panorama Input	0.683	-21.95 %
(d) Panorama with BEV Refine	0.711	-4.99%
(e) Individual Image Baseline	0.543	-6.15%

Table 6.2: Impact of noisy calibration

6.5 Chapter Summary

This chapter focuses on the representation and strategies for the BEV lane structure perception. To answer the what, when, and how question raised in the Introduction section, we propose to use a **local lane map** to convert the perception problem into semantic segmentation. Furthermore, we design and verify that a **early fusion** with **direct BEV supervision** is the optimal strategy among other variants. These proposed variants and the state-of-the-art methods are compared to consolidate our findings.

Chapter 5 and Chapter 6 mainly study the static elements on the road. However, raw data contains both dynamic and static entities, and it would be wasteful if we only utilize the static entities on the road. Indeed, moving objects on the road are particularly useful for detection teams. To exploit the full potential of the raw data, we will move on to multi-task learning in the next chapter.

Chapter 7

Multi-task Learning, Mapping with Perception

Data is valuable. When a mapping team gathers data from raw sensors, it is natural to exploit the limited data for more tasks. In urban scenes, beyond static road elements, dynamic objects are of interest for perception teams as well. Furthermore, understanding the movement of moving cars, cyclists, and pedestrians may provide insights into higher layers of the HD map hierarchy.

The materials covered in this chapter show that it is possible to perform all perception tasks and some mapping tasks via a simple and efficient multi-task network. The proposed network, LidarMTL, takes raw LIDAR point clouds as inputs and predicts six perception outputs for 3D object detection and road understanding. The network is based on an encoder-decoder architecture with 3D sparse convolution and deconvolution operations. Extensive experiments verify the proposed method with competitive accuracies compared to state-of-the-art object detectors and other task-specific networks. LidarMTL is also leveraged for online localization[1].

7.1 Introduction

Reliable traffic object detection and road understanding near the ego-vehicle are fundamental perception problems in autonomous driving. Movable objects are often perceived at the instance level with class labels and bounding boxes, or at the 2D image pixel or 3D point level depending on sensing modalities. A comprehensive road understanding requires the perception algorithm to identify drivable areas, lane markings, and road shapes, to name a few. Furthermore, all these perception tasks need to run accurately and quickly for online deployment. However, most existing methods, especially those using deep learning approaches, focus on improving each task separately, with task-specific network architectures and evaluation metrics. This task-specific solution is inefficient when dealing with multiple tasks.

[1]This chapter includes materials from the author's previously published work [30]

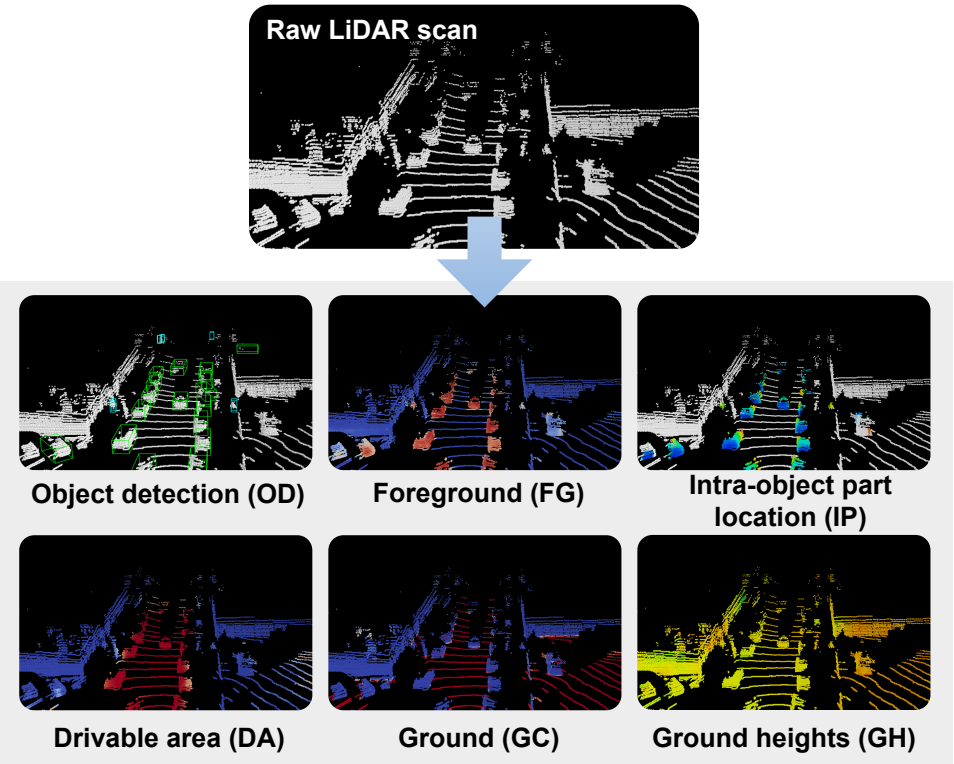


Figure 7.1: The proposed multi-task network, LidarMTL

While high inference speed might maintain via parallel computing, the memory footprints and computation costs scale linearly with the number of networks, which quickly become infeasible with limited hardware resources.

Multi-Task Learning (MTL) provides a strategy to largely reduce memory footprints and computation costs by performing all tasks via a unified model in one forward pass [149]. In deep learning, MTL means learning the shared representation of multiple tasks, typically via an encoder-decoder network architecture.

MTL is applied to 2D object detection and road understanding using RGB camera images [144, 120, 17] and has been recently introduced to 3D perception using LIDAR point clouds [162, 80, 160].

In this chapter, we propose a LIDAR-based multi-task learning network called LidarMTL to jointly perform six perception tasks for 3D object detection and road understanding, as shown in Figure 7.1. Objects are detected with class labels and 3D bounding boxes (task OD). Furthermore, their associated LIDAR points are segmented as foreground (task FG), and their relative locations to object centroids are regressed (task IP). Road perception includes point-wise drivable area and ground area classification (tasks DA and GC) as well as ground height estimation (task GH).

The benefits of those perception tasks have been studied in previous works. For example, FG and IP are leveraged to refine bounding boxes in the second stage of a two-stage object detector [134], as they provide useful point-level semantic and geometrical information about objects. GC and GH are used to remove ground [78] or normalize the heights of LIDAR points [80], especially when the ground is not flat. DA provides strong road priors to reduce false-positive predictions in object detection [162] and motion forecasting [14]. Unlike previous works which explore each perception task separately, we show that it is possible to perform all perception tasks efficiently and accurately via a unified network. Besides, previous works such as [134, 80] focus on how to employ one or several perception tasks as auxiliary tasks to support the target task, without analyzing the performance of those auxiliary tasks. In this work, we consider each perception task of equal importance and conduct comprehensive experiments to analyze their performance in single-task and multi-task settings.

In principle, the LidarMTL network works by adding task-specific heads to a 3D UNet architecture and training the full network with a multi-task loss in an end-to-end manner. UNet is a well-performed encoder-decoder network widely applied to 2D image segmentation. Following [134], we extend UNet to efficiently process 3D LIDAR points represented as voxels with 3D sparse convolution and deconvolution operations. The resulting network has only 6.5M trainable parameters and runs at an inference speed of 30FPS on a Titan RTX GPU, which is 2× smaller and 6× faster than performing all tasks sequentially using task-specific networks. Extensive experiments on the Argoverse Dataset [14] show that the LidarMTL network achieves competitive accuracies compared to state-of-the-art object detectors and other task-specific networks. The network is also employed to substantially improve online localization.

LIDAR-based Object Detection

LIDAR point clouds are usually represented by 2D projected images [163, 19], raw LIDAR points [133, 159], and voxels [169]. Compared to the other methods, voxel representation can not only be processed efficiently using 3D sparse convolution [161] but also preserve approximately similar information to raw point cloud with a small voxel size. Therefore, voxel-based backbone networks have been widely applied to learn LIDAR features in conjunction with a 2D CNN detection head [161, 134, 132, 47]. A special case is PointPillars [73], which efficiently processes LIDAR points by vertical 3D columns called pillars. Our proposed LidarMTL network follows this "voxel-based backbone + 2D CNN detection head" pipeline to perform object detection.

Road Understanding

Understanding the 3D road information online is crucial for safe autonomous driving, especially when HD maps are not available. A variety of methods have been proposed for online mapping, such as road area classification [10, 28], lane/boundary detection [1, 79],

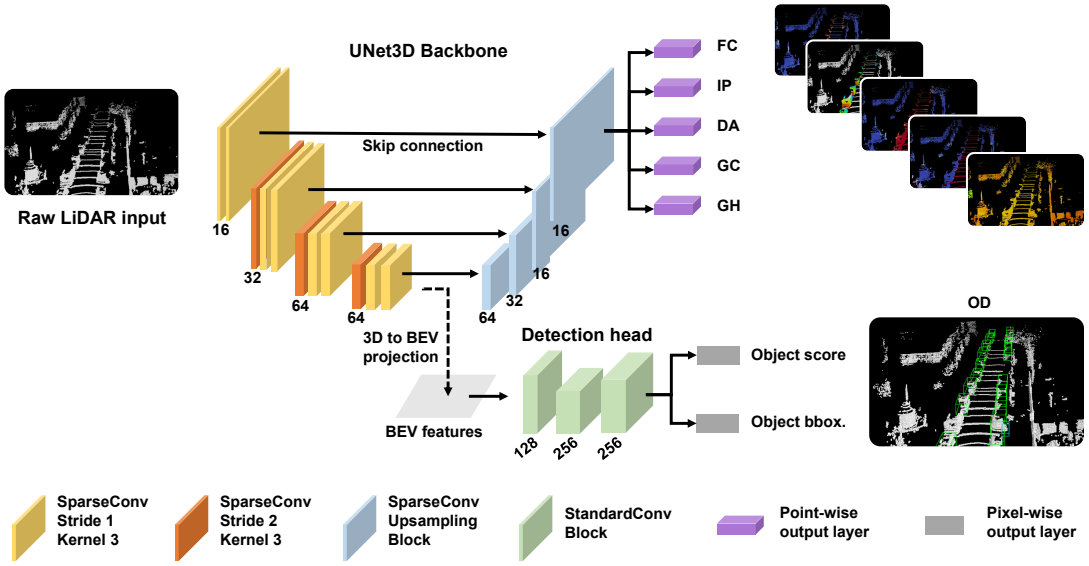


Figure 7.2: The LidarMTL network architecture

ground plane estimation [18], road topology recognition [4, 108, 171], and road scene semantic segmentation [99, 158, 124, 139]. In [160], a multi-task network is designed for multiple object-free road perception tasks, including drivable area classification, road height estimation, and road topology classification.

Joint Object Detection and Road Understanding

Existing methods usually follow the hard parameter sharing scheme [149], where networks consist of a shared encoder and several task-specific decoders. MultiNet [144] jointly performs object detection, street recognition, and road area classification. It is built by a large 2D CNN encoder based on the VGG16 or ResNet backbones, followed by task-specific branches with several convolution layers. DLT-Net [120] follows a similar architecture for object detection, road area classification, and lane detection. Besides, HDNet [162] and MMF [80] propose to use drivable road maps or ground heights as auxiliary inputs for LIDAR-based object detectors to improve the detection accuracy by adding road priors. Our proposed LidarMTL also uses hard parameter sharing: a detection head and a decoder with sparse deconvolutions are added to the encoder for object detection and point-wise predictions, respectively.

7.2 Multi-task Formulation

Task Definition

The proposed LIDAR-based multi-task learning network, LidarMTL, jointly performs six perception tasks via a single feed-forward pass, namely, 3D object detection (OD), foreground classification (FG), intra-object part location regression (IP), drivable area classification (DA), ground area classification (GC), and ground height regression (GH). The method is developed based on the Argoverse dataset [14], because to our knowledge it is the only public dataset that provides both dynamic object labels and static map information with ground heights and drivable areas.

More specifically for these tasks, LIDAR points within the bounding boxes of the dynamic objects are regarded as foreground. Intra-part object locations are defined as the positions of *foreground* points relative to their corresponding object's centroids. Driveable areas are *object-free* regions that could be driven by vehicles. Ground height estimation is performed both for ground areas and non-ground areas (such as foreground and buildings).

Overview

We aim to design a simple and efficient multi-task network for joint 3D object detection and road understanding. In this regard, the LidarMTL is based on the voxelized LIDAR point cloud representation and the UNet backbone with 3D sparse convolution and deconvolutions (we name the model "UNet3D"). Figure 7.2 shows the network architecture. The 3D space is voxelized into regular voxels, with no-empty voxels being encoded with LIDAR features. The voxelized LIDAR point cloud is processed by the UNet backbone network with the encoder-decoder architecture [126]. The encoder consists of several 3D sparse convolutions and downsamples the input spatial resolution by 8 times to extract high-level LIDAR features. The decoder gradually upsamples the LIDAR features to the original spatial resolution via 3D sparse deconvolutions. We choose the UNet backbone network and voxelized LIDAR representation following the idea from [134]. The network well-preserves the geometric information of LIDAR points by setting a proper voxel size and has been shown in [134] to achieve higher efficiency than the raw point-based methods (such as PointRCNN [133]).

Point-wise predictions are made by adding output layers directly to the decoder network, including tasks FG, IP, DA, GC, and GH. To perform object detection (OD), the 3D LIDAR features from the encoder are projected onto the Bird's Eye View (BEV) and then processed by a detection head with several standard 2D convolution layers for classification score prediction and bounding box regression. Note that employing 2D CNN on LIDAR BEV features is a common way to do object detection [161, 73, 132]. Besides, it is found more effective than performing object detection from the decoder network.

Input and Output Representation

Given an LIDAR scan, let y be the target output. The input features of each voxel are encoded as the mean values of the LIDAR point positions in the LIDAR coordinate system. The perception tasks FC, DA, and GC are formulated as the point-wise binary classification problems, with their labels $y_{\text{FG}}, y_{\text{DA}}, y_{\text{GC}} = 1$ indicating positive samples, and 0 negative samples. The tasks IP and GH are formulated as the point-wise regression problems, with $y_{\text{IP}} = [x', y', z']$ a continuous vector indicating 3D LIDAR point locations relative to their corresponding object centroids, and y_{GH} the ground heights. As for OD, the label y_{OD} consists of object classes y_{cls}, and bounding box regression variables y_{bbox}, i.e. $y_{\text{OD}} = [y_{\text{cls}}, y_{\text{bbox}}]$. Bounding boxes are parameterized by $y_{\text{bbox}} = [\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h, \Delta\theta]$, with $\Delta x, \Delta y, \Delta z$ being the residual centroid 3D positions, $\Delta l, \Delta w, \Delta h$ the residual length, width, and height, and $\Delta\theta$ the residual orientation relative to pre-defined anchors. The network makes predictions for $y_{\text{FG}}, y_{\text{DA}}, y_{\text{GC}}, y_{\text{cls}}$ via the softmax function, and directly regresses the bounding box parameters. Following [134], y_{IP} are normalized to be within $[0, 1]^3$ and are predicted by the softmax function as well, as this encoding strategy is found more stable than direct regression.

UNet3D Backbone

As shown in Figure 7.2, the encoder in the backbone processes the voxelized LIDAR point cloud by four stages of 3D sparse convolutions with increasing feature dimensions $16, 32, 64, 64$. The network downsamples the spatial resolution by 8 times through three sparse convolution layers [161] with stride 2, each followed by two submanifold sparse convolution layers [42] with stride 1. The decoder consists of four upsampling blocks with decreasing feature dimensions $64, 32, 16, 16$ and strides $2, 2, 2, 1$. In each block, the features from the previous block are combined with the skip-connected features from the encoder via concatenation and are further processed by a submanifold sparse convolution layer and a sparse inverse convolution layer, in order to upsample the spatial resolution. All convolution and deconvolution layers in the backbone have a kernel size of $3 \times 3 \times 3$. Finally, task-specific $1 \times 1 \times 1$ sparse convolution layers are added to the last upsampling block for point-wise predictions.

Detection Head

The detection head projects the 3D LIDAR features from the UNet3D encoder to the Bird's Eye View (BEV) and processes the BEV features through three 2D convolution blocks. The first block consists of six standard convolution layers with feature dimension 128 and stride 1. The second block increases the feature dimension to 256. It downsamples the spatial resolution by a convolution layer with stride 2, stacked with five 2D convolution layers with stride 1. The last block is an upsampling layer with dimension 256 and stride 2. All convolution layers in the detection head have a kernel size of 3×3. The classification scores and bounding boxes are predicted by the output layers with 1×1 convolution.

Similar to [161], the object detector regresses residual bounding box parameters relative to the pre-defined 3D anchors with fixed sizes, because objects from the same category are of approximately similar sizes. For each pixel and object category, we place two anchors with rotations of 0 and 90 degrees, with their sizes being the mean values from all ground truths in the Argoverse dataset.

Joint Training

The full network is trained end-to-end via a multi-task loss function. Denote L as a loss function. For an input data frame, the multi-task loss function, L_{MTL}, is formulated as a weighted sum of the task-specific losses:

$$L_{\text{MTL}} = \sum_{i \in \{\text{OD,FG,IP,}\atop \text{DA,FC,GH}\}} w_i L_i, \tag{7.1}$$

where w_i and L_i represent the task-specific loss weights and loss functions, respectively. To learn $y_{\text{DA}}, y_{\text{GC}}, y_{\text{IP}}$, we use the standard cross entropy loss. As for y_{FG} and y_{cls}, we use the focal loss [81] due to the large positive-negative sample imbalance problem. Finally, y_{GH} and y_{bbox} are learnt by the standard L_1 loss.

The loss weight w_i controls the influence of a task. It can be pre-defined through grid search or be optimized by task balancing approaches [149]. In this work, we employ the uncertainty weighting strategy proposed by Kendall *et al.* [21]. It uses the task-dependent uncertainty, parameterized by the noise parameter σ, to balance the single-task losses. Such noise parameters are jointly optimized during training, resulting in an adaptive multi-task loss function $L_{\text{MTL}}^{\text{adaptive}}$ written as:

$$L_{\text{MTL}}^{\text{adaptive}} = \sum_{i \in \{\text{OD,FG,IP,}\atop \text{DA,FC,GH}\}} \frac{1}{2\sigma_i^2} L_i + \frac{1}{2} \log \sigma_i^2 \tag{7.2}$$

7.3 Experiments

The experimental results are structured as follows. First, we evaluate the performance of each perception task separately. We compare the proposed multi-task network with single-task networks and show its benefits in achieving on-par performance with task-specific networks but with substantially lower memory footprints and higher inference speed. Afterward, we conduct ablation studies regarding the number of tasks and the loss weights, and we further test the network's robustness with downsampled LIDAR points. Finally, we demonstrate that our proposed multi-task network provides useful semantics that could largely improve online localization.

Experimental Setup

Dataset

All experiments are conducted on the Argoverse 3D Tracking Dataset [14], which is recorded in Miami and Pittsburgh in the USA under various weather conditions and times of the day. The dataset provides 3D bounding boxes and tracks annotations, with RGB images from seven cameras, LIDAR point clouds from two 32-beam Velodyne LIDAR sensors, as well as HD maps annotating drivable areas, ground heights, ground areas, and centerlines.

For the object detection task, we focus on the "VEHICLE" and "PEDESTRIAN" classes. For the point-wise perception tasks, we prepare the ground truth labels for each LIDAR point. The data is recorded in sequence with lengths varying from 15 to 30 seconds (10 Hz). To reduce the sequential dependency between frames, we down-sample the dataset by a factor of 5.

The resulting dataset we use contains 2609 training frames and 996 evaluation frames, with over 20K VEHICLE and 6.7K PEDESTRIAN objects.

Implementation Details

All networks are trained with the same optimization settings from scratch up to 80 epochs. The ADAM optimizer is used with an initial learning rate of 0.01, a step decay of 0.1, and a batch size of 4.

For a fair comparison with state-of-the-art object detectors (such as PV-RCNN [132] and PointPillars [73]), which only process LIDAR point clouds on the camera front-view, we extract LIDAR point clouds corresponding to synchronized front-view images from the original Argoverse dataset, and train the front-view networks for most experiments. In this regard, we use the LIDAR point cloud within the range length $\times$ width $\times$ height $= [0, 70.4] \times [-40, 40] \times [-1.5, 4, 0]$in meters, and do discretization at 0.1 meter voxel resolution. Besides, to employ our proposed LidarMTL network in online localization, we train a full-range network that processes LIDAR point cloud within the range $[-70.4, 70.4] \times [-70.4, 70.4] \times [-1.5, 4, 0]$ in meters. All experiments are conducted using a Titan RTX GPU. The inference time for the front-view LidarMTL reaches 30FPS and for the full-range LidarMTL 7.7FPS.

Performance Evaluation

Object Detection (OD)

We evaluate the object detection performance using the standard Average Precision for 3D detection (AP_{3D}) and on the Bird's Eye View (BEV) (AP_{BEV}). Both parameters are measured at the Intersection Over Union IOU=0.7 threshold for "VEHICLE" objects and IOU=0.5 for "PEDESTRIAN" objects, respectively, as suggested by [38]. The IOU scores in object detection are geometric overlap ratios between bounding boxes, and they measure the localization accuracy. We report the AP scores as a triplet with respect to increasing

Table 7.1: A comparison of Object Detection (OD) performance, the number of trainable parameters and inference speed

Methods	VEHICLE		PEDESTRIAN		$\text{mAP}_{BEV}(\%)$	$\text{mAP}_{3D}(\%)$	Trainable param. (M)	Inference speed (FPS)
	AP_{BEV} @0.7(%)	AP_{3D} @0.7(%)	AP_{BEV} @0.5(%)	AP_{3D} @0.5(%)				
PV-RCNN [132]	77.5, 62.0, 21.1	63.2, 38.0, 3.8	51.8, 26.6, 4.5	45.7, 22.2, 3.0	56.1	43.2	13.10	14.6
PointPillars [73]	75.3, 57.2, 16.6	53.5, 27.8, 2.7	37.4, 22.3, 4.0	30.3, 16.8, 2.3	51.5	35.3	4.82	71.5
Second [161]	72.0, 53.9, 14.1	50.9, 25.0, 1.9	41.1, 22.8, 5.0	33.6, 17.5, 2.6	49.7	35.1	5.31	54.2
UNet3D	73.0, 35.9, 4.4	50.9, 13.7, 0.5	56.7, 25.1, 2.9	44.4, 17.4, 1.5	46.4	32.4	1.90	33.2
LidarBEV	71.8, 56.1, 14.0	50.3, 23.8, 1.7	42.0, 22.9, 4.4	36.6, 16.3, 2.1	49.9	34.6	5.31	59.6
LidarMTL	72.9, 56.9, 14.1	53.4, 24.3, 1.8	40.6, 22.9, 6.1	33.3, 17.0, 4.2	49.8	35.0	6.52	30.0

LIDAR ranges ($0-30$m, $30-50$m, and $50-70$m), as well as the mean AP scores, mAP, by averaging over all distances and object classes (similar to [27]). Besides, we report the number of trainable parameters and the inference speed for each object detector. Table 7.1 summarizes the results.

The proposed multi-task network (LidarMTL) is compared against several LIDAR-based object detectors. The LidarBEV network follows the same detection architecture as LidarMTL (Encoder with sparse 3D convolution + BEV detection head with 2D convolution). It serves as the baseline to study the object detection performance when introducing multiple tasks. The UNet3D network directly employs the encoder-decoder architecture from LidarMTL to predict object classes and bounding boxes on each LIDAR point (without BEV detection head and pre-defined anchors), and is used to verify the network architecture design. Furthermore, we re-train state-of-the-art detectors, PV-RCNN [132], PointPillars [73], and Second [161], using our experimental setup. Note that PV-RCNN is a two-stage object detector, whereas all other detectors are one-stage. UNet3D directly regresses bounding box parameters, whereas the others are based on pre-defined anchors and BEV detection heads.

As Table 7.1 illustrates, the proposed LidarMTL network achieves similar detection accuracy to LidarBEV, SECOND, and PointPillars, with comparable number of parameters (6.52M) and reasonable inference speed (30FPS). PV-RCNN has the highest AP scores compromised by over $2\times$ more parameters and computation cost compared to LidarMTL. Though UNet3D has only 1.9M parameters, it has the worst detection accuracy with $2-3\%$ smaller mAP scores compared to LidarMTL, indicating the importance of adding anchor priors and BEV detection heads for precise object detection. In conclusion, the proposed LidarMTL shows competitive detection performance to other detectors in prediction accuracy, model size, and inference speed.

Foreground (FG), Drivable Area (DA), and Ground Classification (GC)

We evaluate the foreground, drivable area, and ground classification tasks using the Average Precision (AP), Intersection Over Union (IOU), and classification accuracy scores at a probability threshold of 0.5. Those evaluation metrics measure the classification performance at each LIDAR point and have been used as the standard metrics for road detection [31] or semantic segmentation [27]. Note that unlike the IOU metric for object detection in the previous section, here an IOU score is measured by $\text{IOU} = 100 * \text{TP}/(\text{TP}+\text{FP}+\text{FN})$ according to [27], with TP, FP, FN being the number of points categorized as true positive, false positive, and false negative samples. For each task, a task-specific UNet3D network is trained to compare with the LidarMTL network. Furthermore, since these point-wise perception tasks can be regarded as semantic segmentation, we additionally train two state-of-the-art semantic segmentation networks, namely, RangeNet++ [99] and SqueezeSegv3 [158], to perform foreground and ground classification. We do not conduct experiments for drivable area classification, because it is not mutually exclusive with the other two tasks.

Experimental results are shown in Table 7.2, Table 7.6, and Table 7.4. The proposed multi-task network achieves on-par performance with the single-task network, with less than

1% difference in all evaluation criterion. The network also shows competitive results with RangeNet++ and SqueezeSegv3, verifying the effectiveness of the network architecture design.

Ground Height Estimation (GH)

We evaluate the ground height estimation performance using the Root Mean Squared Errors (RMSE) and Mean Average Errors (MAE) metrics, which are widely used in the depth prediction task for RGB camera images [147]. Table 7.3 reports the performance for all LIDAR points, grouped with respect to the LIDAR ranges ($0-30$m, $30-50$m, $50-70$m). The LidarMTL network is compared with the task-specific UNet3D network, as well as a simple heuristic that assumes a ground plane given the ego vehicle's pose information. Note that the ground plane assumption has been widely used to remove ground LIDAR points for object detection [68, 18].

Table 7.3 shows that the ground plane method results in larger errors at longer distances, indicating that the ground is not flat. The errors produced from the UNet3D and LidarMTL networks are much smaller than the ground plane method (over 40% RMSE reduction for all points), showing the benefits of the point-wise ground height estimation. The LidarMTL network produces slightly larger errors than the UNet3D network ($<$ 2cm), showing small negative transfer phenomena often seen in multi-task learning [149].

A more interesting experiment is to evaluate the ground height estimation for LIDAR points that belong to objects. Such information could be used to normalize objects' heights and has the potential to improve detection performance, as shown in [80]. Table 7.5 shows that both networks predict ground heights accurately, with RMSE errors smaller than 20cm even at $50-70$m range.

Intra-object part locations (IP)

Finally, we evaluate the performance for intra-object part location predictions, with the same evaluation metrics (RMSE and MAE) used in the previous sections. Both LidarMTL and UNet3D networks perform similarly, with the LidarMTL network producing slightly smaller errors (<1) at long distances ($50-70$m) than the UNet3D network.

Model Size and Inference Speed

We quantitatively show the benefits of lower memory footprints and higher inference speed brought by the proposed multi-task network, compared to the "Single-task models" that perform all tasks separately by a chain of task-specific networks. In this regard, we employ the LidarBEV network introduced in 7.3 for object detection, and train UNet3D networks for other tasks. Starting from object detection, we gradually increase the number of perception tasks and calculate the required memory footprints and the inference speed averaged overall predictions on the evaluation data. Figure 7.3a and Figure 7.3b show the model size (in MegaByte) and the inference speed (in FPS), respectively. The LidarMTL network

Table 7.2: Foreground (FG).

Methods	$AP(\%)$	IOU (%)	Accu. (%)
RangeNet++ [99]	-	82.4	-
SqueezeSegv3 [158]	-	84.2	-
UNet3D	96.2	85.4	98.7
LidarMTL	97.0	85.6	98.7

Table 7.3: Ground heights (GH) (all LIDAR points).

Methods	RMSE (cm)				MAE (cm)			
	All	0-30m	30-50m	50-70m	All	0-30m	30-50m	50-70m
Plane	31.2	21.0	38.0	53.5	21.6	16.3	28.2	35.5
UNet3D	17.8	7.9	21.0	40.0	7.8	4.9	9.8	20.1
LidarMTL	18.6	8.8	22.2	40.4	8.8	5.7	11.0	21.4

Table 7.4: Ground areas (GC).

Methods	$AP(\%)$	IOU (%)	Accu. (%)
RangeNet++ [99]	-	95.2	-
SqueezeSegv3 [158]	-	95.9	-
UNet3D	99.6	94.5	98.2
LidarMTL	99.6	94.0	98.0

Table 7.5: Ground heights (GH) (only foreground points).

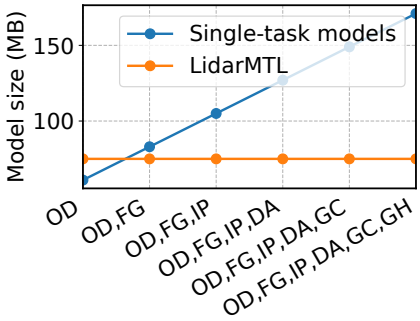
Methods	RMSE (cm)				MAE (cm)			
	All	0-30m	30-50m	50-70m	All	0-30m	30-50m	50-70m
Plane	20.8	17.8	27.3	35.4	15.3	12.9	22.4	28.7
UNet3D	8.6	6.5	11.7	19.1	5.5	4.4	8.0	13.9
LidarMTL	9.8	8.2	12.2	19.6	6.7	5.8	8.9	14.7

Table 7.6: Drivable areas (DA).

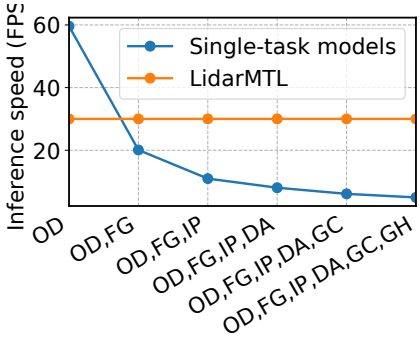
Methods	$AP(\%)$	IOU (%)	Accu. (%)
UNet3D	97.9	86.5	94.2
LidarMTL	97.4	84.5	93.4

Table 7.7: Intra-object part locations (IP).

Methods	RMSE				MAE			
	All	0-30m	30-50m	50-70m	All	0-30m	30-50m	50-70m
UNet3D	10.0	8.1	13.5	18.8	5.6	4.6	8.0	13.8
LidarMTL	9.9	8.2	13.3	18.1	5.7	4.7	8.0	13.2



(a) Point-cloud to Pixels



(b) Pixels to Point-cloud

Figure 7.3: A comparison of model size and inference speed

outperforms the single-task models when considering more than one perception task. While the LidarMTL network remains constant in model size and inference speed regardless of the number of tasks, the single-task model approaches require linearly-increasing memory and much lower inference speed. When performing all six perception tasks, the multi-task network is more than 2× smaller and 6× faster, showing its high efficiency, which is critical for online deployment.

Ablation Study

Number of Tasks

This section studies the performance of the single task which we focus on ("target task"), with an increasing number of multiple tasks ("auxiliary tasks") in the LidarMTL network. Figure 7.4a, Figure 7.4b, and Figure 7.4c select object detection, foreground classification, and ground height estimation as target task, respectively. We report the perception performance from the multi-task network relative to the singe-task network, with an increasing number of auxiliary tasks from left to right on the x-axis. No clear tendency is observed between the object detection performance and the number of tasks. The mAP scores fluctu-

ate between -1.5% and 1%. Introducing more auxiliary tasks increases AP for foreground classification, as well as regression errors for ground height estimation. However, the difference is small (less than 1% AP and 1.5cm errors). In conclusion, we could achieve on-par single-task perception performance, regardless of the combination of multiple tasks.

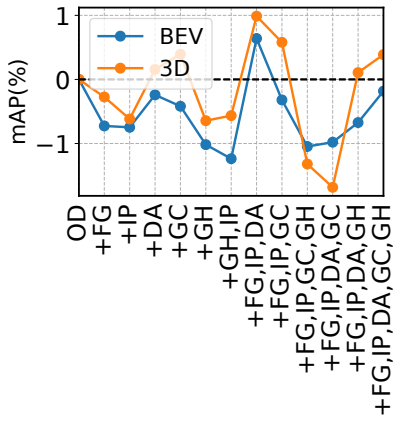
Impact of Loss Weights

It is known that a proper selection of the loss weight for every single task is crucial for multi-task learning [149]. In this ablation study, we train the LidarMTL network with different combinations of loss weights and compare their multi-task performances. "Fixed (equal weights)" assumes that each loss weight is equal. "Fixed (balanced)" balances the losses to similar scales. "Fixed (grid search)" finds a set of loss weights by grid search on the training dataset. Note that the loss weights from those three methods are fixed, and do not change during training (Equation 7.1). Instead, "Adaptive" employs the uncertainty weighting strategy shown by Equation 7.2 to balance single-task losses adaptively. "Adaptive + grid search" first puts a set of pre-defined loss weights from the grid search, and then balances the learning with uncertainty weighting.

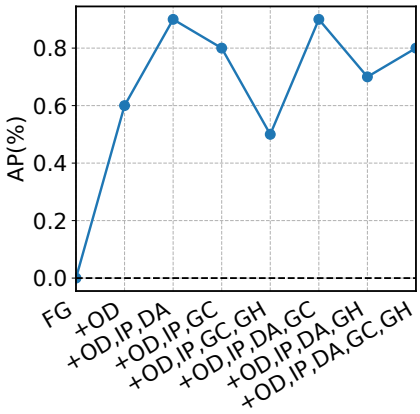
We report the perception performance for every single task as well as the averaged network's ranking in Table 7.8. Surprisingly, "Fixed (balanced)" shows inferior performance even slightly worse than "Fixed (equal weights) on the averaged ranking, indicating that simply balancing losses might not be the optimal choice in multi-task learning, as different single tasks may have different learning paces. "Adaptive" ranks last, with $4-6$cm larger ground height errors compared to the best results, showing the challenge to learn a proper set of loss weights from scratch. The networks trained with loss weights from grid search depict visible improvements (e.g. comparing "Fixed (grid search)" with "Fixed (equal weights)". When combining uncertainty weighting and grid search, the network slightly outperforms the "Fixed (grid search)" strategy and achieves the best multi-task performance. We conclude the necessity of using loss weights with grid search.

Robustness Testing

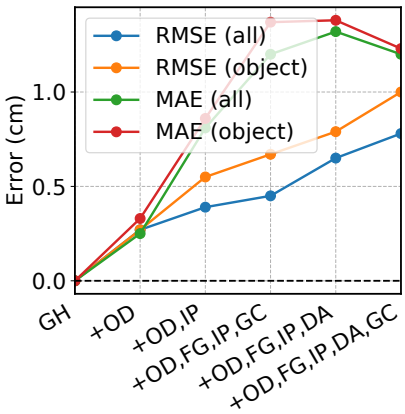
Finally, we study the robustness of point-wise prediction tasks (tasks FG, IP, DA, GC, GH) with increasingly sparse LIDAR points. Evaluating this robustness is crucial for autonomous driving, because the sparsity of point cloud varies significantly among LIDAR sensors and vehicle setup, and largely affects the perception performance [29]. In this regard, we use the LidarMTL network trained with full LIDAR points (8000 voxels) to make inferences on the evaluation data with downsampled LIDAR points by factors $2, 4, 8, 16, 32$. Results are shown in Figure 7.5. The performance of DA, GC, and IP drops slightly with downsample factors smaller than 8. FG remains high AP scores above 90% even with 32 as the downsample factor (i.e. 250 non-empty voxels). The performance of GH drops quickly with a downsample factor of 4. The experiment shows that different tasks have different robustness against LIDAR point cloud sparsity.



(a) Point-cloud to Pixels



(b) Pixels to Point-cloud



(c) Pixels to Point-cloud

Figure 7.4: The performance of the target task when trained with increasing number of auxiliary tasks

Table 7.8: A comparison among the LidarMTL networks trained with different loss weights.

Loss weights	OD		FG	DA	GC	GH		IP		Avg.
	mAP$_{BEV}$ (%)	mAP$_{3D}$ (%)	*AP* (%)	*AP* (%)	*AP* (%)	RMSE (cm)	MAE (cm)	RMSE	MAE	Rank
Fixed (equal weights)	49.6	34.5	96.7	97.7	99.6	20.5	10.7	10.7	6.4	2.7
Fixed (balanced)	48.4	32.9	97.0	97.8	99.6	19.2	9.2	12.7	8.1	3.0
Fixed (grid search)	49.2	34.7	97.2	97.5	99.6	18.6	8.7	10.0	5.7	1.8
Adaptive [21]	49.2	34.7	97.0	97.2	99.6	24.0	14.1	10.8	6.5	3.3
Adaptive [21] + grid search	49.8	35.0	97.0	97.4	99.6	18.6	8.8	9.9	5.6	1.5

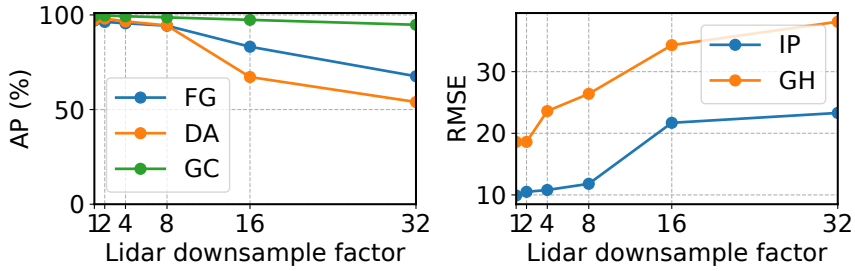


Figure 7.5: The performance of point-wise predictions with increasingly sparse LIDAR points

Application to Online Localization

Localization in urban environments requires point cloud maps and point registration algorithms. However, LIDAR-based localization typically suffered from dynamic objects and undulating road surfaces[157]. By semantically segmenting the scene, LidarMTL provides an ideal pre-processing for such localization modules. To study LidarMTL's impact on localization, we used the outputs from DA and FG to help localize the vehicle. As a comparison, we have 4 types of inputs for the localization algorithm: the raw scan, the point cloud without DA, the point cloud without FG, and the point cloud without both DA and FG.

We perform localization experiments on 24 trajectories spanning 2.69KM, and the vanilla NDT registration algorithm in Autoware [62] is chosen as the real-time localization module. In our experiment, the NDT voxel resolution is 1 meter. The map is created with the ground-truth scan without DA and FG downsampled to 0.2 meters.

Table 7.9 shows the performance of the localization algorithm with various point cloud inputs. We also include the result from ground-truth DA and FG tasks for comparison. The performance is evaluated with Root Mean Squared Error along three axes and the yaw angle. Furthermore, we list the success rate for these tests: one is considered a failure if the translation RMSE is larger than 3m or if the rotation RMSE is larger than 4°. Compared with the raw input, the LidarMTL-processed inputs yield more accurate localization. Furthermore, since dynamic objects are removed from the scan, the algorithm performed robustly in complicated environments. As compared with the ground truth point cloud inputs, the LidarMTL pre-processing reaches a similar level of localization accuracy and success rate. Given the stochastic nature of the NDT algorithm, the LidarMTL even out-performs ground truth segmentation in certain evaluation matrices.

7.4 Chapter Summary

The discussion in Chapter 7 provides an insight into the more efficient utilization of the limited data resources. We present the multi-task network to jointly perform six perception tasks for 3D object detection and road understanding. Comprehensive experiments demonstrate that the proposed network can complete all six tasks with a similar level of

Table 7.9: Online localization results.

Point Cloud Type		Translation RMSE (m)			Rotation RMSE (°)	Success Rate (%)
		X	Y	Z	Yaw	
Raw		1.80	1.27	1.13	1.16	83.3
GT	no DA	1.13	0.67	0.34	1.00	95.8
	no FG	1.46	0.56	0.46	0.71	95.8
	no DA, FG	1.51	0.87	0.34	1.25	95.8
LidarMTL	no DA	1.83	0.59	0.21	0.84	95.8
	no FG	1.62	0.58	0.43	0.82	95.8
	no DA, FG	1.73	0.65	0.06	1.13	100

performance from single-model designs. The LidarMTL network is small, fast, accurate, and useful for localization, making it highly desirable for online deployment in autonomous cars.

With the conclusion of Chapter 7, we finish the discussion on cartographic algorithms. In summary, Part II starts with a road module-focused algorithm for the robust exploration of the scene. Then, shifting the concentration to intersections, we discuss sensor fusion strategies in the BEV domain. Lastly, we propose a multi-task learning scheme to fully utilize the limited data resource.

The next step in the life cycle of the HD map development lies in the hands of the customer: how to efficiently query and use the map will be the main topic of Part III.

Part III

HD Map Management

Chapter 8

Submap Query and Stitching

The final step in the map development process, for either paper maps or HD maps, is the distribution. In traditional paper map businesses, the final step for the publisher is to bind individual maps together as an atlas, and then the distributor sends out the collection to individual customers. Typically, contents within an atlas are divided into smaller maps on each printed page, and these pages are indexed on the borders for readers' queries. Such map collection and indexing system has been in production for hundreds of years [119].

In the modern HD map distribution process, efficient map management and query mechanism are also needed. Even though HD maps are stored in digital formats on hardware that is significantly smaller than paper map books, it is still not ideal to load a complete HD map onto the onboard system. For each autonomous agent on the road, both the storage and the computation capacities are extremely limited. Thus, a divide-and-conquer method is preferred in the HD map distribution process.

A common practice is to divide a large city map into smaller tiles for a faster query. In this chapter, an efficient submap query and alignment mechanism ATLAS is discussed. To begin with, the urban map is firstly divided into square-shaped tiles to form a spatial occupancy tree. Here, a Red-Black tree representation is introduced and the computational complexity is analyzed to demonstrate the fast query capacity. To utilize these submaps in real-time, a stitching algorithm based on point cloud registration is proposed. To demonstrate the effectiveness of the map management system, the localization function in the autonomous driving stack is tested with real-world data. The result demonstrates an efficient query operation of the ATLAS framework[1].

8.1 Introduction

As demonstrated in previous chapters, the HD map plays a significant role in urban autonomous driving. These maps are loaded with detailed urban driving information such as lane semantics, traffic rules, and road conditions. With the constructed maps stored in the

[1]This chapter includes materials from the author's previous work [172]

server, however, autonomous agents still have to gain access to these maps at the time of deployment.

Map researchers are facing two questions. First, how to store the HD map, and second, how to use the map efficiently in real-time. A naive approach is to load the entire map onto the onboard storage system, and the agent could directly query the map information from the hardware. However, a direct loading operation is usually infeasible as the onboard storage devices are often of limited capacity. Furthermore, querying an entire city map might exceed the computing power of the onboard computer.

When facing a hard-to-solve problem, computer scientists turn to the divide-and-conquer strategy. In the mapping application, the map is firstly divided into smaller parts, namely submaps [164]. Only a few submaps will be loaded onto the onboard device at a time, and they will be replaced by new submaps once the related information is utilized.

The material covered in this chapter leverages the concept of submap in HD map management. Under the proposed framework ATLAS, the urban landscape is divided into smaller rectangular tiles as submaps. These tiles are organized into a Red-Black tree data structure for fast queries. For inter-map transitions in real-time, we further design a submap stitching algorithm to match two neighboring submaps. With pre-built geometric HD map layers, the proposed framework is constructed and tested on localization tasks in urbanized areas. The ATLAS management-query-stitching framework demonstrates efficient operations in real-time.

8.2 Tile-based Query

Submaps are pre-built map patches representing only a fraction of the complete urban environment $\{S\}$. In the scope of this chapter, each submap will be a 3D point cloud of the corresponding area. Here, we assume that all the submaps are under the Universal Transverse Mercator (UTM) coordinate system.

Each submap is an $r \times r$ grid G_i on the xy-plane. We denote

$$G_i(u, v) = \{(x, y) \in \mathbb{R}^2 \,:\, x \in [ur, (u+1)r), y \in [vr, (v+1)r)\}. \tag{8.1}$$

Here, r, x, and y are measured by meters, consistent with the UTM coordinates. For each tile $G_i(u, v)$, $G_i(u, v) \in S$ if and only if $G_i(u, v)$ has been visited by mapping vehicles and contains geometrical information. Considering the shape of each submap is a square, we will use the name square or tile interchangeably with submap.

An intuitive data structure to represent a collection of submaps is a list. Assuming the maximal map distance in the interior of the map is L, both the storage and the query complexity will be $O(L^2 r^{-2})$.

Alternatively, we propose to use a Red-Black tree (RB tree) representation, which decreases the query time complexity to $O(\log(L^2 r^{-2}))$. Since most routing tasks are linear between two points in a map, the query operations along the trajectory have a complexity of $O(\log(L^2 r^{-2}))$. Here, the space complexity will still be $O(L^2 r^{-2})$, because every map needs

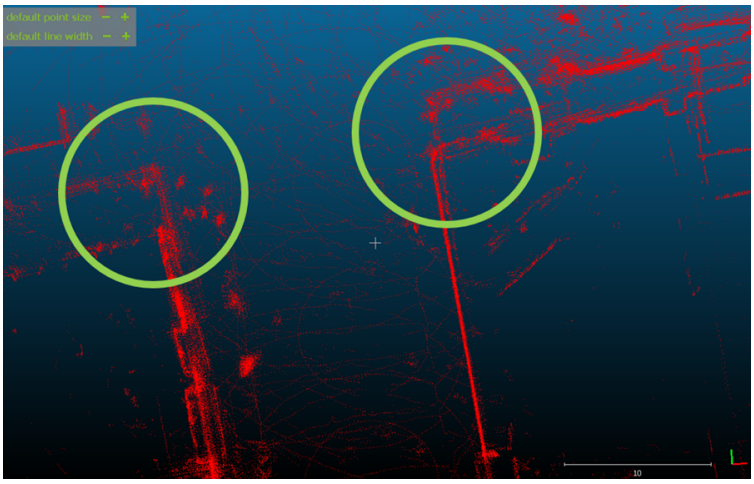


Figure 8.1: Inconsistent fused maps with simple overlapping

to be stored. However, since the storage device is located off-board, the onboard systems bear no burden.

8.3 Map Stitching

When an autonomous agent is planning to go from point A to point B within a master map $\{S\}$, a series of submaps will be queried sequentially following the aforementioned data structure. However, the list of queried submaps will not be loaded directly onto the onboard system due to the storage space limitation. Thus we propose to use a first-in-first-out (FIFO) queue for the onboard map management processing, using at most three submaps at a time.

Within the FIFO queue, one submap will be the ego map where the vehicle is traveling in, and the other two will be forecasting maps on the horizon of the vehicle's path planning. These submaps have some overlapping areas on the border as buffer zones.

However, due to the noises and errors in each submap creation process, naively overlapping two submaps according to their UTM tags would result in inconsistent buffer zones as shown in Figure 8.1. Thus, we further calibrate each map on the buffer zones with point registration algorithms. With the UTM tags as initial guesses, we apply the Normal Distribution Transformation [89] point matching between the two LIDAR point clouds to optimize the relative translation and rotation. Once the vehicle passes through the buffer zones, we will update the global coordinate reference with the ego map's UTM tag. Thus, the relative localization solution of the vehicle is always stable.

Figure 8.2: Experiment map coverage in Downtown Berkeley

8.4 Experiments

Dataset and Preparation

The data used for our experiment is collected in downtown Berkeley with a similar sensor setup in the UrbanLoco dataset [157]. Figure 8.2 is a demonstration of the submaps projected onto the Google Map [41]. Here, each submap is constructed with the SLAM algorithm provided by Koide et al. [67], and a global UTM tag is added to each map for initial alignment.

Considering the geometrical nature of the proposed framework, the geographical localization task is chosen for evaluation purposes. The ground truth localization is from an RTK-IMU combined localization solution as introduced in [157], and we use a single-point GPS as the baseline. We evaluate the absolute transnational error for both methods. For ATLAS, the error is calculated in each corresponding submap frame, and for the GPS, the error is calculated in the global frame.

Experiment Result

To begin with, the experiment is performed in real-time on a laptop with Intel 8750H CPU and 16GB memory, and the proposed framework could query and stitch up the submaps without impacting other sensing processes.

A visualization of the completely stitched map could be seen in Figure 8.3, but during

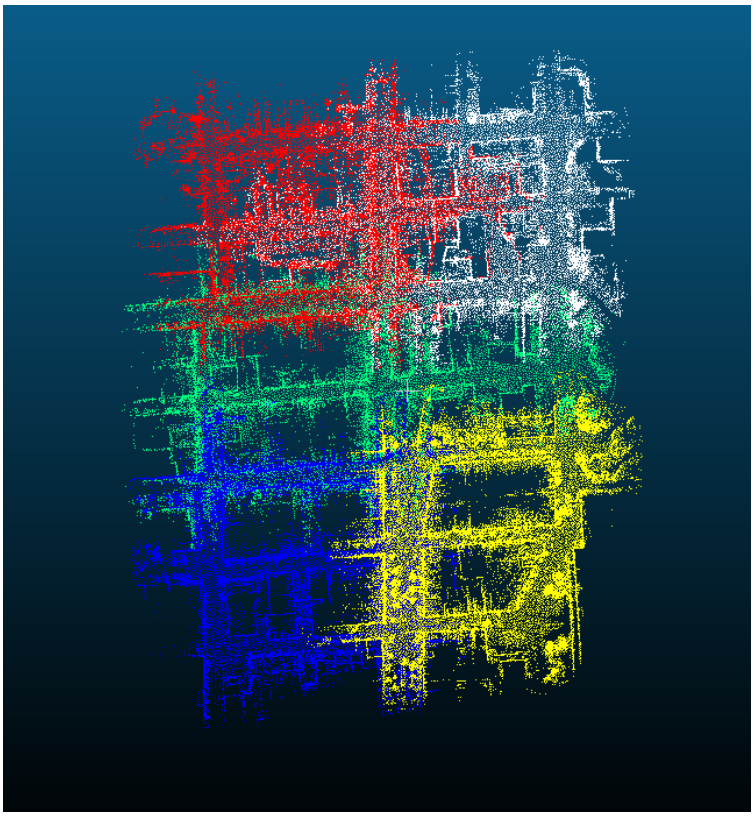
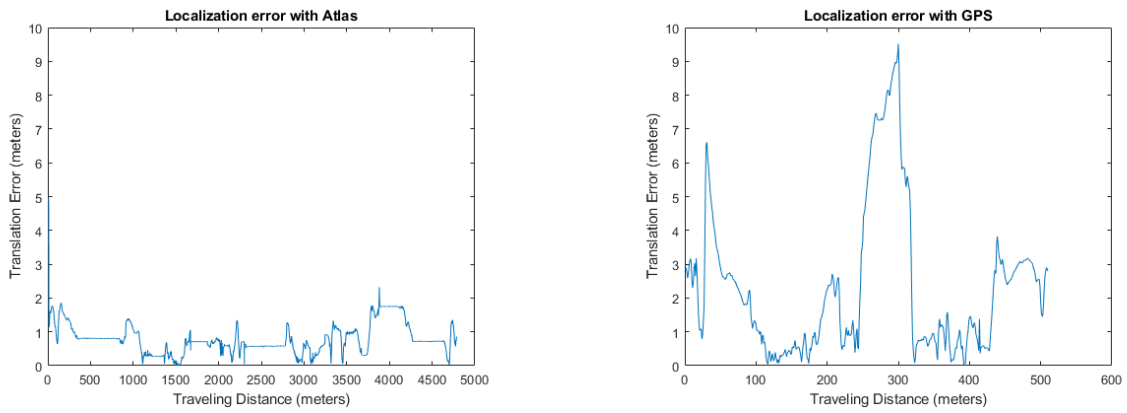


Figure 8.3: Experiment stitching result in Downtown Berkeley

	Mean Absolute Translation Error(m)	Translation RMSE
ATLAS	0.79	0.92
GPS baseline	2.33	3.19

Table 8.1: Translation error with ATLAS and GPS baseline

implementation, we only use three maps in the FIFO queue. The translation error of the localization experiment with the ATLAS map system is shown in Figure 8.4a. Compared with the GPS baseline in Figure 8.4b, the mean absolute error of the ATLAS-backed localization is significantly lower than that of the baseline. The periodic increases in absolute errors with the ATLAS systems correspond to the map transition phase. It is worth noticing that the maximum deviation of 2.15m of the ATLAS framework is still lower than the average deviation of the baseline. Quantitatively (Table 8.1, both absolute errors and the RMS are significantly smaller than those of the baseline GPS localization. The low RMS of ATLAS shows a smooth trajectory prediction. Thus, the ATLAS-backed localization is efficient and reliable in urban settings.



(a) ATLAS (b) GPS

Figure 8.4: Algorithm performance in transnational error along the trajectory

8.5 Chapter Summary

As the last step in the complete life cycle of the HD map development process, this chapter discusses a submap management framework ATLAS in the urban driving scenario. The ATLAS framework stores submaps within an RB tree data structure for fast query and efficient storage. A following map stitching mechanism is further designed to smooth the transition between submaps during online deployment. The proposed framework is tested with real-world localization tasks, and the result shows an efficient and robust localization in urban settings.

Chapter 9

Conclusion and Future Work

9.1 Summary

Recognizing the HD map as critical infrastructure for urban autonomous driving, this dissertation reviews the complete life cycle of the HD map development process. To make the HD map development suitable for large-scale urban applications, the materials included in this work contribute to a salable HD map solution from multiple perspectives. We start the discussion from the initial sensor setups and data collections for mapping purposes and then extend to the automatic HD map construction algorithms. Lastly, we review the map management system for real-time deployment of the HD map.

Mapping Platform and Data

Sensors and data are fundamental elements for any mapping tasks. To begin with, we first discuss the mapping platform and the dataset for mapping applications in Part I. Chapter 2 focuses on the joint calibration and synchronization of the sensor suite on a mobile mapping platform. In this section, the complementary LIDAR-camera configuration is discussed, and a semantic-based optimization algorithm is proposed to estimate both the geometric and the temporal relationship between these two sensor modalities. In Chapter 3, an exemplar mapping platform and an urban dataset are introduced. The design of the mapping vehicle considers complicated urban scenarios, and the dataset includes some of the most challenging city driving scenes. The dataset is open to the public to encourage research in the mapping field. With the mapping platform configured, the next question in the life cycle of an HD map is the routing problem. With more than one mapping vehicle, how to efficiently route a mapping fleet is discussed in Chapter 4. Here, a Model Predictive Control-based algorithm is proposed to accommodate traffic conditions and map updating problems.

HD Map Construction

With the sensor calibrated and the data collected, Part II focuses on the algorithms related to the automatic generation of the HD map. Chapter 5 introduces a particle filter-based algorithm to efficiently explore the lanes in complicated urban situations. The algorithm specifically solves the merging, forking, and irregular lane cases on city roads, and the proposed method was tested in densely urbanized areas to demonstrate its robustness in complicated scenes. Chapter 6 moves more towards the intersections and potential solutions with a multi-sensor setup. Here, we treat the mapping problem as semantic segmentation in the Bird's Eye View frame. Network design comparisons are also provided to demonstrate a preferred strategy in cross-domain fusion tasks. Chapter 7 studies the potential of multi-task learning for both static and dynamic objects on the road to exploit the information in limited data. Built upon a single backbone, the proposed method compresses six tasks into one neural network, and the evaluation shows that the performance was comparable with single-task models.

HD Map Management

In Part III, the management and deployment of the HD map are discussed. Chapter 8 introduces a tile-based map management system to query and combine smaller HD maps for real-time application. The proposed framework leverages an RB tree data structure and uses a submap queue during vehicle operation to store only useful maps.

9.2 Future Work

With the HD map applications infiltrating into more perspectives of urban autonomous driving, more mapping-related topics come into the view of researchers recently. Within the scope of this dissertation, some of these topics are not entirely covered, but more research is needed in the following domains.

Data Efficiency

Methods mentioned in Part II leverage public datasets to parse the driving scene. Looking back at the datasets, one might find that the scene labels are still created by human beings. In the past few years, self-supervised learning has been advancing quickly to help humans label scenes. More recently, the raise of MLOps research pushes data efficiency to another level. Map researchers also need to consider the data source and efficiency in the coming years for more economic map construction.

Map Update Problem

"A map expires the moment it is created." Such a statement, though cruel to map researchers, is true. More research is needed to see how maps could be updated accurately at a reasonable cost. First, recognizing and localizing similar objects in the scene under different lighting/traffic conditions are yet to be solved. Secondly, how to modify the old map without replacing it completely is another potential research direction in the future.

Bibliography

[1] Min Bai, Gellert Mattyus, Namdar Homayounfar, Shenlong Wang, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. "Deep multi-sensor lane detection". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3102–3109.

[2] Xiwei Bai, Weisong Wen, Guohao Zhang, and Li-Ta Hsu. "Real-time GNSS NLOS Detection and Correction Aided by Sky-Pointing Camera and 3D LiDAR". In: (2019). ION GNSS+ 2019 Pacific PNT Meeting.

[3] Sven Bauer, Yasamin Alkhorshid, and Gerd Wanielik. "Using High-Definition maps for precise urban vehicle localization". In: *Proceedings of 2016 IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil, Nov. 2016, pp. 492–497.

[4] Ulrich Baumann, Yuan-Yao Huang, Claudius Gläser, Michael Herman, Holger Banzhaf, and J Marius Zöllner. "Classifying road intersections using transfer-learning on a deep neural network". In: *Proceedings of 2018 IEEE International Conference Intelligent Transportation System (ITSC)*. IEEE. 2018, pp. 683–690.

[5] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[6] Pietro Belotti, Pierre Bonami, John J. Forrest, Lazlo Ladanyi, Carl Laird, Jon Lee, Francois Margot, and Andreas Waechter. *Boomin*.
Available at `https://www.coin-or.org/Bonmin/`.

[7] Paul J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (Feb. 1992), pp. 239–256.

[8] Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo-optimization modeling in python*. Third. Vol. 67. Springer Science & Business Media, 2021.

[9] 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*. `https://www.nuscenes.org/`. 2019.

[10] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. "Fast LIDAR-based road detection using fully convolutional neural networks". In: *Proceedings of 2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1019–1024.

[11] Yigit Baran Can, Alexander Liniger, Ozan Unal, Danda Pani Paudel, and Luc Van Gool. "Understanding Bird's-Eye View Semantic HD-Maps Using an Onboard Monocular Camera". In: *Computing Research Repository (CoRR)* abs/2012.03040 (2020).

[12] Carmera. *CARMERA Launches Inventory Map, Provides Live Look at Road Changes for Autonomous Driving and More*. `https://medium.com/field-of-view/carmera-launches-inventory-map-provides-live-look-at-road-changes-for-autonomous-driving-and-more-f4c284d4c77b`. Accessed: 2022-03-10. 2021.

[13] Turner-Fairbank Highway Research Center. *Intersection Safety*. `https://highways.dot.gov/research/research-programs/safety/intersection-safety`. Accessed: 2022-03-10. 2021.

[14] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. "Argoverse: 3D Tracking and Forecasting with Rich Maps". In: *Proceedings of 2019Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[15] Hongyu Chen, Zhijie Yang, Xiting Zhao, Guangyuan Weng, Haochuan Wan, Jianwen Luo, Xiaoya Ye, Zehao Zhao, Zhenpeng He, Yongxia Shen, and Sören Schwertfeger. "Advanced Mapping Robot and High-Resolution Dataset". In: *ArXiv* abs/2007.12497 (2020).

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *Proceedings of 2018 European Conference on Computer Vision (ECCV)*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Cham: Springer International Publishing, 2018, pp. 833–851. ISBN: 978-3-030-01234-2.

[17] Liangfu Chen, Zeng Yang, Jianjun Ma, and Zheng Luo. "Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation". In: *Proceedings of 2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1283–1291.

[18] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "3d object proposals for accurate object class detection". In: *Proceedings of 2015 Advances in Neural Information Processing Systems (NIPS)*. Citeseer. 2015, pp. 424–432.

[19] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3D Object Detection Network for Autonomous Driving". In: *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6526–6534.

[20] Yihua Chen and John Krumm. "Probabilistic modeling of traffic lanes from GPS traces". In: *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. New York, New York, USA: ACM Press, 2010, pp. 81–88. ISBN: 9781450304283. DOI: 10.1145/1869790.1869805. URL: http://portal.acm.org/citation.cfm?doid=1869790.1869805.

[21] Roberto Cipolla, Yarin Gal, and Alex Kendall. "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7482–7491.

[22] William Cook. *Concorde VLP Solver*. URL: https://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm.

[23] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of 2016 the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[24] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of 2017 The Annual Conference on Robot Learning (CoRL)*. Mountain View, U.S.A., Nov. 2017, pp. 1–16.

[25] Mahdi Elhousni, Yecheng Lyu, Ziming Zhang, and Xinming Huang. "Automatic Building and Labeling of HD Maps with Deep Learning". In: *Proceedings of 2020 AAAI Conference on Artificial Intelligence*. 2020.

[26] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[27] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The PASCAL Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338.

[28] Rui Fan, Hengli Wang, Peide Cai, and Ming Liu. "Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection". In: *Proceedings of 2020 European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 340–356.

[29] Di Feng, Zining Wang, Yiyang Zhou, Lars Rosenbaum, Fabian Timm, Klaus Dietmayer, Masayoshi Tomizuka, and Wei Zhan. "Labels Are Not Perfect: Inferring Spatial Uncertainty in Object Detection". In: *arXiv preprint arXiv:2012.12195* (2020).

[30] Di Feng, Yiyang Zhou, Chenfeng Xu, Masayoshi Tomizuka, and Wei Zhan. "A Simple and Efficient Multi-task Network for 3D Object Detection and Road Understanding". In: *Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 7067–7074. DOI: 10.1109/IROS51168.2021.9635858.

[31] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms". In: *Proceedings of 2013 IEEE International Conference Intelligent Transportation System (ITSC)*. 2013.

[32] Paul Furgale, Joern Rehder, and Roland Siegwart. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems". In: *Proceedings of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[33] 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 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[34] Michael Garey and David Johnson. *Computers and intractability: A guide to the theory of npcompleteness*. Computers and Intractability, 1979.

[35] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. "3D-LaneNet: End-to-End 3D Multiple Lane Detection". In: *Proceedings of 2019 International Conference on Computer Vision (CVPR)*. 2019.

[36] Andreas Geiger, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun. "3D traffic scene understanding from movable platforms". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.5 (2014), pp. 1012–1025. ISSN: 01628828. DOI: 10.1109/TPAMI.2013.185.

[37] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets robotics: The KITTI dataset". In: *International Journal of Robotics Research* (2013), pp. 1229–1235.

[38] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2012, pp. 3354–3361.

[39] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. "Automatic camera and range sensor calibration using a single shot". In: *Proceedings of 2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3936–3943. DOI: 10.1109/ICRA.2012.6224570.

[40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[41] Google Earth. *Satellite Images of the San Francisco Bay Area*. Accessed: 2022-03-10. 2021.

[42] Benjamin Graham and Laurens van der Maaten. "Submanifold sparse convolutional networks". In: *arXiv preprint arXiv:1706.01307* (2017).

[43] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. "A Tutorial on Graph-Based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43. ISSN: 1941-1197. DOI: 10.1109/MITS.2010.939925.

[44] Paul D. Groves and Mounir Adjrad. "Performance assessment of 3D-mapping–aided GNSS part 1: Algorithms, user equipment, and review". In: *Navigation* 66.2 (2019), pp. 341–362. DOI: 10.1002/navi.288. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/navi.288. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.288.

[45] Chunzhao Guo, Kiyosumi Kidono, Junichi Meguro, Yoshiko Kojima, Masaru Ogawa, and Takashi Naito. "A low-cost solution for automatic lane-level map generation using conventional in-car sensors". In: *IEEE Transactions on Intelligent Transportation Systems* 17.8 (Aug. 2016), pp. 2355–2366. ISSN: 15249050. DOI: 10.1109/TITS.2016.2521819.

[46] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jinghao Miao, Jingao Wang, and Tae Eun Choe. "Gen-LaneNet: A Generalized and Scalable Approach for 3D Lane Detection". In: *Proceedings of 2020 European Conference on Computer Vision (ECCV)*. 2020.

[47] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. "Structure Aware Single-stage 3D Object Detection from Point Cloud". In: *Proceedings of 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11873–11882.

[48] Keld Helsgaun. *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems*. 2017.

[49] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. "Using kinect-style depth cameras for dense 3D modeling of indoor environments". In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663.

[50] Namdar Homayounfar, Justin Liang, Wei-Chiu Ma, Jack Fan, Xinyu Wu, and Raquel Urtasun. "DAGMapper: Learning to Map by Discovering Lane Topology". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 2911–2920.

[51] 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*. 2020. DOI: 10.48550/ARXIV.2006.14480. URL: https://arxiv.org/abs/2006.14480.

[52] Li-Ta Hsu, Feiyu Chen, and Shunsuke Kamijo. "Evaluation of multi-GNSSs and GPS with 3D map methods for pedestrian positioning in an urban canyon environment". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 98.1 (2015), pp. 284–293.

[53] Li-Ta Hsu, Yanlei Gu, and Shunsuke Kamijo. "3D building model-based pedestrian positioning method using GPS/GLONASS/QZSS and its reliability calculation". In: *GPS Solutions* 3 (2016), pp. 413–428.

[54] Albert Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. "Visual odometry and mapping for autonomous flight using an RGB-D camera". In: *Proceedings of 2011 International Symposium on Robotics Research (ISRR)*. 2011.

[55] Ganesh Iyer, R. Karnik Ram, J. Krishna Murthy, and K. Madhava Krishna. "CalibNet: Geometrically Supervised Extrinsic Calibration using 3D Spatial Transformer Networks". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1110–1117. DOI: 10.1109/IROS.2018.8593693.

[56] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. "Complex urban dataset with multi-level sensors from highly diverse urban environments". In: *The International Journal of Robotics Research* 38.6 (2019), pp. 642–657. DOI: 10.1177/0278364919843996.

[57] Avdhut Joshi and Michael R. James. "Generation of Accurate Lane-Level Maps from Coarse Prior Maps and Lidar". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (2015), 19‘–29.

[58] Avdhut Joshi and Michael R. James. "Joint probabilistic modeling and inference of intersection structure". In: *Proceedings of 2014 IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Institute of Electrical and Electronics Engineers Inc., Nov. 2014, pp. 1072–1078. ISBN: 9781479960781. DOI: 10.1109/ITSC.2014.6957830.

[59] Jaehyeon Kang and Nakju Doh. "Automatic targetless camera–LIDAR calibration by aligning edge with Gaussian mixture model". In: *Journal of Field Robotics* 37 (Aug. 2019). DOI: 10.1002/rob.21893.

[60] Andrej Karpathy. *Tesla Vision Presentation on Tesla AI Day*. 2021. URL: https://youtu.be/j0z4FweCy4M?t=2928.

[61] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. "An Open Approach to Autonomous Vehicles". In: *IEEE Micro* 35.6 (2015), pp. 60–69.

[62] 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: *Proceedings of 2018 ACM/IEEE International Conference on Cyber-Physical Systems*. ICCPS '18. Porto, Portugal: IEEE Press, 2018, pp. 287–296. ISBN: 9781538653012. DOI: 10.1109/ICCPS.2018.00035. URL: https://doi.org/10.1109/ICCPS.2018.00035.

[63] Shodai Kato, Mitsunori Kitamura, Taro Suzuki, and Yoshiharu Amano. "Nlos satellite detection using a fish-eye camera for improving gnss positioning accuracy in urban area". In: *Journal of robotics and mechatronics* 28.1 (2016), pp. 31–39.

[64] Christian Kerl, Jürgen Sturm, and Daniel Cremers. "Robust odometry estimation for RGB-D cameras". In: *Proceedings of 2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013.

[65] Eung-su Kim and Soon-Yong Park. "Extrinsic Calibration between Camera and LiDAR Sensors by Matching Multiple 3D Planes". In: *Sensors* 20.1 (). ISSN: 1424-8220. DOI: 10.3390/s20010052.

[66] Akio Kodaira, Yiyang Zhou, Pengwei Zang, Wei Zhan, and Masayoshi Tomizuka. "SST-Calib: Simultaneous Spatial-Temporal Parameter Calibration between LIDAR and Camera". In: *2022 IEEE Conference on Intelligent Transportation Systems (ITSC), under review*. 2022.

[67] Kenji Koide, Jun Miura, and Emanuele Menegatti. "A Portable 3D LIDAR-based System for Long-term and Wide-area People Behavior Measurement". In: (2019). Advanced Robotic Systems.

[68] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. "Joint 3D Proposal Generation and Object Detection from View Aggregation". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–8.

[69] Julius Kümmerle, Tilman Kühner, and Martin Lauer. "Automatic Calibration of Multiple Cameras and Depth Sensors with a Spherical Target". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–8. DOI: 10.1109/IROS.2018.8593955.

[70] Lars Kunze, Tom Bruls, Tarlan Suleymanov, and Paul Newman. "Reading between the Lanes: Road Layout Reconstruction from Partially Segmented Scenes". In: *Proceedings of 2018 IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. Vol. 2018-November. Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 401–408. ISBN: 9781728103235. DOI: 10.1109/ITSC.2018.8569270.

[71] Kiho Kwak, Daniel F. Huber, Hernan Badino, and Takeo Kanade. "Extrinsic calibration of a single line scanning lidar and a camera". In: *Proceedings of 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 3283–3289. DOI: 10.1109/IROS.2011.6094490.

[72] Leonid Laboshin. *loam_velodyne*. https://github.com/laboshinl/loam_velodyne. 2016.

[73] 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 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[74] Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric Xing, and Ruslan Salakhutdinov. "Gated Path Planning Networks". In: *Proceedings of 2018 International Conference on Machine Learning (ICML)*. 2018.

[75] Kun Lei, Peng Guo, Yi Wang, Xiao Wu, and Wenchao Zhao. *Solve routing problems with a residual edge-graph attention neural network*. 2021. arXiv: 2105.02730 [cs.LG].

[76] Jun Li, Xue Mei, Danil Prokhorov, and Dacheng Tao. "Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.3 (Mar. 2017), pp. 690–703. ISSN: 21622388. DOI: 10.1109/TNNLS.2016.2522428.

[77] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. "HDMapNet: An Online HD Map Construction and Evaluation Framework". In: *Proceedings of 2021 Mapping workshop of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Virtual, June 2021.

[78] Xuyou Li, Shitong Du, Guangchun Li, and Haoyu Li. "Integrate point-cloud segmentation with 3D lidar scan-matching for mobile robot localization and mapping". In: *Sensors* 20.1 (2020), p. 237.

[79] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Shenlong Wang, and Raquel Urtasun. "Convolutional recurrent network for road boundary extraction". In: *Proceedings of 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9512–9521.

[80] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. "Multi-Task Multi-Sensor Fusion for 3D Object Detection". In: *Proceedings of 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7345–7353.

[81] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal Loss for Dense Object Detection". In: *Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[82] Rong Liu, Jinling Wang, and Bingqi Zhang. "High Definition Map for Automated Driving: Overview and Analysis". In: *Journal of Navigation* 73.2 (2020), pp. 324–341. DOI: 10.1017/S0373463319000638.

[83] Zhijian Liu, Haotian Tang, Sibo Zhu, and Song Han. "SemAlign: Annotation-Free Camera-LiDAR Calibration with Semantic Alignment Loss". In: *Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[84] Bruce D Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *Proceedings of 1981 International Conference on Arrtificial Intelligence (IJCAI)*. 1981, pp. 674–679.

[85] Xudong Lv, Boya Wang, Ziwen Dou, Dong Ye, and Shuo Wang. "LCCNet: LiDAR and Camera Self-Calibration using Cost Volume Network". In: *Proceedings of 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*. 2021, pp. 2888–2895. DOI: 10.1109/CVPRW53098.2021.00324.

[86] Wei-Chiu Ma, Ignacio Tartavull, Ioan Andrei Bârsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. *Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization*. 2019. arXiv: 1908.03274 [cs.CV].

[87] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. ISBN: 0387008934.

[88] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. "1 Year, 1000km: The Oxford RobotCar Dataset". In: *The International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498. eprint: http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+html. URL: http://dx.doi.org/10.1177/0278364916679498.

[89] Martin Magnusson, Achim Lilienthal, and Tom Duckett. "Scan registration for autonomous mining vehicles using 3D-NDT". In: *Journal of Field Robotics* 24.10 (Oct. 2007), pp. 803–827.

[90] Mark Maimone, Yang Cheng, and Larry Matthies. "Two years of visual odometry on the mars exploration rovers". In: *Journal of Field Robotics* 24.2 (2007), pp. 169–186.

[91] Andrew Makhorin. *GLPK (GNU Linear Programming Kit)*.
Available at http://www.gnu.org/software/glpk/glpk.html.

[92] Kaustubh Mani, Swapnil Daga, Shubhika Garg, Sai Shankar Narasimhan, Madhava Krishna, and Krishna Murthy Jatavallabhula. "MonoLayout: Amodal scene layout from a single image". In: *Proceedings of 2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Snowmass, U.S.A, Mar. 2020, pp. 1689–1697.

[93] John Marshall. "Creating and viewing skyplots". In: *GPS Solutions* 6.1 (2002), pp. 118–120.

[94] Gellert Mattyus, Shenlong Wang, Sanja Fidler, and Raquel Urtasun. "HD Maps: Fine-Grained Road Segmentation by Parsing Ground and Aerial Images". In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[95] Annika Meyer, N. Ole Salscheider, Piotr F. Orzechowski, and Christoph Stiller. "Deep Semantic Lane Segmentation for Mapless Driving". In: *Proceedings of 2018 IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 869–875. ISBN: 9781538680940. DOI: 10.1109/IROS.2018.8594450.

[96] Annika Meyer, Jonas Walter, and Martin Lauer. "Fast Lane-Level Intersection Estimation using Markov Chain Monte Carlo Sampling and B-Spline Refinement". In: *Proceedings of 2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020.

[97] Annika Meyer, Jonas Walter, Martin Lauer, and Christoph Stiller. "Anytime Lane-Level Intersection Estimation Based on Trajectories of Other Traffic Participants". In: *Proceedings of 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 3122–3129.

[98] Microsoft. *Microsoft Bing Map*. 2020.

[99] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. "Rangenet++: Fast and accurate lidar semantic segmentation". In: *Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4213–4220.

[100] Peyman Moghadam, Michael Bosse, and Robert Zlot. "Line-based extrinsic calibration of range and image sensors". In: *Proceedings of 2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 3685–3691. DOI: 10.1109/ICRA.2013.6631095.

[101] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. "Junior: The Stanford Entry in the Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 91–123. ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_3. URL: https://doi.org/10.1007/978-3-642-03991-1_3.

[102] Mong H. Ng, Kaahan Radia, Jianfei Chen, Dequan Wang, Ionel Gog, and Joseph E. Gonzalez. "BEV-Seg: Bird's Eye View Semantic Segmentation Using Geometry and Semantic Point Cloud". In: *Proceedings of Scalability in Autonomous Driving Workshop of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Virtual, June 2020, pp. 11138–11147.

[103] Marcos Nieto, Jon Arróspide, and Luis Salgado. "Road environment modeling using robust perspective analysis and recursive Bayesian segmentation". In: *Mach. Vis. Appl.* 22 (Nov. 2011), pp. 927–945. DOI: 10.1007/s00138-010-0287-7.

[104] Marcos Nieto, Andoni Cortés, Oihana Otaegui, Jon Arróspide, and Luis Salgado. "Real-Time Lane Tracking Using Rao-Blackwellized Particle Filter". In: *J. Real-Time Image Process.* 11.1 (Jan. 2016), pp. 179–191. ISSN: 1861-8200. DOI: 10.1007/s11554-012-0315-0. URL: https://doi.org/10.1007/s11554-012-0315-0.

[105] David Nister, Oleg Naroditsky, and James Bergen. "Visual Odometry for Ground Vehicle Applications". In: *Journal of Field Robotics* 23.1 (2006), pp. 3–20.

[106] David Nistér. "An efficient solution to the five-point relative pose problem". In: *IEEE transactions on pattern analysis and machine intelligence* 26.6 (2004), pp. 756–770.

[107] Nvidia. *NVIDIA DRIVE Mapping*. 2020. URL: `https://developer.nvidia.com/drive/drive-mapping`.

[108] Malte Oeljeklaus, Frank Hoffmann, and Torsten Bertram. "A combined recognition and segmentation model for urban traffic scene understanding". In: *Proceedings of 2017 IEEE International Conference Intelligent Transportation System (ITSC)*. IEEE. 2017, pp. 1–6.

[109] OpenStreetMap contributors. *OpenStreetMap*. 2022.

[110] Bowen Pan, Jiankai Sun, Ho Yin Tiga Leung, Alex Andonian, and Bolei Zhou. "Cross-View Semantic Segmentation for Sensing Surroundings". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4867–4873.

[111] Gaurav Pandey, James R. McBride, Silvio Savarese, and Ryan M. Eustice. "Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information". In: *J. Field Robotics* 32 (2015), pp. 696–722.

[112] Chanoh Park, Peyman Moghadam, Soohwan Kim, Sridha Sridharan, and Clinton Fookes. "Spatiotemporal Camera-LiDAR Calibration: A Targetless and Structureless Approach". In: *IEEE Robotics and Automation Letters* 5 (2020), pp. 1556–1563.

[113] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[114] David Paz, Hengyuan Zhang, and Henrik I. Christensen. *TridentNet: A Conditional Generative Model for Dynamic Trajectory Generation*. 2021. arXiv: `2101.06374 [cs.RO]`.

[115] David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik Christensen. "Probabilistic Semantic Mapping for Urban Autonomous Driving Applications". In: *Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[116] Juraj Peršić, Luka Petrović, Ivan Marković, and Ivan Petrović. "Spatiotemporal Multisensor Calibration via Gaussian Processes Moving Target Tracking". In: *IEEE Transactions on Robotics* 37.5 (2021), pp. 1401–1415. DOI: 10.1109/TRO.2021.3061364.

[117] Jonah Philion and Sanja Fidler. "Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D". In: *Proceedings of 2020 European Conference on Computer Vision*. 2020.

[118] 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: *Proceedings of 2018 IEEE Intelligent Transportation System Conference (ITSC)*. Hawaii, USA, Nov. 2018. URL: http://www.mrt.kit.edu/z/publ/download/2018/Poggenhans2018Lanelet2.pdf.

[119] Ptolemy. *Ptolemy's Map*. http://artscimedia.case.edu/wp-content/uploads/sites/190/2016/07/14223557/769.G.2.jpg. Accessed: 2022-03-31. 2021.

[120] Yeqiang Qian, John M Dolan, and Ming Yang. "DLT-Net: Joint detection of drivable areas, lane lines, and traffic objects". In: *IEEE Transactions on Intelligent Transportation Systems (T-ITS)* 21.11 (2019), pp. 4670–4679.

[121] Tong Qin, Peiliang Li, and Shaojie Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[122] Tong Qin and Shaojie Shen. "Online Temporal Calibration for Monocular Visual-Inertial Systems". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3662–3669.

[123] Fitsum A Reda, Guilin Liu, Kevin J Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. "Sdc-net: Video prediction using spatially-displaced convolution". In: *Proceedings of 2018 European Conference on Computer Vision (ECCV)*. 2018, pp. 718–733.

[124] Thomas Roddick and Roberto Cipolla. "Predicting Semantic Map Representations from Images using Pyramid Occupancy Networks". In: *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Virtual, June 2020, pp. 11138–11147.

[125] Oliver Roeth, Daniel Zaum, and Claus Brenner. "Road network reconstruction using reversible jump MCMC simulated annealing based on vehicle trajectories from fleet measurements". In: *Proceedings of 2016 IEEE Intelligent Vehicles Symposium, Proceedings*. Vol. 2016-August. Institute of Electrical and Electronics Engineers Inc., Aug. 2016, pp. 194–201. ISBN: 9781509018215. DOI: 10.1109/IVS.2016.7535385.

[126] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *Proceedings of 2015 International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[127] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[128] Nick Schneider, Florian Piewak, Christoph Stiller, and Uwe Franke. "RegNet: Multimodal sensor registration using deep neural networks". In: *Proceedings of 2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1803–1810.

[129] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. "Generalized-ICP". In: June 2009. DOI: 10.15607/RSS.2009.V.021.

[130] Heiko G. Seif and Xiaolong Hu. "Autonomous Driving in the iCity—HD Maps as a Key Challenge of the Automotive Industry". In: *Engineering* 2.2 (2016), pp. 159–162. ISSN: 2095-8099. DOI: https://doi.org/10.1016/J.ENG.2016.02.010. URL: http://www.sciencedirect.com/science/article/pii/S2095809916309432.

[131] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905.

[132] 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 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 10529–10538.

[133] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. "Point-RCNN: 3D Object Proposal Generation and Detection from Point Cloud". In: *Proceedings of 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 770–779.

[134] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2020).

[135] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. "RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement". In: *Proceedings of 2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2510–2515.

[136] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: (2012). IEEE/RSJ International Conference on Intelligent Robots and Systems.

[137] Quinlan Sykora, Mengye Ren, and Raquel Urtasun. "Multi-Agent Routing Value Iteration Network". In: *Proceedings of 2020 International Conference on Machine Learning (ICML)*. 2020.

[138] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 1848829345.

[139] Yuichi Takeda, Yiyang Zhou, Masayoshi Tomizuka, and Wei Zhan. "Sensor Fusions Strategy for Bird's Eye View Mapping Tasks in Autonomous Driving: A Matter of What, When and How". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), under review*. 2022.

[140] Levente Tamas and Zoltan Kato. "Targetless Calibration of a Lidar - Perspective Camera Pair". In: *Proceedings of 2013 IEEE International Conference on Computer Vision Workshops (ICCV)*. 2013, pp. 668–675. DOI: 10.1109/ICCVW.2013.92.

[141] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *Proceedings of 2019 International Conference on Machine Learning (ICML)*. Long Beach, U.S.A., May 2019, pp. 6105–6114.

[142] C. Vincent Tao and Jonathan Li. *Advances in Mobile Mapping Technology*. ISPRS Book Series v. 4. CRC Press, 2007. ISBN: 9781134090686. URL: https://books.google.com/books?id=uXmmDwAAQBAJ.

[143] Zachary Taylor and Juan Nieto. "Motion-Based Calibration of Multimodal Sensor Extrinsics and Timing Offset Estimation". In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1215–1229. DOI: 10.1109/TRO.2016.2596771.

[144] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. "Multinet: Real-time joint semantic reasoning for autonomous driving". In: *Proceedings of 2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1013–1020.

[145] Haileleol Tibebu, Jamie Roche, Varuna De Silva, and Ahmet Kondoz. "LiDAR-Based Glass Detection for Improved Occupancy Grid Mapping". In: 21.7 (2021). ISSN: 1424-8220. URL: https://www.mdpi.com/1424-8220/21/7/2263.

[146] Chikao Tsuchiya, Yuichi Takeda, and Abdelaziz Khiat. "A Self-Localization Method for Urban Environments using Vehicle-Body-Embedded Off-the-Shelf Sensors". In: *Proceedings of 2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1159–1165. DOI: 10.1109/IVS.2019.8813785.

[147] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. "Sparsity Invariant CNNs". In: *Proceedings of 2017 International Conference on 3D Vision (3DV)*. 2017.

[148] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. "Autonomous Driving in Urban Environments: Boss and the Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–59. ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_1. URL: https://doi.org/10.1007/978-3-642-03991-1_1.

[149] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. "Multi-Task Learning for Dense Prediction Tasks: A Survey". In: *arXiv preprint arXiv:2004.13379* (2020).

[150] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *Proceedings of 2020 Annual Conference on Neural Information Processing Systems (NIPS)*. Long Beach, U.S.A., Dec. 2017, pp. 6000–6010.

[151] Di Wang, Jianru Xue, Zhongxing Tao, Yang Zhong, Dixiao Cui, Shaoyi Du, and Nanning Zheng. "Accurate Mix-Norm-Based Scan Matching". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018.

[152] Zining Wang, Wei Zhan, and Masayoshi Tomizuka. "Fusing Bird's Eye View LIDAR Point Cloud and Front View Camera Image for 3D Object Detection". In: *Proceedings of 2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1–6. DOI: 10.1109/IVS.2018.8500387.

[153] Ziyan Wang, Buyu Liu, Samuel Schulter, and Manmohan Chandraker. "A Parametric Top-View Representation of Complex Road Scenes". In: *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, U.S.A., June 2019, pp. 10317–10325.

[154] *Waymo Open Dataset: An autonomous driving dataset*. 2019.

[155] Xinkai Wei, Ioan Andrei Barsan, Shenlong Wang, Julieta Martinez, and Raquel Urtasun. "Learning to Localize through Compressed Binary Maps". English (US). In: *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, June 2019, pp. 10308–10316. DOI: 10.1109/CVPR.2019.01056.

[156] Weisong Wen, Guohao Zhang, and Li-Ta Hsu. "Correcting GNSS NLOS by 3D LiDAR and Building Height". In: (2019). ION GNSS+ 2019.

[157] Weisong Wen, Yiyang Zhou, Guohao Zhang, Saman Fahandezh-Saadi, Xiwei Bai, Wei Zhan, Masayoshi Tomizuka, and Li-Ta Hsu. "UrbanLoco: A Full Sensor Suite Dataset for Mapping and Localization in Urban Scenes". In: *Proceedings of 2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 2310–2316. DOI: 10.1109/ICRA40945.2020.9196526.

[158] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation". In: *Proceedings of 2020 European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 1–19.

[159] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. "PointFusion: Deep Sensor Fusion For 3D Bounding Box Estimation". In: *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[160] Fuwu Yan, Kewei Wang, Bin Zou, Luqi Tang, Wenbo Li, and Chen Lv. "LiDAR-based multi-task road perception network for autonomous vehicles". In: *IEEE Access* 8 (2020), pp. 86753–86764.

[161] Yan Yan, Yuxing Mao, and Bo Li. "Second: Sparsely Embedded Convolutional Detection". In: *Sensors* 18.10 (2018), p. 3337.

[162] Bin Yang, Ming Liang, and Raquel Urtasun. "HDNET: Exploiting HD Maps for 3D Object Detection". In: *Proceedings of 2018 Annual Conference on Robot Learning (CoRL)*. 2018, pp. 146–155.

[163] Bin Yang, Wenjie Luo, and Raquel Urtasun. "PIXOR: Real-Time 3D Object Detection From Point Clouds". In: *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7652–7660.

[164] Sheng Yang, Xiaoling Zhu, Xing Nian, Lu Feng, Xiaozhi Qu, and Teng Ma. "A robust pose graph approach for city scale LiDAR mapping". In: *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1175–1182. DOI: 10.1109/IROS.2018.8593754.

[165] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review". In: *Sensors* 21.6 (2021). ISSN: 1424-8220. DOI: 10.3390/s21062140. URL: https://www.mdpi.com/1424-8220/21/6/2140.

[166] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[167] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time". In: (2014). Robotics: Science and Systems (RSS).

[168] Zhengyou Zhang. "Iterative Point Matching for Registration of Free-Form Curves". In: *IRA Rapports de Recherche, Programme 4: Robotique, Image et Vision* 4.1658 (1992).

[169] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[170] Yiyang Zhou, Sixu Li, Tao Shu, and Pengwei Zang. *MR.MPC: An MPC-based Framework for Mapping Fleet Routing in Dynamic Urban Scenes*. 2021.

[171] Yiyang Zhou, Yuichi Takeda, Masayoshi Tomizuka, and Wei Zhan. "Automatic Construction of Lane-level HD Maps for Urban Scenes". In: *Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 6649–6656. DOI: 10.1109/IROS51168.2021.9636205.

[172] Yiyang Zhou and Zishuo Zhao. *Atlas: Realtime large-scale localization with organized submaps*. Feb. 2020.

[173] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauß, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph Gustav Keller, Eberhard Kaus, Ralf G. Herrtwich, Clemens Rabe, David Pfeiffer, Frank Lindner, Fridtjof Stein, Friedrich Erbs, Markus Enzweiler, Carsten Knöppel, Jochen Hipp, Martin Haueis, Maximilian Trepte, Carsten Brenk, Andreas Tamke, Mohammad Ghanaat, Markus Braun, Armin Joos, Hans Fritz, Horst Mock, Martin Hein, and Eberhard Zeeb. "Making Bertha Drive—An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine* 6 (2014), pp. 8–20.