NORTHWESTERN UNIVERSITY

Integrating Heterogeneous Traffic Data Sources with High Definition Maps in Autonomous Driving

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Andi Zang

EVANSTON, ILLINOIS

June 2022

ABSTRACT

Integrating Heterogeneous Traffic Data Sources with High Definition Maps in Autonomous Driving

Andi Zang

Automated driving has become a very popular topic in the recent years, and is becoming more and more of a reality. In this new trend, High Definition (HD) maps play an important role in many ways that will provide a safer and more efficient driving experience, especially in terms of path planning and vehicle localization. Challenges and problems in HD maps data extraction, dataset creation, and data usage prediction are consequent on the developing of HD maps.

One of the greatest challenges in automated driving is the ability to acquire, access and query the data pertaining to high resolution 3D objects from multiple heterogeneous sources. Specifically, the information extraction needs to be done by fusing data from both sensors and databases, and with real-time constraints. Existing structures and algorithmic approaches designed for regular maps - or even regular features in High Definition maps - are not capable to handle the various challenges.

In the first chapter (Chapter 2) of this thesis, we review the importance and roles of high resolution 3D objects in High Definition maps being used in autonomous driving applications and summarize the characteristics of 3D objects compared to other regular map features. We also describe an end-to-end pipeline of a system targeting such problems and emphasize the challenges and feasible solutions to each part of the pipeline. Last but not least, we define the quantified evaluation metrics for each task and introduce the dataset that we built for this objective.

Given an overview of HD maps and core application, the next challenge is HD maps component extraction. Other than road furniture such as 3D objects, lane boundary geometry is the most fundimental components of HD map. It is typically created from ground level LiDAR and imagery data, which have their limitations such as prohibitive cost, infrequent update, traffic occlusions, and incomplete coverage. In the next chapter paper (Chapter 3), we propose a novel method to automatically extract lane information from satellite imagery using pixel-wise segmentation, which addresses the aforementioned limitations. We also publish a dataset consists of satellite imagery and the corresponding lane boundaries as ground truth to train, test, and evaluate our method.

Moreover, another challenges in automated driving is the ability to determine the vehicle's location in realtime – a process known as self-localization or ego-localization. An automated driving system must be reliable under harsh conditions and environmental uncertainties (e.g. GPS denial or imprecision), sensor malfunction, road occlusions, poor lighting, and inclement weather. To cope with this myriad of potential problems, systems typically consist of a GPS receiver, in-vehicle sensors (e.g. cameras and LiDAR devices), and 3D HD maps. In Chapter 4, we review state-of-the-art self-localization techniques, and present a benchmark for the task of image-based vehicle self-localization. Our dataset was collected on 10km of the *Warren Freeway* in the San Francisco Area under reasonable traffic and weather conditions. As input to the localization process, we provide timestamp-synchronized, consumer-grade monocular video frames (with camera intrinsic parameters), consumer-grade GPS trajectory, and production-grade 3D HD Maps. For evaluation, we provide survey-grade GPS trajectory. The goal of this dataset is to standardize and formalize the challenge of accurate vehicle self-localization and provide a benchmark to develop and evaluate algorithms.

Furthermore, as the HD maps are limited to a particular geographic area with respect to a given point along a trip, different portions need to be downloaded (and processed) on multiple occasions throughout a given trip, along with the other data from internal and external sources. To close the scope, in Chapter 5, we take a first step towards formalizing the problem of Predicting Map Data Consumption (PMDC) in the future time instants for a given trip, based on a (time) window from its history, and investigate the use of Long Short-Term Memory (LSTM) networks – a special type of Recurrent Neural Networks (RNN). Significant efforts were focused on generating an appropriate dataset for this study, towards which we fused the information available in multiple heterogeneous data sources. We conducted experimental observations demonstrating the benefits of the proposed approach. Part of our efforts focused on generating an appropriate dataset for this study, towards which we fused the information available in multiple heterogeneous real data sources. Our experimental observations demonstrate the benefits of the proposed approach over the baselines.

Last but not least, we take a further step towards providing an effective learning approach for the recently introduced problem of Predicting Map Data Consumption (PDMC) in the future time instants for a given trip. We propose a novel methodology that integrates multiple sources (road network, traffic, historic trips, HD maps) and, for a given trip, enables prediction of the map data consumption.

Acknowledgements

First and foremost, I would say thank you to my advisor, Professor Goce Trajcevski. As a great advisor, Professor Trajcevski shows his rigor, wisdom, patience, encouragement, and extreme passion in research (yes, he sends emails at 4:00 AM). His insightful thinking guided (and pushed) me to sharpen my knowledge and vision in the domain of High Definition (HD) maps. As a close friend, we had wide-ranging discussions outside academia that covers politics, history, and boring daily life. I still remember the first time I stepped in his L360 in 2015 (before I came to Northwestern). He silently appeared in L580 when Leo was playing League of Legends. Every night when I left Tech around 11:00 PM, he always standing outside the building and smoking like an NPC. Before every deadline, he spent hours and hours to revise my terrible drafts... There are a lot of such wonderful (not always) moments will bring me back to my academia life.

I would also show appreciation to Professor Xin Chen, who guided me to the field of HD maps when this concept was an uncultivated land in 2014. He is a passionate and creative person and always comes up with interested ideas. He was my teacher (at university), supervisor (at company) and is my great not-that-old friend.

In addition, I would like to acknowledge my committee member Professor Aleksandar Kuzmanovic; my university lab mate and colleges Xiaofeng Zhu, Yuxiang

Guo, Shiyu Luo and Ce Li; my internship mentor and colleges David Doria, Zichen Li and Runsheng Xu. Thank you all for your great help and indispensable contributions to our works.

Special thanks to Northwestern EECS/CS department offices and HERE Technologies, and the warm-hearted staff for their profession and kindness. It was my pleasure working with you.

Finally, I would like to thank my parents. You are always there for me.

I cannot have completed this dissertation without the support of all these great people aforementioned and not mentioned.

My research was partially supported by National Science Foundation, National Natural Science Foundation of China and HERE Technologies.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction and Motivation

Partly and fully autonomous driving systems, as well as other autonomous platforms and systems have been extensively studied throughout the recent years [**21**]. In addition to sensing environmental and traffic-related values, one of the biggest challenges in automated driving is the ability to localize the vehicle position with a high precision, typically within ten-centimeter-level accuracy[1]. High Definition (HD) maps play an important role in this task [**11, 165, 232**], as they can aid assisting/correcting the unreliable and unstable GPS-based self-localization methods. Such approaches are also known as map-based techniques, in contrast to the sensor-based techniques. This type of application – real-time vehicle self-localization using HD maps– consumes significant computational resources, and needs consistently instance responses – also known as “real-time”, bring us a whole new challenge we propose as “real-time applications in HD maps”.

1.1. Understanding the HD Maps

Typical HD maps mainly consist of lane boundary geometries, road signs and 3D objects, as illustrated Figure 1.1. Painted lines and physical road boundaries (shown in green lines in Figure 1.1) offered in a lane model can guide the vehicle where to

[1]The precision requirements of high level autonomous driving from different companies and institutions vary from 20-centimeter-level to centimeter-level.

stay while driving - or, more specifically, localize the vehicle laterally. Lane boundary geometry related extraction/detection and vehicle self-localization approaches have been widely discussed in several state-of-the-art papers [**143, 234, 137**] and the results presented have demonstrated nice performances. In basic terms, the lane boundary geometry is composed by points (representation can be either polyline or spline), attached with other descriptive tags such as lane type (e.g., solid line, dashed, etc.), ID, color and width. The data size of this type of lane model is relatively small - for example, as reported for the HD maps dataset in [**232**], the number of points per meter is typically less than 10^2, which makes storing, retrieving and query processing over such data type relatively easy/efficient. To localize the vehicle longitudinally, non-repetitive markers along the particular road direction are needed, so road signs (including both the text information and the signs themselves) are involved. This lateral and longitudinal localization combination is still under developing, and a recent demo was presented by NVidia in Consumer Electronics Show 2019[3]. Additionally, we note that these datasets are sparse, so the data size is quite compact and well defined in the Navigation Data Standard (NDS) [**179**].

Being different from lane boundary geometry and road signs, the data size used to describe 3D objects (shown in yellow voxels in Figure 1.1) is enormously larger,

[2]The number of points in a lane model depends on the number of lanes (including shoulders) and the actual polynomial interpolation method. Typically, one only needs three control points to describe a line/curve in a short distance.

[3]NVIDIA Teams Up with Leading HD Mapping Companies to Deliver End-to-End Autopilot Systems for the World's Major Markets: `shorturl.at/bkDOP`

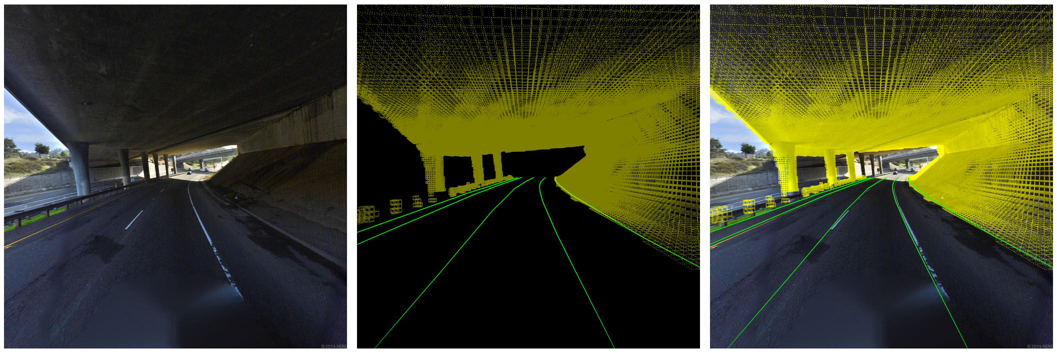


Figure 1.1. Illustration of a) a common highway under an overhead structure scenario, b) visualizations of two major HD maps components: lane marking geometry in green polylines and 3D object in yellow voxels, and c) the overlay of HD maps components on real world image.

especially at higher resolution. The current Navigation Data Standard (NDS) specifies that the data simply uses point representation to store the 3D objects [**179**]. However, even with the state-of-the-art workflow, the existing solutions are not sufficiently tailored to handle large-scale data storing and efficient query processing, as well as other operations important for real-time settings (e.g., autonomous passenger vehicles, trucks and drones), when the size of data increases several orders of magnitude.

Real-time (from observation to decision) is another critical challenge that such applications need to tackle. Human reaction times in traffic are ranging from $200ms$ [**87**] to the order of $1s$ [**90, 78**] and varies between tasks (i.e., from brake reaction times to much more complicated tasks). A safe autonomous driving system should be no slower than human reaction time, and the faster the better. At the same time, a consistent fast response time is also mandatory.

Typically the applications that need 3D objects are computationally expensive from both acquisition-end (from sensors) and database-end (from maps). For instance. We postulate that, in order to efficiently use the 3D objects in HD maps for application with real-time constraints, methodologies and tools are needed that will simultaneously optimize the transmission, computational and storage costs.

1.2. HD Maps Furniture Extraction

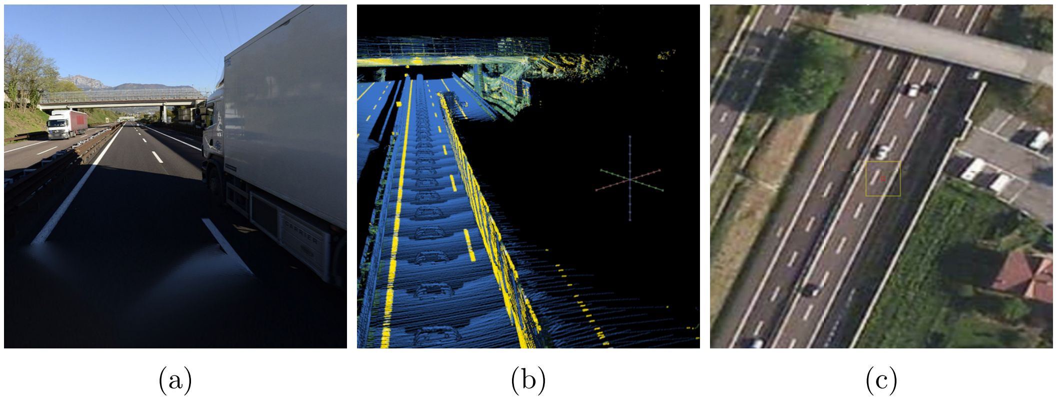


Figure 1.2. Object occlusion in perspective view (a), a wall caused by occlusion in LIDAR point cloud (b), and satellite image (c).

Recently, a lot of work has been done to automate lane-level map generation using vehicle-sensor meta data crowd-sourced from large fleet of vehicles [**32**] in addition to ground level data such as imagery [**30, 31, 28**], LiDAR [**236**], GPS, and Inertial Measurement Unit (IMU) collected by mobile mapping vehicles.

Extracting road/lane information from airborne imagery has its advantage over terrestrial data due to its comprehensive coverage, low-cost, ease to update. The history of road extraction from orthogonal imagery (e.g. satellite and aerial) can be traced back to more than forty years ago; however, limited by image resolution (typically over 2 meters per pixel), traditional approaches rely on edge detection, color segmentation, linear feature detection, and topological linking [**196, 6**] to extract road networks from overhead imagery. In recent years, more machine-learning based approaches are proposed to detect patch/pixel-wise road region [**123, 73, 25, 124, 76**]. These road centric approaches still cannot model the lane-level features even though the common satellite image resolution has improved to 0.5 meter per pixel.

Now, satellite imagery can have a resolution of 0.5 meter per pixel or less, which allows us to utilize the classic approaches with much detailed imagery to model lane-level features [**119, 120, 168, 143**]. There are still two challenges after the lane boundary line is successfully detected: the representation of the lane model and the evaluation metric of model accuracy and performance. In paper [**143**], the road model is represented as a collection of unstructured lines without attributes; while in paper [**168**], the definition of accuracy is based on the percentage of pixel-wise overlap comparing to their manually drawn line masks in the input images. Hence, their claimed accuracy is less persuasive.

Autonomous vehicles are becoming more of a reality. The increasing demand of HD mapping can be predicted, especially for interstate transportation (i.e. autonomous truck [**55**]. The three largest highway networks in the world, U.S., China, and India, are 103, 446, and 79 thousand kilometers [**1, 185**] long, respectively, which motivates us to concentrate on highway-level road network in this dissertation. We propose a novel, automated lane boundary extraction technique from satellite imagery. Our approach consists of two stages: pixel-wise line segmentation and hypotheses-grouping classification linking. The pixel-wise line segmentation approach contains patch-based lane-marking classification, and for each positive patch, we segment line pixels to generate line candidates. Hypotheses-linking connects each line candidate by minimizing the proposed cost function to generate structured lane model. A formalized road-model-accuracy-metric is designed to evaluate the results rigorously. We also manually extracted lane boundary ground truth from our dataset. Along with satellite imagery, it can be used for training, testing, and evaluation for comparative studies.

1.3. HD Maps Dataset

Vehicle self-localization (ego-localization) is a key component of autonomous driving and often depends on a combination of sensor-based and HD-Map-based location data. It demands high precision, real-time, and robust data management and algorithmic techniques that can handle very harsh conditions such as GPS denial/imprecision, traffic occlusion, and low lighting. Moreover, autonomous vehicles must operate on roads in the presence of non-automated vehicles with unpredictable human drivers for the next decade[4]. Finally, the price that consumers are willing

[4]10 Million Self-Driving Cars Will Hit The Road By 2020 (Forbes: shorturl.at/deC03)

to pay for an automotive technology module is quite low[5] [6]. All these factors lead to a conclusion that self-localization solutions must be reasonably priced, easily installed/integrated, and calibration/maintenance free.

A Global Navigation Satellite System (GNSS) typically localizes receiver ranging from 100-meter level to 10-centimeter level by using low cost GPS or high-end Differential GPS (DGPS), which is known as active self-localization, in both harsh or ideal condition. Combining maps (normally standard definition maps) and raw GPS point/trajectory, several location correction [**187**] and map matching [**13, 147**] approaches have been designed to tackle vehicle self-localization. Such approaches have generated promising results when the task was to localize the vehicle to a particular road. However, for lane-level self-localization, there are two factors that limit the performance of GNSS-based localization: (1) Environment sensitivity and reliability – generally speaking, GNSS in urban environments suffers inaccuracy due to multi-path interference, ionospheric delay, obstructions, and even satellite clock error/unsynchronization[7]. Even if the vehicle is equipped with a survey-grade DGPS device, the localization performance (accuracy and reliability) is insufficient for the purpose of autonomous driving; (2) The unavoidable misalignment (relative error) between HD Map coordinates and GNSS coordinates. This is a problem since vehicles need to compute their locations in map coordinates for self-localization purposes.

[5]Deloitte Study: Fact, Fiction and Fear Cloud Future of Autonomous Vehicles in the Minds of Consumers, `shorturl.at/gvOT6`
[6]Intel: Autonomous Driving Survey, `shorturl.at/nMRU3`
[7]Global Positioning System - Augmentation Systems: `http://www.gps.gov/systems/augmentations/`

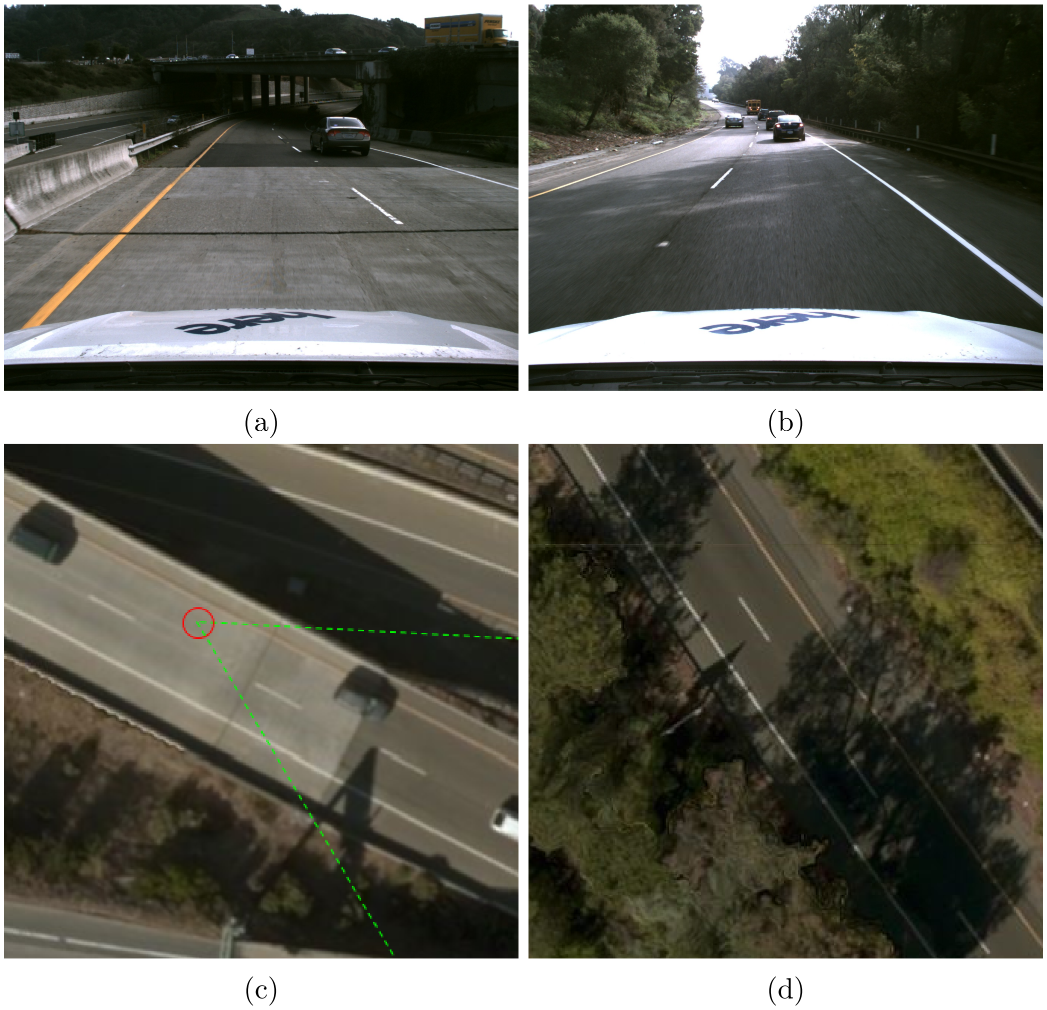


Figure 1.3. Illustration of the challenge, given real-time images (a) and (b), locate vehicle positions in corresponding road models (c) and (d) (use satellite images as examples). In case (a) and (c), localization is easy due to the remarkable visual feature, but is opposite in case (b) and (d).

With commercialization and miniaturization of LiDAR devices, many Simultaneous Localization And Mapping (SLAM) systems have been proposed. LiDAR techniques are known for their high precision as 3D information is captured directly, compared to reconstructing 3D information from a 2D stereo camera. These two approaches are typically used to solve point cloud based localization. SLAM can be performed either in an unknown environment [**147, 40, 88**] or a known environment (i.e. map) [**226, 211, 99, 100, 166**]. Alternatively, lane-level objects such as lane markings [**157**], pole-like objects [**19, 159, 178**], curbs [**52**], and even occupancy grids [**118, 42, 209**] can be detected and used for self-localization. Using additional information such as HD Maps, features can be used to estimate vehicle/camera position using triangulation. While LiDAR-based solutions are superior in terms of effectiveness, their shortcomings in terms of cost and weather dependency [**54**] limit their use in performing point-cloud based self-localization and cannot be ignored.

Compared to point-cloud based localization, image-based localization has been widely utilized in the self-localization domain for over a decade, due to its ease of acquisition, installation, and low cost. With stereo cameras, a 2D scene can be reconstructed to 2.5D, or even 3D information. With monocular cameras (e.g., dash cameras and cellphone cameras), the proposed methodology can be divided into large scale (city level), medium scale (city block level), and small scale (road level). Perspective images and (omnidirectional/orthogonal) aerial images are two major data sources that can be used to match and localize a query image. Image-based localization in large scale using perspective image to perspective image matching [**239, 44, 27**] and perspective image to aerial image matching [**170**] are designed for characterizing city identity. These techniques have the highest localization error and limited application. With the increasing demand of Location-based Services (LBS), quite a few medium-scale image-based localization approaches which return rough locations by using perspective image to perspective image matching [**97, 141, 114, 199, 200, 53, 92**] and perspective image to aerial image [**195, 8, 24, 107, 7, 106, 212, 216, 26**] have been proposed. Small scale image-based localization always has a specific purpose that requires relatively high precision like unmanned vehicle navigation in GPS denied/jammed situations [**200**], road level self-localization [**213, 130, 198, 142, 117**], and even more precise lane-level self-localization [**85, 161, 218, 148, 36, 146, 218, 182**] by using detailed local patterns. Beyond that, numerous papers that utilize image and LiDAR point cloud fusion [**211**] have been proposed to tackle the localization problem.

An important aspect of evaluating such techniques is providing a benchmark, and several datasets have been published for that purpose in the context of self-localization [**180, 193, 194**]. However, vehicle self-localization remains an uncharted domain and, in an attempt to standardize the vehicle self-localization problem, this work introduces a dataset that contains:

- **•:** manually coded, high precision 3D static map containing lane boundaries (geometry, color, and function), road signs (text and spatial location), and occupancy grid voxels;
- **•:** consumer-grade dash camera image sequence and GPS (synchronized with timestamps);

- **•:** high precision GPS locations (with associated timestamps) aligned with the provided map coordinates;
- **•:** camera configuration file containing camera calibration and distortion parameters.

The dataset includes realistic scenarios that would be typically encountered by an autonomous vehicle. This is the first version of the dataset, and we are open to feedback and suggestions from the community (for example, additional features/attributes, extended coverage, etc.). A blinded test dataset is also prepared to persuasively and fairly evaluate any proposed algorithm for its performance and generalization. Currently our dataset can be requested by email.

1.3.1. HD Maps Data Usage Prediction

Multiple on-board vehicular sensors, along with external data sources (e.g., traffic, weather), are generating various data types which, in turn, are enabling multiple functionalities (efficiency, re-routing), improving the overall trip experience. One particular source important for the overall navigation in autonomous and assisted driving settings are the HD maps [**111**]. While crucial for improving self-localization and safety [**165, 232, 233**] – they are also the largest consumers of bandwidth and processing power. State-of-the-art applications using HD maps are still built on the old chassis – conventional "Standard Definition" (SD) maps, which consist of road networks, topology and limited road features/objects attached to links and nodes.

However, HD maps are much richer in terms of the sets (and types) of objects that they provide.

3D data with higher resolution and accuracy can be acquired from either multi-image reconstruction or point cloud along with the improvement of acquisition hardware and algorithms. Multiple objects ("furniture") are used to help the autonomous vehicle to make decisions, such as lane boundaries, curbs/guardrails, and pole-like objects. Combining these two aspects (acquisition-end and application-end) amplifies the impact of (the increase of) the size of the map data, relative to when the concept of HD maps were first introduced a decade ago. Nowadays, the HD maps can easily contain over a thousand voxels (highway scenario) or even tens of thousands of voxels (urban scenario) per road meter at a higher resolution, in contrast to dozens of points per road link in conventional maps [**233**]. This, in turn, affects the processing that enables decisions made by the drivers and autonomous vehicles, as the complexity of many important algorithms (e.g., points registration [**153**] and segmentation [**129**]), brings them on the cusp of being computationally over-expensive for certain application scenarios, given hardware limitations. To tackle the problem of optimizing the use of ever increasing (demands for) map data in real-time applications, existing approaches are mainly focusing on improving data structures and data flow.

Researchers and companies focusing on improving the HD map structures have proposed their own models to shrink the size of the data. In general, hierarchical structures are downloading different resolutions on demand by various use cases, which could potentially decrease the data throughput from the server to the vehicle. Due to the characteristics of HD maps and the applications using them, downloading (or pre-fetching) the data for entire trip at once is practically infeasible under current hardware constrains. This, in turn, implies that data needs to be downloaded and processed multiple times during a particular trip – which requires efficient management, based on various factors related to a particular vehicle and trip.

Similar problems are encountered in hybrid electric vehicle energy management [**57**]. If the energy consumption of each trajectory point can be predicted entirely before the trip, the vehicle itself can have a better energy management plan to improve fuel economy (the combination between conventional energy and renewable energy) and to reduce emissions. However, given different fluctuations in traffic, even in this scenarios, one needs to re-asses the energy consumption in points throughout the trip. What further complicates the matters is that multiple external factors (e.g., traffic fluctuations, weather changes, etc.) may still affect the quality of various predictions in real time.

In this dissertation we take a fist step towards considering the problem of Map Data Consumption (MDC) and introduce the Prediction of Map Data Consumption (PMDC) for a given trip. Since, to our knowledge, the PMDC problem has not been formally addressed, one of the consequences is that there are no existing datasets built/collected for this task. To this end, we integrated data obtained from multiple sources and studies related to maps, trips and traffic data. We created a "synthetic city" dataset (SCD) to be used in both training and experiments, and checked the use of an LSTM-based approach towards the PMDC problem, including additional (internal and external) data sources.

The primary contributions are summarized as follows:

- **•:** we first define the MDC problem follows a straightforward idea, and then solve the a sub-PMDC-problem by using a navie LSTM structure to predict the MDC consumption for next portion/segment of a trip, to proof the feasibility of the PMDC.
- **•:** we refine the definition of MDC problem. More rigorous expressions are given to formalize the input and output of this task;
- **•:** we propose a novel neural network structure that takes heterogeneous information as input (graph/trips and tiles) to solve PMDC problem;
- **•:** the PMDC problem now solved by using GNN-based solution which learns not only local sequential properties (from a trip), but spatial and temporal information from adjacent edges;
- **•:** we provide experimental evaluation over a synthetic dataset, demonstrating that our solution provides significant improvements over baselines.

1.4. Thesis Outline

As the forerunner in this field, in this dissertation, we tackle several challenges at different HD maps stages, from basic HD maps domain knowledge and structure, HD maps furniture extraction, HD maps dataset and map data consumption prediction. The rest of this dissertation is organized in five chapters. In Chapter 2, we present a detailed introduction to HD maps components, applications and challenges. In Chapter 3, we design a extraction technique of lane marking geometries from overhead imagery. A HD map dataset – which is the first HD maps dataset in both industry and academia – is introduced in Chapter 4. As the core contribution of this dissertation, the groundbreaking HD map data consumption problem is proposed and solved in Chapter 5. Finally, we summarize the concluding remarks, discussions and future improvements of each stage of the HD maps creation.

CHAPTER 2

Introduction to High Definition Maps

2.1. Introduction and Motivation

In this chapter, we provide study the challenges that are faced when designing a system for real-time applications – specifically, ones that typically need very large datasets, with a high quality and high resolution 3D objects. The study has three main objectives:

(1) Provide understanding of the 3D objects and the pipeline that enables their use in map systems. The concept of using specific kinds of 3D object in specific cases has been widely studied (e.g., feature-based 3D localization, object recognition and detection, etc.). However, to our knowledge, there are no systematic studies on how to seamlessly integrate 3D object in a system - especially for real-time applications. Towards this, we will discuss the 3D objects in several aspects: from the representation/coordinates, to their real-world features such as density, quality and distribution.

(2) Introduce a "template" pipeline to be used by applications that require large scale and high precision 3D objects. For each step, we will discuss the potential challenges and solutions. The end-to-end (from data acquisition to application) pipeline using 3D objects is relatively similar to most of feature-based applications, but the critical

difference is that the data size is much larger and the data structures used can be complicated. In addition to the aspects and limitations due to the characteristics of the hardware (i.e., cpu, memory, transmission bandwidth, etc.), the problems related to efficiently orchestrating different modules bring a specific set of challenges to be tackled, and more/different trade-offs to be considered.

(3) Define and formalize the evaluation of the different sub-problems arising in different modules constituting a particular system. For the core challenge of the entire system - which is the system performance - we take steps towards providing definitions of what is the performance in this type of system. We also render several real-world cases to make such definitions easier to understand.

2.2. Preliminaries

We now discuss in detail the importance of the 3D objects, and provide separate discussions related to their representation and storage.

Typically, 3D objects consist of voxels/grids (also known as occupancy grids) and semantic objects for different purposes/applications such as visualization, self-localization and route planning. Pole-like objects [**205, 207, 19, 125, 164, 18, 159, 155, 98**], lane markings [**161, 81, 149, 160, 70, 9, 62**], curbs [**71, 72**], signs [**206, 146, 138**], surfaces/corns [**19, 80, 79**] and even voxels/grids [**101, 121, 160, 109, 81**] are the popular objects being used in these applications.

While being compact and easy to manage, which makes them suitable for real-time sensing and processing – using painted line as a self-localization attribute has inevitable limitations.Namely, a painted line is easy to be mis-detected which, in turn, can cause serious accidents involving autonomous vehicles. The paint always wears out over time due to heavy traffic and weather condition (according to surveys, lane markings (and other painted signs) wear out due to the volume of traffic and inclement weather, result in a service life ranging from 1 to 5 years due to different materials being used [**132, 173**]). Moreover, lane boundaries cannot be repainted immediately by road maintenance service. From a complementary perspective, another limitation is that they can only be used to localize the vehicle laterally, since this feature is repetitive along road direction (longitudinal). Road signs are distinctive markers that can be used to triangulate vehicle location longitudinally – however, this kind of markers are rather infrequently placed along actual road segments. By comparison, road boundary (e.g., guardrail and curb) and pole-like (e.g., light pole and traffic light) objects stay relatively longer (the lifespan of a pole is ranging from 30 to 60 years varies between highway and urban, and type of the pole [**126, 186**]) and occur more frequently in road network. Last but not least, objects should also be robust to weather from both pysical and sensor perspectives. For instance, lane markings may become invisible from a LiDAR sensor in rain due to low reflectance [**134**]. Snow can also cover lane markings and other features on a horizontal plane. All of these components, when utilized for HD maps, aim to build a safe(r) autonomous driving system – however, the 3D objects can offer a higher level of precision/robustness.

A specific benefit of using 3D objects is that they can include high definition terrain model, which can be used in several autonomous driving applications, such as high precision path planning [**172**]. The detailed road surface contour information ahead of vehicle sensing range gives the vehicle plenty of time to adjust its suspension and traction systems – not to mention other self-driving benefits, such as better fuel economy and improved safety.

Last but not least, considering that the autonomous cars and trucks will not be the only autonomous platform in future, the 3D objects can also be used in airborne platform (for example, drones) for survey, surveillance and logistics purposes. Similarly for other kinds ofmobile platforms like, for example, assisting the navigation for impaired individual (e.g., path planning, blind navigation, etc).

2.2.1. 3D Object Representation

When designing a system that is to operate with certain types of spatial data, there are always certain foregoing questions to be asked: how to represent the data, which coordinate system to use, and what other attributes does the data have. In this section, we provide a brief introduction of 3D object representation in discrete point, voxel, and polygon, 3D object coordinates (from 1D to 3D), local to global, and 3D object attributes.

Representation: There are three basic types to store these 3D objects: discrete points (non-uniformly sampled points), voxels (uniformly sampled over a grid, i.e., raster data) and polygons/polylines (i.e., vector data). Figure 2.2 illustrates the transformations between these three types of objects representation. The raw points can be "polygonized" in order to generate polygons (meshes) to represent the objects, or they can be uniformly sampled to generate gridded voxels. Depending on the polygonization approach, some meshes can be converted to equivalent or lower resolution voxels, but not the opposite.

On both creation-end of HD maps, as well as (most of) the vehicle-ends, the inputs are always discrete points acquired by vehicular sensors (e.g., rotating LIDAR, solid-state LIDAR, depth cameras, etc.). An entry level industrial LIDAR unit (e.g., Velodyne VLP-16) can easily produce 10^5 points per second, even though many of the points may be missing (either caused by target being out of the sensing range, or target being too dark and absorbing the laser). Saving and subsequently using the vast number of such points to localize vehicle directly (on the fly) is infeasible. Most of the time, the raw points are too dense. As shown in Figure 2.1, nearly 80% of the points are close to other points – i.e., within 10 centimeters. The nearest point to point distance distribution depends on many factors such as LIDAR configurations (e.g., refresh rate, layout and sensor orientation), vehicle speed and scenario itself. Thus, often times, the raw discrete points need to be simplified and converted to other representations to satisfy the requirements of the system, especially the space–time-quality trade-off. Polygons and voxels are the most common representations used in practice.

In computer graphics, objects are often saved in polygons (triangles) for more lifelike, faster and smoother rendering. The vector-like represent has many other

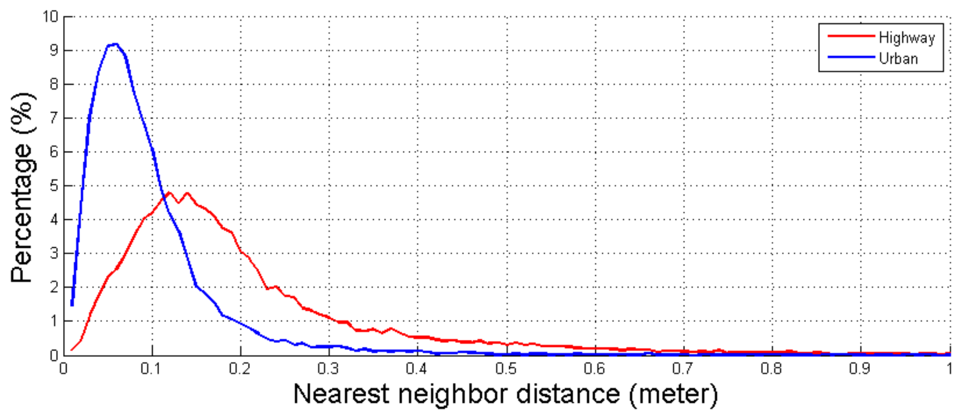


Figure 2.1. Histograms of point to its nearest point distances of sample partitions from urban street and highway scenarios. Majority of distances is below 20 centimeters for both urban and highway scenarios, which are 93.08% and 64.84% respectively. Very few points (0.7% and 6.14%) are further than 50 centimeters from its nearest point in these two scenarios.

advantages such as scale-invariance, shift-invariance, and anti-aliasing (orientation-invariant). However, it has uncertainty associated with the compression ratio (uncompressed size over compressed size). For example, the more regular shaped objects the particular scenario contains, the higher (better) compression ratio one can obtain. In certain settings, it may get quite bad (less than 1) compression ratio if the scenario contains lots of irregular shaped objects such as bushes [**60**]. The method that converts discrete points to triangles in 3D space is called Tetrahedralization [**150, 108**]. With points reduction and polygon simplification (e.g., triangle merging, plane extraction, etc.), the points can be converted to an abstracted mesh model, but the cost is extremely expensive and the result is sometimes unpredictable (i.e., no assurances in terms of robustness) [**144, 34, 50, 181**]. Last but not the least, partitioning and

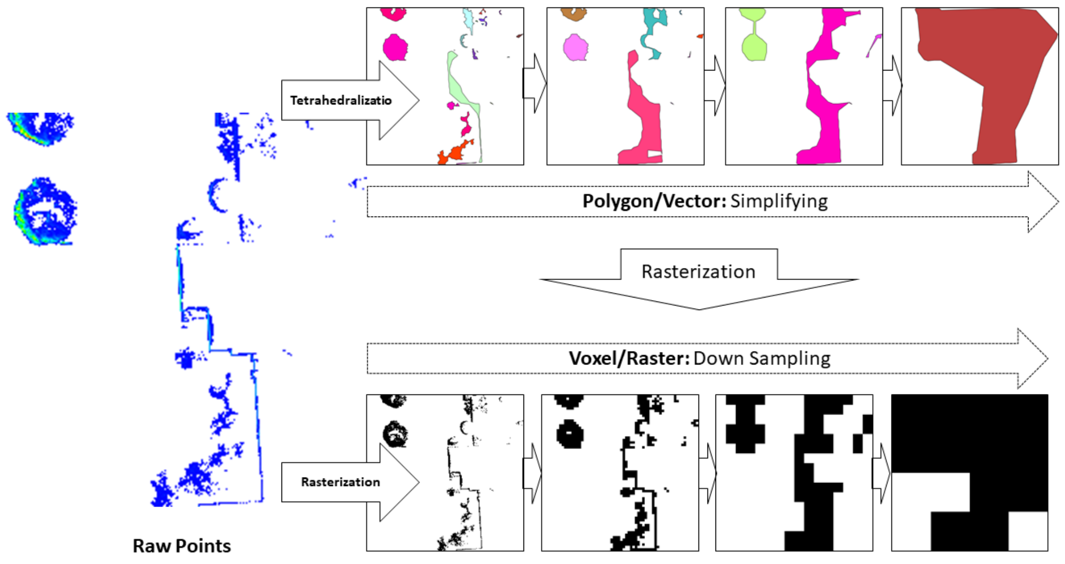


Figure 2.2. Raw points and two point representation: polygon (top row) and voxel (bottom row), with decreasing level of details. This scenario contains two trees (on the left, two circles), a building (one the right) and some vegetation in front.

merging meshes into/from adjacent containers/portions is also computationally expensive. These are the main reasons why in majority of the applications vectors are used to represent simple and sparse objects (e.g., road networks, building footprints, area contours, etc.).

In robotics, for localization and mapping purposes, voxel-like data – or in general, raster map – is most frequently used[**67, 68, 135, 5**]. The conversion is known as *rasterization*. Compared to vector maps, raster data has the advantage of simplicity, robustness and computationally efficiency of the processing algorithms. Almost every operation (e.g., rasterize, merge, update, etc.) is within a linear complexity bound.

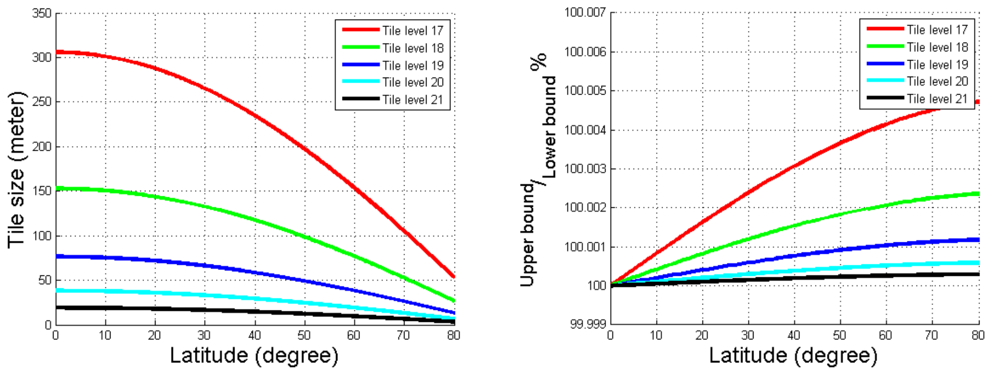
However, there are disadvantages, including the aliasing problem, less reusability and less extensibility, mainly because the voxels orientations and resolution are fixed when the raw points are being rasterized [**23**]. When it comes to aliasing, reducing voxel size can "smooth" out the jaggy problem, but it will increase the number of voxels within a cubic-order. In vector representation, polygons can be converted to voxels on demand [**75**], but it is dependent on the required resolution and orientation. This, however, would alleviate the aliasing problem, as well as the information loss (from resampling and upsampling) and spatial distortion. But at the same time, triangulation algorithms are not robust enough, and sometimes those algorithms can introduce non-existent lines and fill out non-existent holes which yields more uncertainties to cope with.

Lastly, we note that raster data is potentially compatible with most of cutting edge machine learning techniques, especially deep learning for object classification [**177, 219**] and segmentation [**145**].

3D Object Coordinate: 1D Road, 2D Mercator Tile, 3D Cartesian Grid: There are three types of containers that we can use to organize the road objects which are distinct from the container dimensions: attach objects to one-dimensional road network , two-dimensional tile and three-dimensional grid. One-dimensional data structure is rarely used [**192, 102**], because the objects do not have a global view, which makes global optimization and alignment harder. Multiple adjacent road links need to store the objects individually which, in turn, causes data redundancy.

2D tile structure - more specifically, Mercator projection - is the most widely used map data structure, in domains that include satellite imagery, elevation models, road networks, buildings and other geospatial data. Storing 3D objects in tile structure yields a better compatibility with existing map systems. As all 2D map projections, tile structure potentially has spatial distortion along with the changes in the latitude. Figure 2.3 illustrates the distortion cross-tile and in-tile. The in-tile distortion (lower bound size over upper bound size) is negligible (cf. Figure 2.3 (b)) – however, the cross-tile distortion is significant (cf. Figure 2.3 (a)). In vector map representation, this cross-tile distortion only means a tile at equator contains roughly 6^2 timesmore objects than a tile at latitude 80°, assuming that the objects on earth's surface are uniformly distributed. But if we use raster representation, this causes either the number of voxels to become 6^3 times greater, or a voxel is 6^3 times larger.

When it comes to the 3D grid - which, in most cases is based on a global Cartesian coordinate, for instance, earth-centered, earth-fixed (ECEF) - one can perceive it as a 3D version of 2D tile partition without spatial distortion (because no projection is needed). Using 3D voxel representation in 3D grid then seems be the most obvious solution. However, there is a drawback of a global Cartesian coordinate that is important for many practical scenarios: the orientation of voxel looks counter-intuitive. Because man-made objects are built perpendicular to the surface, a ECEF-like coordinate may cut vertical structures such as buildings and poles into two adjacent sub-containers.



(a) (b)

Figure 2.3. Illustrations of tile spatial distortion. a) tile bound size chart along with latitude from 0° to 80° and b) in-tile spatial distortion (lower bound size over upper bound size).

Attributes: An occupancy voxel in HD maps is more than a simple 3D spatial representation, although presently many implementations rely on using occupancy grid to localize vehicle position [**102, 192**]. Similarly to the concepts used in robotics, depth/distance information from the occupied voxel to vehicle is used to localize the position in a given map coordinate [**67, 68, 135**].

However, in most of the practical scenarios using HD maps, the voxels have other attributes. Following are but a few examples of other attributes, based upon which the objects can be categorized into: (i) stationary or moving[1] (e.g., a parking vehicle versus a moving vehicle); (ii) durable or ephemeral[2] (e.g., curb versus tree crown); (iii) road facility/component[3]; (iv) non-road component (e.g., vehicle and

[1]In this work we assume the object is stationary when the data is collected

[2]Temporal-related categories in this work relate to the physical construction period.

[3]The objects compose entire road network, containing: road surface, road boundary (e.g., curb, end of pavement, guardrail and pole), mark, traffic light, light pole, etc.

pedestrian). Different objects can be assigned with different attributes for better localization and occlusion avoidance performance.

Moving and stationary vehicles and pedestrians can also be captured by 3D data acquisition devices, and the motion is frozen in point cloud. Manually removing the points belonging to vehicles and pedestrians is extremely labor intensive [**61**]. On the other hand, using moving object classification based approaches to remove these objects is computational expensive, risky and not robust, due to the complexity of real world street scenario and too many variables involved [**46, 89, 152**]. A background extraction from fusing multiple point clouds of a same geo-region seems like the most robust approach and, at the same time, feasible to deliver an industrial level product [**83, 82**]. As a side-effect, though, this may require the system to contain an attribute for stationary object's probability. Furthermore, vegetation – especially tree crowns – changes seasonally, and the change may not be distinguished from two independent acquisitions – whereas users (vehicles) need the data every second. Hence, even for those stationary objects, we still need to add an attribute for localization confidence. In sum, the occupancy grids for the street objects not only represent the occupancy status (i.e., occupied or not), but also have require a collection of (attached) attributes.

2.2.2. Characteristics of Occupancy Voxel

Quantity: To design and build a system that stores all objects all around the world, firstly, we need to estimate the total volume of these objects (i.e., how many voxels do we need to handle. There are two straightforward and feasible approaches to estimate this number. The first approach is based on the length of road segments. Specifically, one can divide the road network into two categories: highway and city. Then the total number of voxels can be calculated by using the road length of each category and voxel density (voxel per road length) of each category. Clearly, errors will occur due to several reasons, such as intersection (multiple road links overlap), road with traffic island (which one road is labeled as two separate links in map) and special structures (e.g., tunnels and overpasses).

Another possible approach is the, so called, area based. According to the survey data from [**133, 22**], the total urban road length and urban area of Contiguous United States[4] are $1.95 * 10^9$ meters and $2.74 * 10^{11}$ square meters respectively, while the corresponding voxel densities are $1.26 * 10^4$ voxels per meter and 78.7 voxels per square meter at 10 centimeters voxel resolution as shown in Figure 2.4.

The above two approaches for estimating the size of voxels in HD maps lead to the total number of urban voxels of $2.46 * 10^{13}$ and $2.16 * 10^{13}$ from road-length-based and area-based, respectively.

Density and Distribution: We observe that, despite the size of the data, the occupancy voxels are overall sparse in spatial sense, from earth-scale to street-scale. At earth-scale, urban areas occupy roughly 3% of the earth's land surface[5] (in US, the

[4]Alaska and Hawaii are excluded.

[5]The Growing Urbanization of the World, The Earth Institute, Columbia University: `shorturl.at/gjkHY`

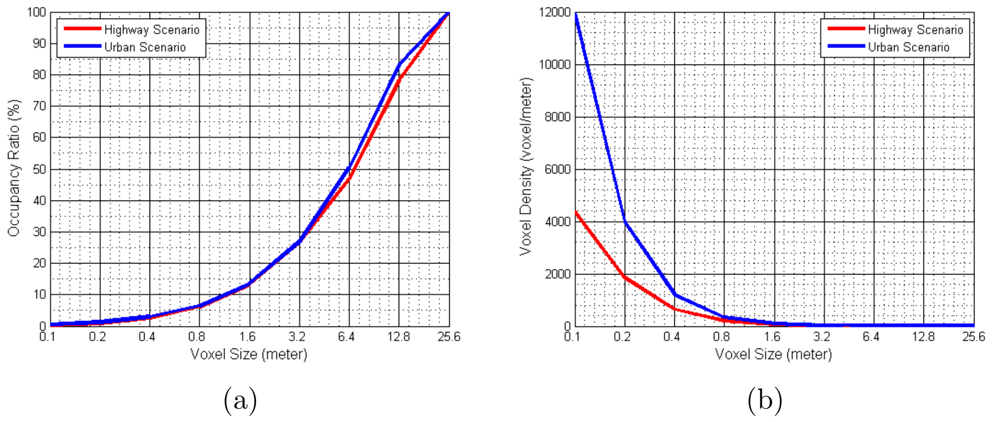


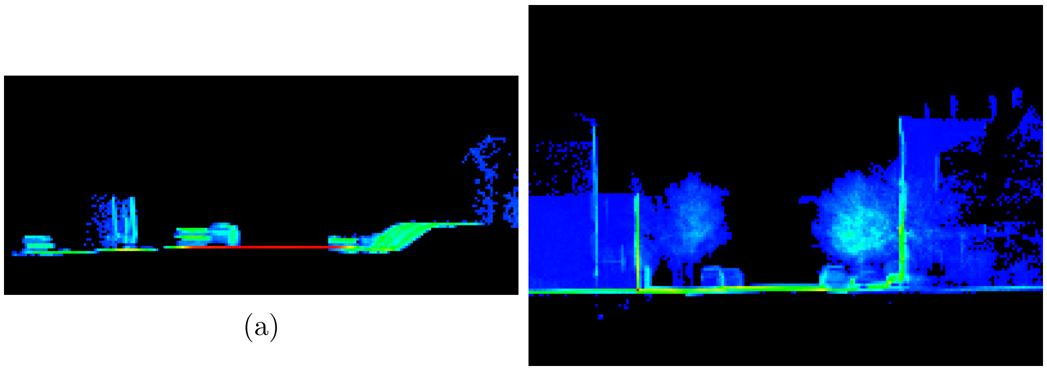
Figure 2.4. Illustrations of (a) grid occupancy ratio with voxel size (longitudinally along the road), and (b) occupied voxels density (number of occupied voxels per meter longitudinal to the road) with voxel size in urban and highway scenarios.

number is 3.06% according to US Bureau of the Census[6] for reference). Meanwhile, the land area only covers 29% of the entire earth. In addition, combining the road network area in rural area in US - which is approximately 0.4%[7], makes the total area that human build and pave in the US less than 3.5%. It is hard to find this type of data by individual countries across the world, but we can can conservatively estimate that the total occupied areas cover around 1% surface area of the earth (via referencing the ratio in US). At street-scale, the occupancy ratio (total number of occupied cells over total number of cells) is also extremely low - of course, depending on the size of the cell. Figure 2.4 (a) shows the occupancy ratio of urban and highway scenarios, and their occupancy ratios are below 2% if the cell size (resolution) is around 20 centimeters.

[6]2010 Census Urban and Rural Classification and Urban Area Criteria, US Bureau of the Census:`shorturl.at/bzFGH`

[7]Functional System Lane-Length – 2013, US Department of Transportation:`shorturl.at/ltyPS`

Complementary to the above, while they are spatially sparse overall, the occupancy voxels are spatially clustered, as illustrated by the heat maps shown in Figure 2.5. The majority of objects fall on the surface (i.e., pavement road surface, curb, sidewalk, etc.) or are vertical attached to the surface (i.e., building, pole-like structure, tree, etc.). The point to nearest point distributions of urban and highway scenarios are shown in Figure 2.1, which illustrates how clustered the objects are.



(a)

(b)

Figure 2.5. Points/Voxels distribution along lane center line (longitudinal). We add up the voxels along road direction and project the them to the plane perpendiculars to road direction. a) 200-meter highway scenario: points distribute majorly on road surface and separator (i.e., guardrail and gap in this case), some points hit a vehicle that passes by and vehicles and road surface on the adjacent road. b) 150-meter urban scenario: points mainly locate on road surface and building facades, some points hit vehicles and trees nearby.

2.2.3. Characteristics of Semantic Object: Quantity, Density and Distribution

Estimating the quantity, density and distribution of each semantic object need a tremendous amount of labeling. Limit references/surveys/datasets with appropriate labels and small coverage can be found which can not reflect the true numbers, and these numbers vary strongly between regions and road function classes (e.g., highway, urban, highway near ramps, etc.). An example of pole distribution is illustrated in Figure 2.6. For instance, pole density ranges from $\sim$ 10 meters (longitudinally) in average in urban scenario [**20**], and $\sim$ 100 to $\sim$ 300 meters on highway [**224, 232**][8]. Meanwhile, signs are highly clustered near intersections in urban and ramps on highway.

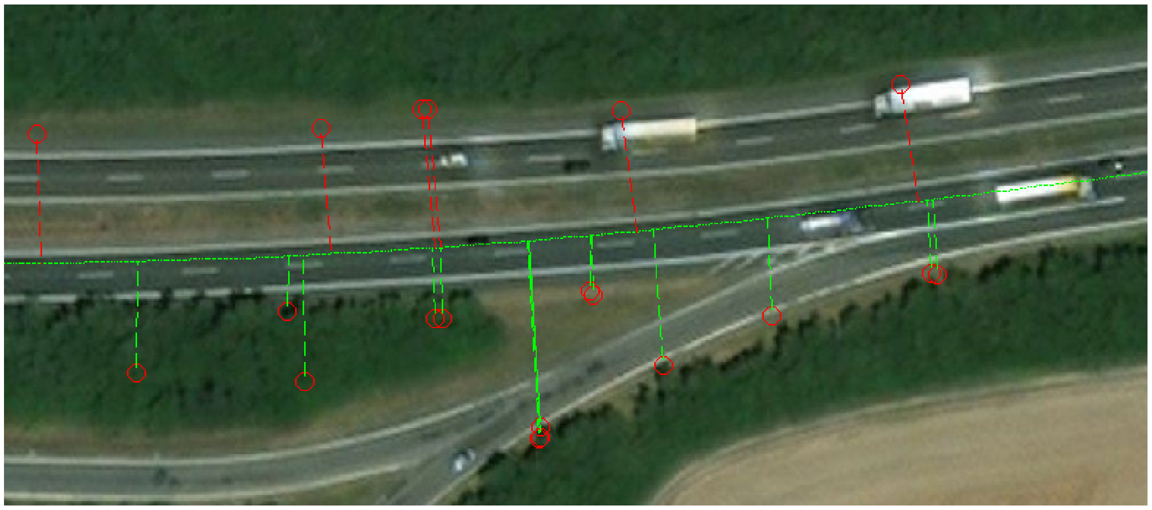


Figure 2.6. Illustration of the poles near an entrance ramp. Green line denotes a center line of a line, red dots denote pole-like objects, and green/red dashed lines denote the placement of poles (right/left) to the road.

2.2.4. Indexing

Tree structures are often being used to access the objects represented by points (or the polygons formed by points). In this section, we review several categories of trees

[8]The pole density range of highway varies between regions.

that are being widely used in similar applications, and discuss the potential of them being used in a vehicle localization system.

Octree and Point Region Octree: Octree - a three-dimensional analog developed from Quadtree [**2**] – along with the R-tree (for higher dimensions) are known as the most straightforward hierarchical data structures that provides an efficient access for 3D points (as well as line segments, polygons and volumes composed by these points).

These two data structures are widely used in geospatial applications, especially in maps, such as satellite imagery, POIs (e.g., buildings and attached attributes) and three-dimentional city model (e.g., building footprints and functional area contours). At the same time, they have different characteristics in terms of the basic operations (creation/indexing, updates (insertion/deletion) and query) because of their designs [**94**]. Octrees partition the space into eight congruent disjoint cubes (also known as octants or subdivisions) and recursively subdivide each octant until there are no more than a predefined-threshold of points located in each subdivision (leaf). A leaf can also represent the entire octant (not a certain point located in this octant), which is an equivalent to a voxel. The trees that maintain voxels only is known as Point Region (PR) Octree. In comparison, R-trees split multidimensional "objects" into multiple minimal bounding boxes (MBBs) to efficiently perform the filter step of a spatial search.

We observe that, on the most fundamental layer, as we mentioned in previous sections, street objects and the voxels corresponding to each object are sparse (cf. Figure 2.4) and clustered (cf. Figure 2.5). This could cause a highly unbalanced PR

Octree to be created when storing the data. By comparison, building a well balanced R-tree for each object (or for each category of objects) may be quite useful for localization scenarios (similar to 3D game rendering) – however, how to split points becomes a challenge and brings us uncertainty. Reliable points/objects segmentation methods are needed to achieve building a R-tree. Furthermore, R-tree is sensitive to noise (e.g. outlier points and error points). Single isolated outlier point will expand the MMB and increase layers of R-tree. We note that the PR Octree is easier and faster to be modified (insertion and deletion) [**94**] whereas in an R-tree, a small change may affect its higher level MBB. Last but not least, as a user-centered system, indexing and modification are considered as a one-time-cost that could be amortized over time, whereas query time and data size are more emphasized. In this comparison, R-trees show distinct advantages over Octrees in basic spatial searches (e.g., inside, contains, touch, cover, etc.) and index size of storing sparse data [**94**].
Variants: Modifiable Nested Octree and Potree: Researchers from different areas have faced very similar challenges: indexing large amount of points, and querying the datasets efficiently. The most relevant solution can be found mainly in the fields of online 3D visualization (of points, polygons, 3D maps, etc). This type of applications is required to render millions, or even billions of points and triangles in real time, while not causing memory crushes [**210**].

A well-known solution used in real time rendering problems is the, so called, Nested Octree and its variants such as Modifiable Nested Octree [**156**], as well as the more advanced Potree [**162**]. These approaches are indexing points and voxels in optimized octrees, with specially tuned query mechanism, so that large amount of objects can be rendered based on their locations (to the camera viewing frustum) with different level of details (determined by the distance to camera) to have an efficient and fine 3D object rendering. These octree variants are optimized for specific real time rendering issues. In our case, some of the problems that they are typically facing are weakened – while additional requirements needed to be tackled. For example, in rendering, the number of points of a still rendering is always much more that the number of occupied voxels. In the systems that are to deal with the vehicle self-localization problem, in contrast, much higher refresh rate is needed, because vehicle is moving at high speed.

2.3. Processing Pipeline Challenges

The end-to-end (from data acquisition to application) pipeline using 3D objects is relatively similar to most of feature-based applications, as shown in Figure 2.7, such as object detection and recognition, gps-based self-localization and path planning. In the case of 3D objects, the critical differential that be distinguished it from other systems is the data size is enormous and the data structure is complicated. The acquisition hardware/techniques and processing algorithms are well studied, such as using image sensor to reconstruct 3D information, or collect 3D points directly with LIDAR and radar sensor.

Applications such as vehicle 3D features/objects based self-localization [**209, 232**], lane-level/micro motion planing algorithms are being developing/improving in industry and academia. One thing that cannot afford be ignored is, all these applications are computationally expensive, which makes these applications sensitive to the input data size under limit hardware constrains. In other words, our focus is on how to efficiently use data from both sensor-end and database-end.

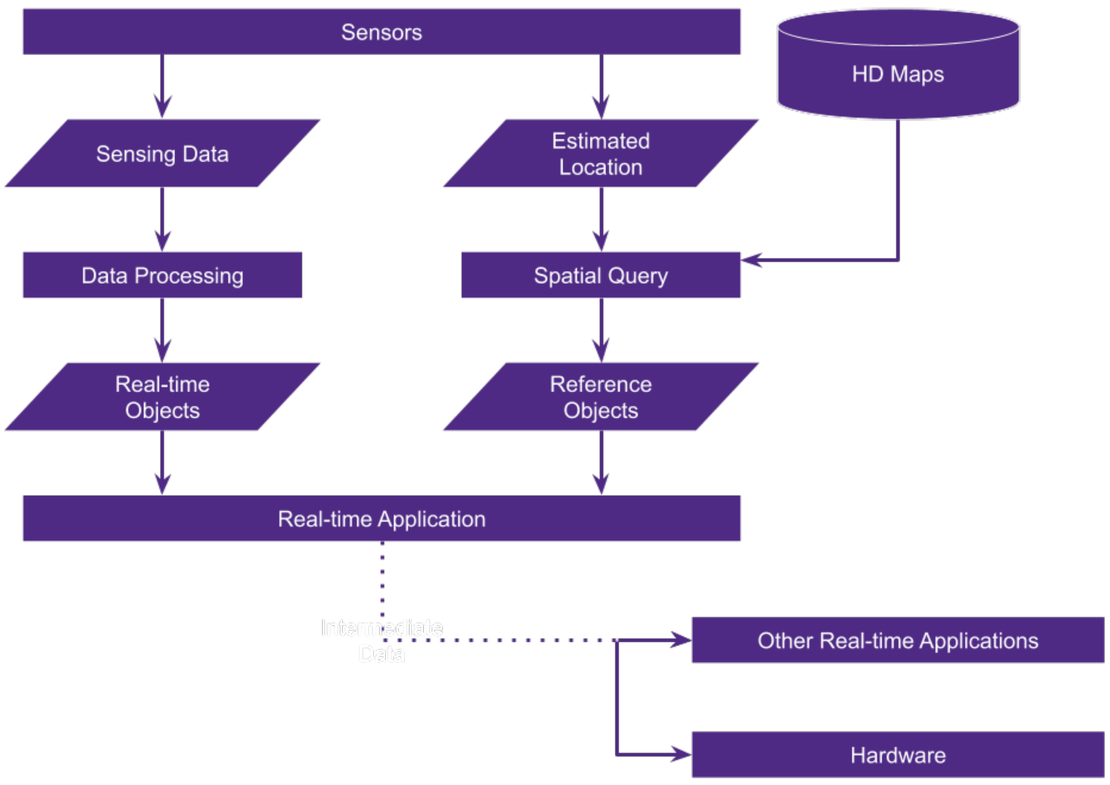


Figure 2.7. Illustration of a general pipeline using 3D object for in-vehicle real-time applications, from acquisition, through processing, database, to application layer.

The main challenges we address are the data flow from multiple sensors to cpu, and the data flow from databases (e.g., internal/local and external/cloud) to cpu.

And we name the first challenge as **Heterogeneous Data Fusion** and the second challenge as **3D Object Query**.

Heterogeneous Data Fusion Mobile unit equipped with different types of sensor to acquire data that needed. Each sensor has its own pros and cons in working environment, computational cost, memory cost, transmission cost, power consumption and sensing specifications (e.g., resolution, ranging, failure rate, etc.). Regarding system requirements of each platform – such as vehicle, mobile phone, and drone – for variant purposes – such as transportation, entertainment, survey, surveillance and agriculture – sensor design follows the rule of system redundancy to cover complicated (or even the most challenging) situations such as inclement weather and hardware failure, to achieve safety-critical, efficiency-critical, and accuracy-critical function. When all sensors work fully functional, the performance may overkill in daily driving scenario. Hence using the optimized combination of sensors with optimized configurations/parameters to satisfy dynamic system requirement is important.

For instance, image-based sensors have higher resolution and sensing range when acquiring 3D data, but the drawbacks at the same time are also obvious, they are relatively computational expensive, and easily be affected by poor lighting and inclement weather (e.g., rain and snow) [**232**]. Radar sensors are known as their low cost, low resolution and low sensing range but high efficiency, while LIDAR sensors have the advantages in high resolution, long sensing range but extremely manufacture cost. In different scenarios (e.g., urban, downtown, highway, etc.), weather condition, driving speed and safety index [**48**], system requirements keep changing with these external environments. The combination and configurations of sensors should be changed as the external environments, to optimize the performance with limit resource.

3D Object Query: Since the related applications are expensive and sensitive to input size, the highest priority task of the database should be return the all the necessary objects without as less redundant objects as possible. In Figure 2.8 (a), we have a real world example of urban scenario. Given a vehicle in red and it drives into an intersection, all the voxels/points (objects) locate within a certain distance near the vehicle are colorized. The color assigned to each pixel represents the distance to this vehicle. At this moment, the number of actual demand voxels is significant less than the total number of voxels in a regular spatial query, which means more objects we query and involve in computation are useless and waste a lot of memory and cpu usage.

Feasible optimizations can be found follow these state-of-the-art algorithms designed for other techniques:

(1) Using visibility to reduce the number of points. If we consider visibility problem [**215, 33**] in solving this task, only load the the visible objects in view frustum, the number of voxels that return from the server can be significantly reduced, especially in urban context [**231**], as shown in Figure 2.8 (b).

(2) Loading object with distance and sensor layout. Similar to the foveated rendering concept in computer graphics rendering [**66**] and online render optimization strategy [**162**], since all sensors have their certain angular resolution, there is no need to

load object in high resolution beyond a certain distance. An object resolution with distance visualization is shown in Figure 2.8 (c). If we use tree to store the voxels, this means the depth of some nodes when retrieving a tree can be reduced.

(3) Out-of-core optimization. When the vehicle drives along a certain path or randomly, we need to arrange the objects that not to waste memory I/O (e.g., from cloud database or local database). Correct objects need to be prioritised in appropriate order and wait to be resolved by cpu when needed (e.g., the vehicle moves to next location).

Map Update: Keeping the HD maps fresh is another hot topic in industry and academia. City and street keep changing all the time, from micro to macro. Macro change, for example, a public road reconstruction project. It is always easy to track such events, re-acquire data and update the map. The hardest challenge is the micro changes – for instance, how to detect (and report) small local changes and update new features (e.g., delete objects, add new object, or move objects) in database. Once map update is triggered, all the objects locate in updating area need to be pulled out and compared to newer collected data. This function requires the system itself is highly compressed.

Trade-offs: Since we need to use limit hardware and software resources to meet the system retirements and especially real-time criteria, the balancing between cost and overall performance is particular important.

Heterogeneous Data Fusion and Functional Performance Trade-offs The cost we spend on collecting and processing the data is a positive correlation between the functional " performance". For instance, in the self-localization task, the more and detailed surrounding 3D objects acquired by different sensors and different types of sensor, the better localization "performance" we get. But beyond a certain number and level of detail of the objects, the performance will overkill the system requirements, even considering the system redundancy. Finding the sweet spot of the heterogeneous data inputs and functional performance under dynamic external environments and requirements is an important trade-off to be studied.

3D Object Query Optimization Trade-offs As we mentioned in previous section, we may need to design several optimization approaches to reduce the number of objects that retried from the database, but the optimization also comes with a price. For example, solving the visibility problem is costly. Then the total performance of a 3D object query needs to be considered as a summation of the cost spend on optimization, and the cost on the application that runs on the reduced input.

2.4. Heterogeneous Data Fusion and Functional Performance

To quantify the performance - especially for feature-based (e.g., pole-like object, tree truck, guardrail, etc.) applications, similarly to Hazardous Misleading Objects (HMI) [**49**], a good indicator candidate is the F-score from object-level to voxel-level. A cost-to-F-score chart can clearly illustrate that the performance increases as the cost raises. An appropriate cost should meet the system minimum voxel-level or object level F-score.

3D Object Query: An important criterion to evaluate the performance is the query efficiency. As we described in previous section, efficiency of a real-time application

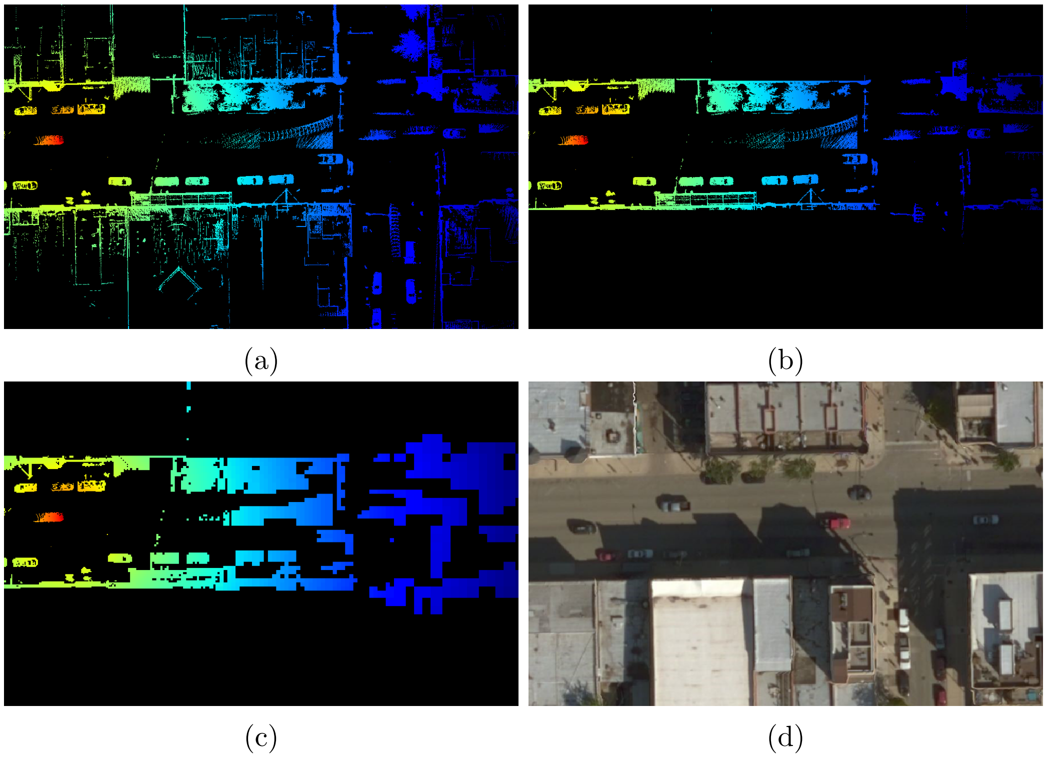


Figure 2.8. A vehicle (red) driving into an intersection with: (a) objects (represented by voxels in HD maps) colorized in their distances to vehicle, (b) visible objects colorized in distance, (c) objects resolution decreased with their distances to vehicle, and (d) a satellite image for reference. The number of voxels is reduced from (a) to (b) to (c). Ideally (every object is needed), the *Simple Stationary Problem* performances in *3D Object Query* challenge are all 100% with $77,736$, $44,468$, and $5,213$ voxels.

using high resolution 3D objects in HD maps is highly correlated to the number of objects/voxels retrieved from the database, so the objective of **3D Object Query** is to both return the correct objects and try to reduce the "useless" objects, and this will also reflect in the run-time performance.

To simplify the problem, we divide the **3D Object Query** problem into three progressive contexts – Stationary Case, Simple Dynamic Case and Dynamic Case – and define their evaluations:

(1) **Stationary Problem**. The simplest case of the system faces is, assume the vehicle is at a certain location, and the vehicle needs to load its surrounding objects for the related applications, how to load these objects efficiently.

Hence the fundamental problem can be defined as follow: assume the vehicle acquires points $P_v = \{p_{v1}, p_{v2}, ...p_{N_v}\}$ at each cycle, given vehicle estimated location $L = \{l_{lat}, l_{lon}, l_{alt}\}$, return the $P_m = \{p_{m1}, p_{m2}, ...p_{N_m}\}$ that surround the vehicle (for localization purpose), try to optimize the $P_v\ IoU\ P_m$ (IoU: Intersection over Union, in voxel representation) - to be accurate, optimize the pair of $\frac{P_v \cap P_m}{P_v}$ and $\frac{P_v \cap P_m}{P_m}$. Theoretically, a $\frac{P_v \cap P_m}{P_v} = 1$ is sufficient and necessary to safe driving (all needed objects/voxels are loaded), while a smaller $\frac{P_v \cap P_m}{P_m}$ is better. In reality, moving and random objects getting involved, a $\frac{P_v \cap P_m}{P_v} = 1$ condition can not be achieved, but in the experiment, the baseline is to satisfy $\frac{P_v \cap P_m}{P_v} = 1$.

A quick example is given in Figure 2.8. Figure 2.8 (a), (b), and (c) illustrate three different query strategies: all voxels, voxels with visibility model and voxel with visibility model and angular resolution respectively. The $\frac{P_v \cap P_m}{P_v} = 1$ for all three strategies, but the $\frac{P_v \cap P_m}{P_m}$ equals to 1491.1%, 853.0% and 100% respectively. The obvious conclusion is returning the voxels with visibility model and considering angular resolution is significant more efficient than a simple spatial query. Also notice that, every time when a system initializes the application, the first position returned by GPS always has the largest error, especially in urban area. Then the system needs to query objects in a larger range to ensure $\frac{P_v \cap P_m}{P_v} = 1$ at this moment. With the system running more iterations, the GPS errors reduces, and a tier range can be applied.

(2) **Simple Dynamic Problem**. An advanced case can be described as if the vehicle moves along a preset route, and the system memory and cpu are not sufficient to load and process all the objects along the entire route – which equivalents to they can only load and process a portion of the route – what is the most efficient strategy to prioritize the order of objects to be loaded, which saves the memory (with transmission bandwidth, local storage, etc. if needed). and run-time.

Adding the vehicle trajectory (one planned route) composed by continuous locations $S = \{s_1, s_2, ..., s_n\}$, assume we have a limit memory space - for instance, we can only maintains N_m tiles (assume we use "tile" as the basic container of voxels) in memory at every moment. N_m tiles are sufficient to handle the surround points at one location, but not enough for the entire route. Given T_{s_i} denotes the tiles set at location s_i, where $i \in [1, n]$. To keep the pipeline running efficiently, $\|T_{s_i}\| = N_m$. Let $T^+_{s_i} = T_{s_{i+1}} - T_{s_i}$ and $T^-_{s_i} = T_{s_i} - T_{s_{i+1}}$ denotes the tiles push into the memory and pop out of the memory respectively. The evaluation can be formalized as $COST(S) = \sum_{i=1}^{n}(\|T^+_{s_i}\| + \|T^-_{s_i}\|)$, while an efficient pipeline should minimize the $COST(S)$.

(3) **Dynamic Problem with Probability**. In reality, the scenario is much complicated, the route planning is dynamic because of either driver real-time decision or

real-time route planing algorithm. Hence a mature query strategy should be able to handle this uncertainty.

Assume $G = (V, E, W)$ is a directed acyclic graph that represents the road network, where $V = \{v\}$, $E = \{e\}$, and $W = \{w\}$ denotes every moves (e.g., change lane or at an intersection), links between each adjacent moves, and the vehicle decision probability $w_{i,j} = P(v_i|v_j)$ on the link from two adjacent moves v_i to v_j. Let $S = \{v_1, v_2, ..., v_n\} = Path(v_1, v_n)$ denotes a path from location v_1 to v_n, and the probability of this vehicle chooses this path is $Prob(S) = \prod_{i=1}^{n-1} w_{i,i+1}$ and the cost is $\sum_{i=1}^{n}(\|T_{v_i}^{+}\| + \|T_{c_i}^{-}\|)$. Since the motion planing is dynamic, we have $\{S\}$ paths that can navigate the vehicle goes from v_1 to v_n, hence the dynamic cost can be represented as $\{COST(S)Prob(S)|S \in \{S\}\} = \sum_{i=1}^{n}(\|T_{v_i}^{+}\| + \|T_{v_i}^{-}\|)w_{i-1,i}$, where $v_i \in Path(v_1, v_n)$ and $w_{0,1} = 1$.

In some cases such as at an intersection, the query strategy is highly determined by the selection of next link. If the probably of each link is similar (low variance), then at this location $v_{intersection}$, the $COST(v_{intersection}) = \|T_{v_{intersection}}^{+}\| + \|T_{v_{intersection}}^{-}\|$ will become larger and cost a lot of memory. Machine learning may get involve here to reduce the variance of the probabilities of all possible links, that can reduce the $COST(v_{intersection})$ to save memory usage.

Notice that no matter which optimization approach is applied, the overall performance of the application is still considered as the end-to-end run-time, which includes optimization time and application itself, and this part relates to the **Optimization Trade-off** challenge. Only if the optimization we use is significantly cheaper than the application algorithm itself – for instance, the application costs n^2 while optimization costs $n \log n$ – we can use the **3D Object Query** performance to determine the efficiency directly.

CHAPTER 3

Lane Boundary Geometry Extraction from Satellite Imagery

3.1. Background

Recently, a lot of work has been done to automate lane-level map generation using vehicle-sensor meta data crowd-sourced from large fleet of vehicles [**32**] in addition to ground level data such as imagery [**30, 31, 28**], LiDAR [**236**], GPS, and Inertial Measurement Unit(IMU) collected by mobile mapping vehicles.

Extracting road/lane information from airborne imagery has its advantage over terrestrial data due to its comprehensive coverage, low-cost, ease to update. The history of road extraction from orthogonal imagery (e.g. satellite and aerial) can be traced back to more than forty years ago; however, limited by image resolution (typically over 2 meters per pixel), traditional approaches rely on edge detection, color segmentation, linear feature detection, and topological linking [**196, 6**] to extract road networks from overhead imagery. In recent years, more machine-learning based approaches are proposed to detect patch/pixel-wise road region [**123, 73, 25, 124, 76**]. These road centric approaches still cannot model the lane-level features even though the common satellite image resolution has improved to 0.5 meter per pixel.

Now, satellite imagery can have a resolution of 0.5 meter per pixel or less, which allows us to utilize the classic approaches with much detailed imagery to model lane-level features [**119, 120, 168, 143**]. There are still two challenges after the lane boundary line is successfully detected: the representation of the lane model and the evaluation metric of model accuracy and performance. In paper [**143**], the road model is represented as a collection of unstructured lines without attributes; while in paper [**168**], the definition of accuracy is based on the percentage of pixel-wise overlap comparing to their manually drawn line masks in the input images. Hence, their claimed accuracy is less persuasive.

Autonomous vehicles are becoming more of a reality. The increasing demand of HD mapping can be predicted, especially for interstate transportation (i.e. autonomous truck [**55**]. The three largest highway networks in the world, U.S., China, and India, are 103, 446, and 79 thousand kilometers [**1, 185**] long, respectively, which motivates us to concentrate on highway-level road network in this dissertation. We propose a novel, automated lane boundary extraction technique from satellite imagery. Our approach consists of two stages: pixel-wise line segmentation and hypotheses-grouping classification linking. The pixel-wise line segmentation approach contains patch-based lane-marking classification, and for each positive patch, we segment line pixels to generate line candidates. Hypotheses-linking connects each line candidate by minimizing the proposed cost function to generate structured lane model. A formalized road-model-accuracy-metric is designed to evaluate the results rigorously. We also manually extracted lane boundary ground truth from our dataset.

Along with satellite imagery, it can be used for training, testing, and evaluation for comparative studies.

3.2. Methodology

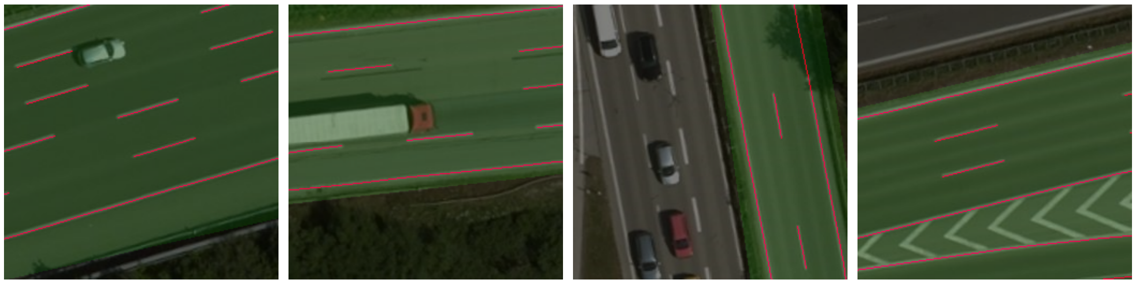
Our lane-boundary-geometry extraction approach contains two stages. In training, similar to [**65**], we use the ground truth lane boundary geometry and the corresponding satellite imagery from Bing Tile Server as input, project lane boundary lines to imagery, crop image into small patches, and train our patch level classifier. In extraction, our approach uses pre-trained classifier, target route/trajectory, and corresponding satellite imagery as input, detects patch-level lane marking candidates [**35**], segment the pixel-wise lane marking candidates, and links [**168**] the pixel-wise candidates to generate lane boundary geometry.

3.2.1. Patch and Patch Level Classification

The objective of this step is to build a classifier that can determine whether an image patch contains any lane marking pixel. Even though the ground truth road model is organized in chunk-wise structure [1], due to the specificity of the tile system, generating training patches in chunks unnecessarily queries the image server twice (each tile always contains two to three chunks at high resolution level). Hence, our solution is designed as tile-wise - for each tile along trajectory, project all control points of each functional line and connect them. To reduce noise (i.e. lane marking pixel of adjacent road surfaces), the surface region is bounded by road boundaries

[1]Divide road along centerline into pieces evenly

(given in the ground truth dataset). Samples contain lane marking pixels in red and road surface region in green are illustrated in Figure 3.1.



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

Figure 3.1. Satellite tile images fused with ground truth lane boundary geometry at tile pixel $[557869, 363521]$, $[557909, 363513]$, $[558036, 363507]$ and $[557879, 363518]$ at level 20 from left to right. Road region highlighted in green, bounded by road boundaries. Lane marking highlighted in red.

A sliding window is designed to crop training patches from corresponding satellite image within the road surface. The label for each patch is determined by whether there is any lane marking pixel hit in the current patch. To reduce misleading ground truth patches (the patch contains two independent lines), an appropriate window size should be thinner than lane width in real scale, which is 3.5 meters [2]. In this project, the ground resolution is approximately 0.15 meter per pixel at tile level 20, which means the patch size should be less than 24. Examples of positive (contain lane marking pixel) and negative patches are shown in Figure 3.2.

[2]New Guidelines for the Geometric Design of Rural Roads in Germany

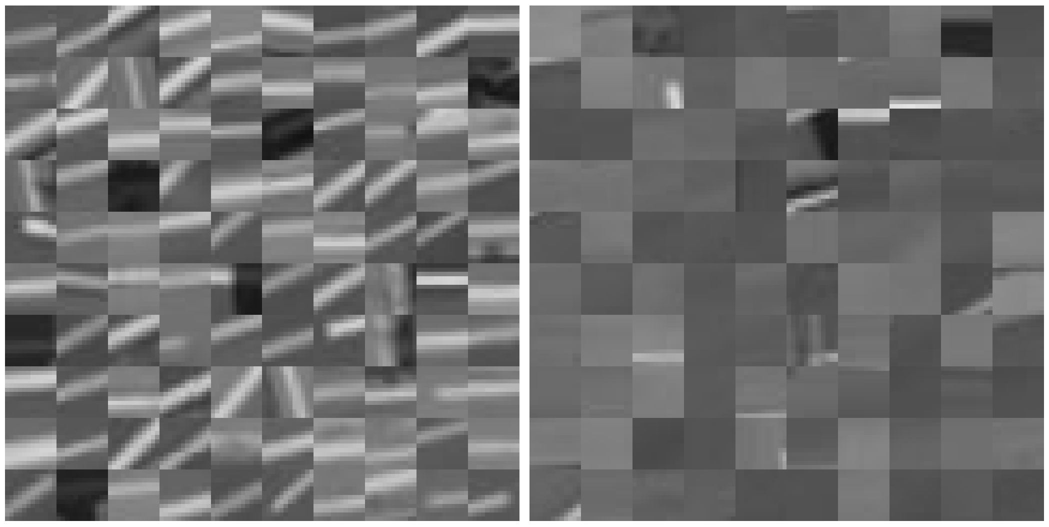


Figure 3.2. 100 sample patches of original orientation lane-marking patches (a) and non-lane-marking patches (b).

Given a satellite tile image (shown in Figure 3.3 (a)), its corresponding probability map of patch level lane marking with certain configuration (patch size is 12 pixels, use pixel representation feature and Random Forest classifier) is illustrated in Figure 3.3 (b).

3.2.2. Pixel-wise Lane Marking Segmentation

Our patch level classification returns a probability map as output, which contains high probability lane marking regions like the red area shown in Figure 3.3(b). However, since each region is always several pixels wide (depending on patch size and step length), this wide range obviously could not meet the requirement considering the definition of HD Maps [**15**].

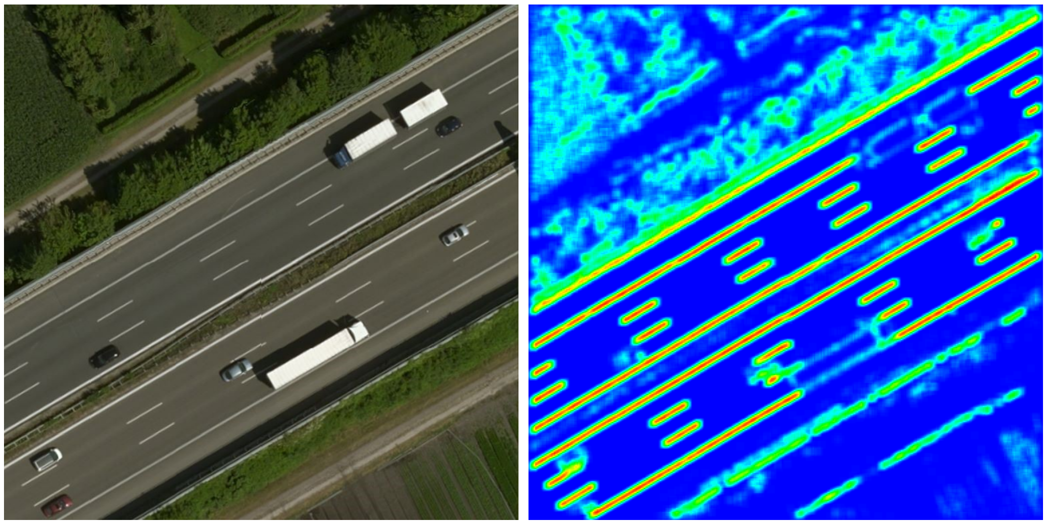


Figure 3.3. Original satellite tile image (a) and its patch level lane marking probability map (b) at location $48.2203°, 11.5126°$.

To segment and locate precise lane marking pixels, we consider pixels with the highest intensity in each slice of lane marking region perpendicular to road trajectory as lane-marking candidate. Then, we fit a line segment through the lane marking pixel candidates. For example, in Figure 3.4, assuming the trajectory is **up**, for each row of this region, the highest intensity points are 151, 154, and 150, respectively, which means the lane marking line segment should be the centerline of this region.

Even though the satellite image resolution has already matched the lane marking width, limited by image compression, hardware constraints(lens, CMOS), and optical limitations (i.e. angular resolution), tiny/thin object will always be blurred at its boundaries when captured. Furthermore, to segment more precise lane marking pixel locations, we introduce sub pixel-wise segmentation. For each slice of the lane marking region, fit a Gaussian model (green lines in Figure 3.4) and find the peak of each model (yellow circles in Figure 3.4). Then, the lane marking pixel location becomes sub-pixel-wise instead of the naive pixel-wise of each slice of the lane marking region. Theoretically, line accuracy can be improved by at most half a pixel. The pixel-wise lane marking segmentation result is illustrated in Figure 3.5.

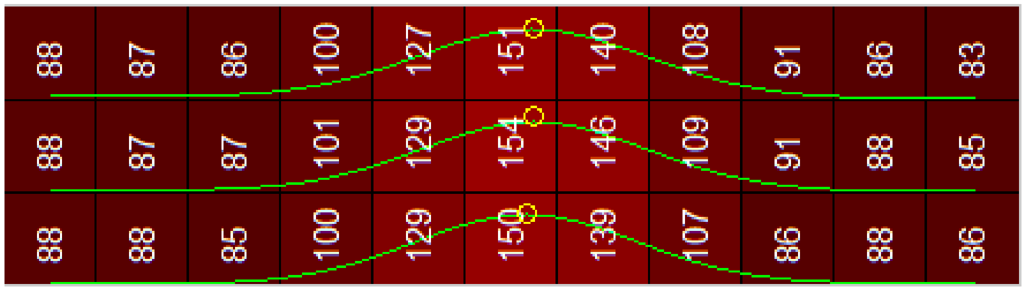
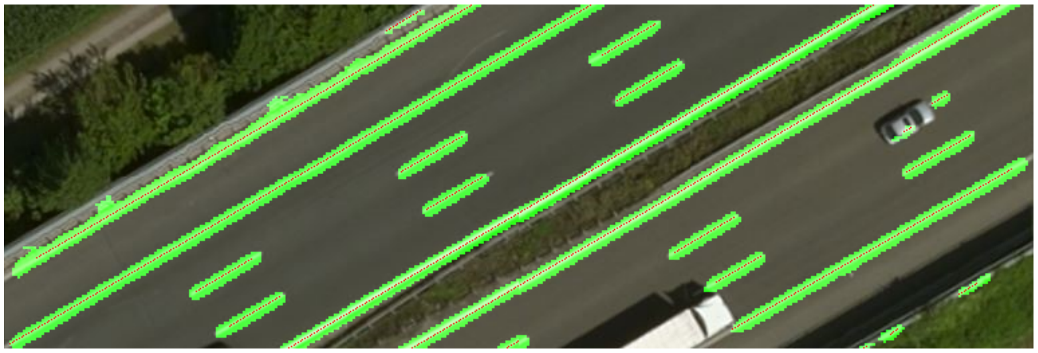


Figure 3.4. Pixel-wise and sub pixel-wise lane marking segmentation visualization, each number inside pixel represents its intensity value converted from the raw RGB image.

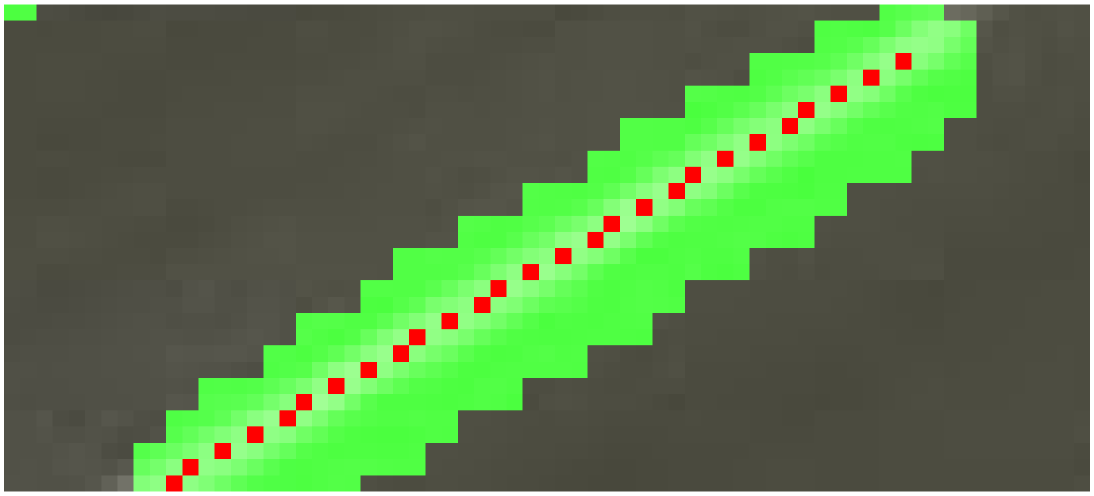
3.2.3. Line Candidates Grouping, Classification and Linking

The previous step generates unstructured line segments without relative position and function label (solid/dashed line). Because of occlusion (i.e. trees, vehicles, buildings, and their shadows) and poorly painted lane markings(examples shown in Figure 3.6 (a)), less true lane marking lines will be detected, while more misleading lines (false positive) will be detected if lane-marking-like objects appear (i.e. guardrail, curb, wall shown in Figure 3.6 (b)).

The method of of transforming the unstructured lines to structured lines with function labels contains three steps: grouping line candidates from each chunk, classify the function of each line group, and link the missing lines.



(a)



(b)

Figure 3.5. Lane marking region candidates (green regions) and pixel-wise lane marking pixel candidates (red dots) overview (top) and zoom view (bottom).

In the grouping step, whether or not to push a line into a group depends on the relative distance [3] between the current line to all other line candidates in the current chunk, in the neighboring chunk(s), and their relative distances to road centerline/vehicle trajectory. For example, line candidates (gray) from four continuous

[3]There is no definition of line segment to line segment distance in a 2-D plane if they are not parallel, the relative distance here is the average distance between each point from one line segment to the other.

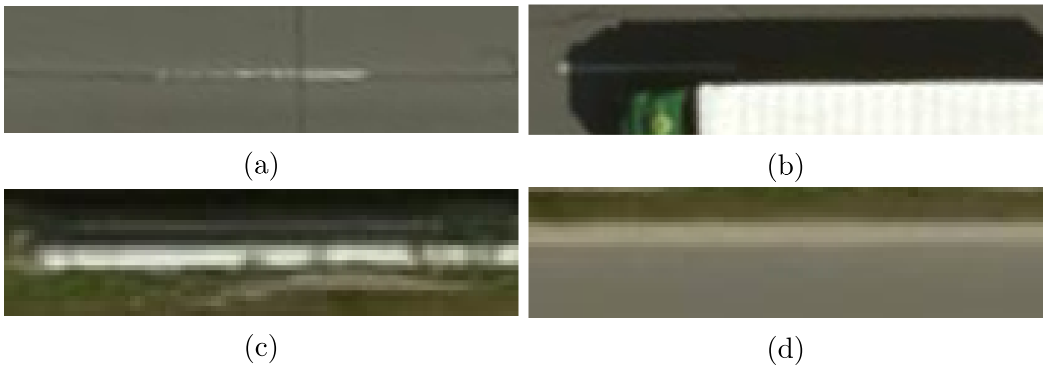
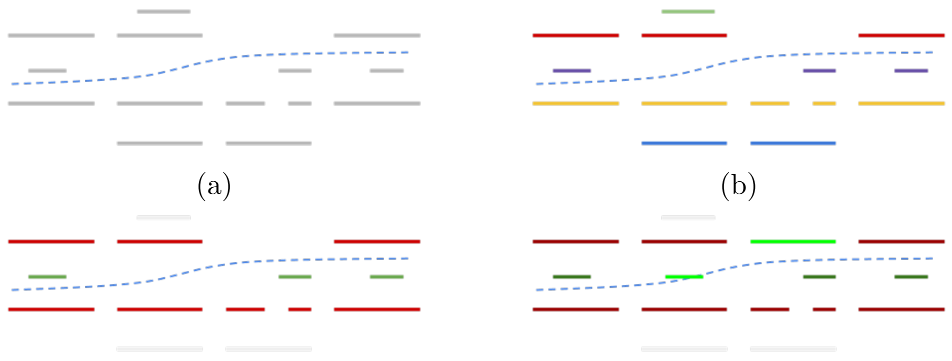


Figure 3.6. Phenomenons cause mis-detection: bad painting quality and shadow (a), misleading objects that cause false positive: high reflective metal guardrail and cement curb (b).

chunks and vehicle trajectory (blue dashed) are illustrated in Figure 3.7 (a). After the grouping step, five groups are generated and colored in Figure 3.7 (b). On a certain portion of the road, for each group, the function label is determined by the ratio of the total length of detected line segments belong to this group, to the total length of road contains this line group. Typically, consider reasonable mis-detection and wrong detection, the length ratio of dashed line is below 40% and the ratio of solid line is above 80%. In the task of modeling highway roads, there is an assumption that each road portion can have at most two solid lines bounding the (drivable) road surface. Figure 3.7 (c) illustrates the groups after the classification step, solid lines and dashed lines are colored in dark red and dark green, respectively, lines out of solid lines (drivable road surface) are colored in gray and will be ignored. In the final step, if one chunk does not contain a line that belongs to the group which passes this chunk, a synthetic line will be interpolated (light green lines shown in Figure 3.7 (d)).



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

Figure 3.7. Illustrations of Line Candidates grouping, classification and linking steps.

Notice that in this grouping, classification, and linking procedure, numerous thresholds and constrains (i.e. distance threshold, search range, etc.) are needed to control the process. Generally speaking, if we abstract all these variables into one degree: loose (longer search range, wider distance threshold) and tight (shorter search range, narrower distance threshold) to reflect the abstract performance of the model, the tightness-to-performance chart is illustrated in Figure 3.8[4]. As we can see, it is a trade off between function accuracy and geometry accuracy [5].

Furthermore, if any constraint (additional information) is provided when extracting the lane boundary geometry - for example, if we know the number of lanes for a certain portion of the road - this process can be fine tuned to generate a much better result.

[4]This chart is subject to mis-detection rate and wrong detection rate for the portion of road
[5]The definitions of function and geometry are described in Section 3.4.1

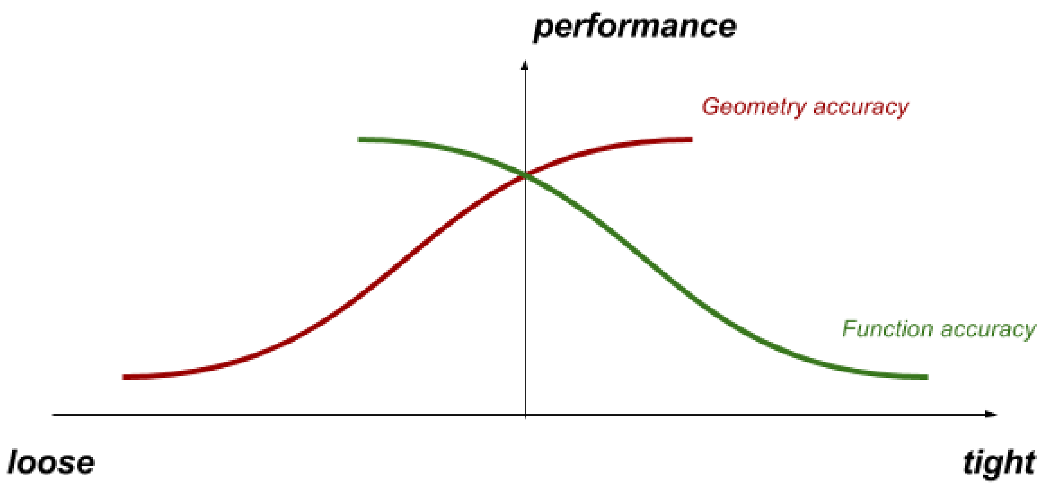


Figure 3.8. Constrains tightness to modeling performance chart.

3.3. Lane Boundary Ground Truth Collection

In paper [**168**], the author uses manually-drew, pixel-based ground truth, represented in 'mask' format [**167**], to evaluate his accuracy. The author does not have detailed description and statistic of his dataset. The number of ground truth masks is 50, which also limits its persuasion.

To code our lane boundary geometry dataset, we built an interactive tool which allows us to manually draw lane boundaries from scratch and present background image from various sources (i.e. point cloud projection, satellite imagery, etc.). The user interface is shown in Figure 3.10. An lane boundary geometry extraction from LiDAR point cloud pipeline will be executed at the very beginning to generate near-perfect lane boundaries to boost our modeling efficiency from 29.2 meters per minute to 12.8 meters per minute[6]. Then we use our tool to edit (delete, move, insert, etc.) the control points on lane boundary lines to make align them with the background imagery perfectly.

In this section, we are going to discuss the road selection, coding rules, lane boundary geometry representation, and potential system errors. More lane boundary geometry data of diversified scenarios (luminance condition, country, etc.) will be published as future work.

3.3.1. Lane Boundary Data Description

We collect lane boundary geometry on Germany Autobahn A99 (Bundesautobahn 99), from location 48.2206°11.5153° to 48.2057°11.4586°, divided into seven portions (five for training and two for testing) to exclude overpass structures and other unexpected scenarios.

This dataset contains training data (approximately 10.14 kilometers) and testing data (6.07 kilometers), which follows similar coding rules but two key differences in coding the dashed line and the coordinate system due to the particularity of this research. In training set (coverage shown in Figure 3.9, highlighted in red), dashed lines are represented as isolated line segments (two end points), shown in Figure 3.10(b), and aligned with satellite imagery coordinates. In testing dataset (coverage shown in Figure 3.9, highlighted in yellow), dashed lane markings belong to one set of continuous/sequential control points if they are semantically treated as

[6]Time efficiency is dependent on the number of lanes and the road structure. In our dataset, the majority of lane number is 3.

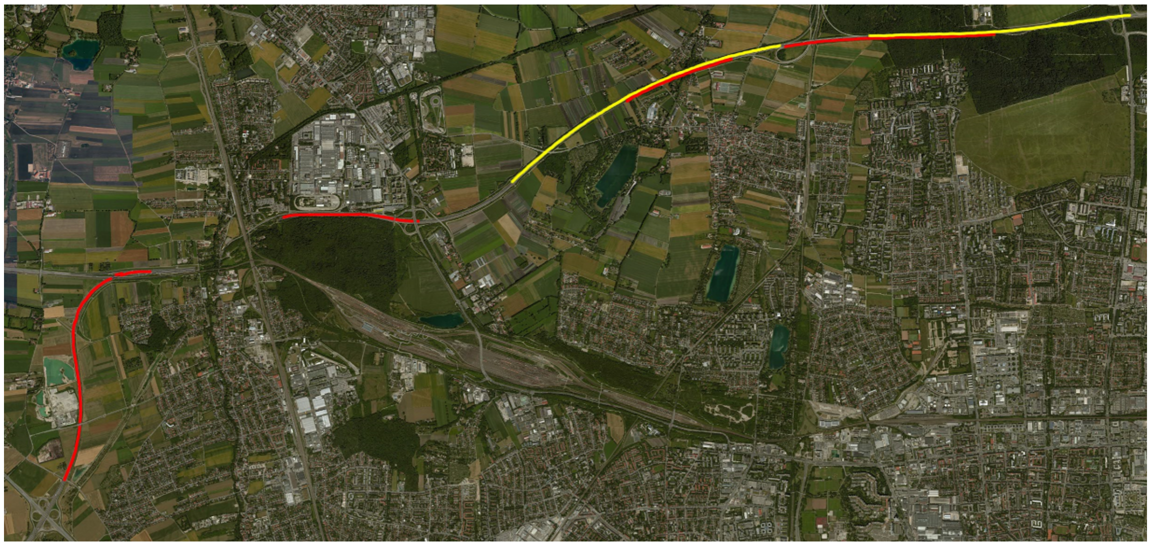


Figure 3.9. Illustration of our ground truth portions highlighted on aerial imagery, red for training and yellow for testing.

one line and aligned with point cloud, as shown in Figure 3.10(c). All control points are placed right in the middle of their corresponding lane marking.

Also, in this dataset, we code road boundaries (i.e. guardrail, curb) and use them to separate road surface from non-road surface (to exclude lane-marking like objects outside the road surface).

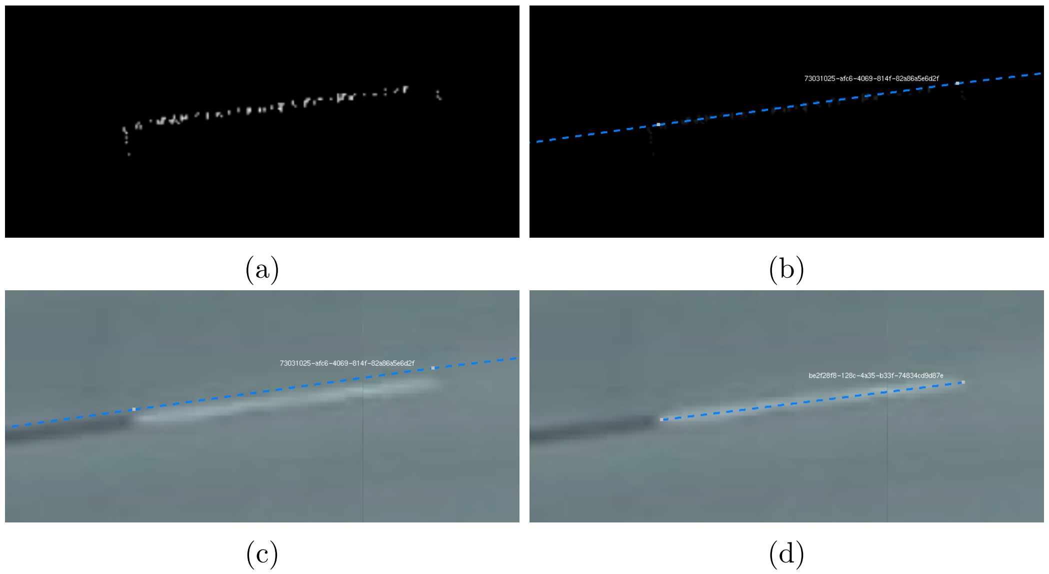

73031025-afc6-4069-814f-82a86a5e6d2f
(a)
(b)
73031025-afc6-4069-814f-82a86a5e6d2f
be2f28f8-128c-4a35-b33f-74834cd9d87e
(c)
(d)

Figure 3.10. Lane boundary line in point cloud view (a), test lane boundary line (continuous dashed line) in point cloud view (b), test lane boundary line in satellite imagery mode (misalignment between two systems can be obtained)(c) and train lane boundary line (isolated dashed line) in satellite imagery view (d). UUID denotes the line id.

3.3.2. Data Annotation

To represent lane boundary geometry and lane boundary lines for user convenience, lane boundary lines are evenly divided into 12-meter chunks. Each chunk is wrapped up in a single JSON file that follows this structure[7]:

[7]Map version specifies the map version which ground truth lane boundary lines aligns to, follows Bing Maps URL http://a0.ortho.tiles.virtualearth.net/tiles/a/[quadkey].jpeg?g=[map version]

```
Chunk JSON file
  id: INT
  map version: INT
  lines:
    line id: STRING: UUID
    type: STRING: ['solid'/'dashed'/'trajectory']
    points: FLOAT: [n by (latitude, longitude)] matrix
```

3.3.3. Errors in latitude, longitude, and altitude

Aerial imagery and point cloud are stored/represented in two coordinate systems - Mercator projection coordinate system [**14**] and Cartesian coordinate system [**4, 163**] - because of their acquisition techniques. Aerial image tile system is designed by demand years ago, but it inevitably has heavy distortion.

To avoid distortion, point cloud processing procedures and our labeling tool are designed to process data in Cartesian coordinate system. The different coordinate systems lead to a slight distance error when coded on these two layers - point cloud representation in a Cartesian local tangent plane: North-East-Up (NEU) and imagery represented in Mercator projection. Assume location $[\phi, \lambda, 0]$ at zoom level l, the Euclidean distance $d_p(\phi, \lambda, d_x, d_y)$ between the points back-projected through Mercator projection and Cartesian coordinate transformations of pixel shift d_x, d_y is complicated.

After simplification, d_p can be represented as the function of latitude ϕ, pixel shift $[d_x, dy]$, and zoom level l, and the illustration of function $d_p(\phi, d_x, d_y, l)$ at certain zoom level $l = 20$ and certain $dy = 0$ is shown in Figure 3.11. According to precision requirements from most 'HD' definitions [**74, 191, 188**], the error caused by fusion of two coordinate systems (less than 5 cm all around the world at tile level 20) does not have an impact on the final accuracy.

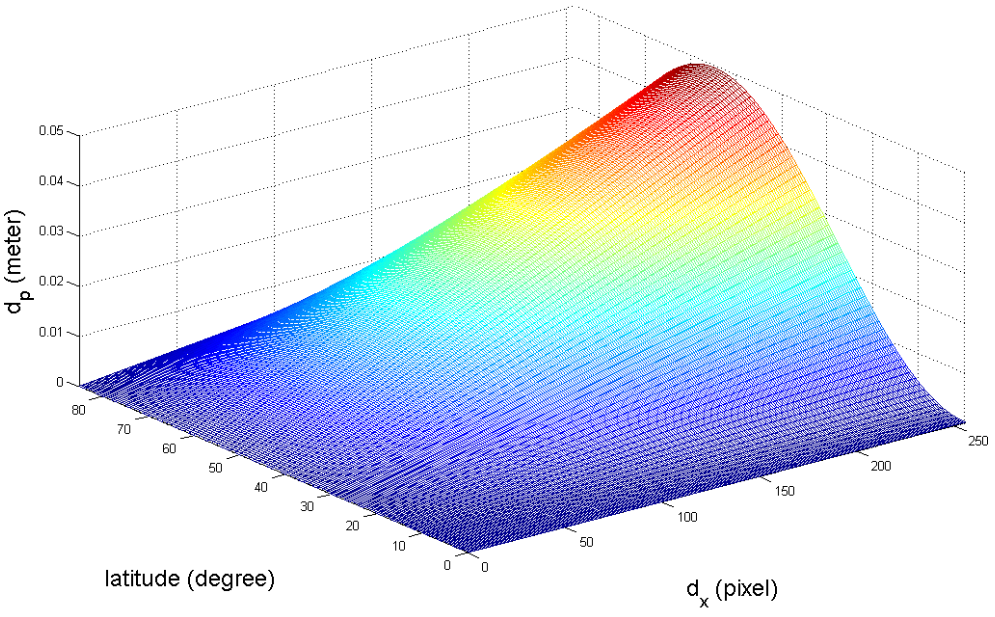


Figure 3.11. Illustration of $d_p(\phi, d_x, d_y, l)$ when $l = 20$ and $d_y = 0$.

3.4. Experiments

In the proposed methodology, we test numerous patch configurations (patch feature, size) and machine learning techniques to find the appropriate classifier and run our end-to-end program to extract lane boundary geometry using this optimum classifier. In this section, we are going to present both patch level accuracy and final extracted model accuracy by using the metrics described in the next sub-section.

3.4.1. Accuracy Definition

Lane boundaries is a collection of numeric and functional polylines/splines in control point representation [**158**], we cannot simply report the accuracy in pixel-wise representation as [**168**] proposes.

Considering the features of lane boundary and the misalignment between ground truth and modeling coordinates, we propose two metrics for a persuasive performance score: **function level** and **geometry level** accuracy. Theoretically, the misalignment between two coordinates causes transformation from one model to another to shift, scale, rotate, and even skew. In our task, scaling, rotating, and skewness are unnoticeable so they can be ignored, only shift transformation will be considered.

Given ground truth lines $L_i = \{l_{i,1}, l_{i,2}, ..., l_{i,n}\}$ and predicted lines $L'_i = \{l'_{i,1}, l'_{i,2}, ..., l'_{i,m}\}$ from ith chunk, the first step is to match $l \in L_i$ and $l' \in L'_i$. Let $d(l, l')$ denotes the distance between pair lines l and l' with sign (for example, left for negative) and $Pair(l_i, L'_i)$ denotes the paired line in L'_i of l_i. Align two models by using each l_i and each l'_i, find the

$$\text{argmin}_{all\,pairs}\left(\sum_{l_{i,j} \in L_i} (d(l_{i,j}, Pair(l_{i,j}, L'_i)))\right) \tag{3.1}$$

Figure 3.12 shows two pairs: green dashed arrow and red dashed arrow. By comparing the total length of the green and red solid arrows, the best match is green dashed arrow. Given distance threshold T_d from requirement (for example, HD Maps), a correct detection of l'_i and l_i is counted if their functions are matched and $d(l'_i, l_i) < T_d$. Then, the accuracy of the predicted model compared to ground truth can be represented in

$$precision_{\text{function}} = \frac{number\ of\ correct\ detections}{(\|\mathbf{L}\|)} \tag{3.2}$$

$$recall_{\text{function}} = \frac{number\ of\ correct\ detections}{(\|\mathbf{L}'\|)} \tag{3.3}$$

$$\mathbf{L} = \{L_i\}, \mathbf{L}' = \{L'_i\}, i \in road \tag{3.4}$$

To calculate geometry accuracy, for each pair $\{l_{i,j} : Pair(l_{i,j}, L'_i)\}$ in L_i, the *shift* is defined as $AVG_i(d(l_i : Pair(l_{i,j}, L'_i))), L'_i\ and\ L_i \in \mathbf{L}$, and the $performance_{geometry}$ is defined as $AVG_i(1 - \frac{\sigma(d(l_i:Pair(l_{i,j},L'_i)))}{\sigma_{max}(\|L_i\|)})$, while $\sigma_{max} = \sigma(L_{max})$ is used to normalized precision for each chunk, where $L_{max} = M \cup N$, set M contains *floor*$(\frac{\|L_i\|)}{2})$'s T_d and set N contains *ceiling*$(\frac{\|L_i\|)}{2})$'s $-T_d$ (for example, if $\| L_i \|)$ equals 4, then $\sigma_{max}(\| L_i \|) = \sigma([T_d, T_d, -T_d, -T_d]))$. With this $\{perfomance : shift\}$ metric, we can present the lane boundary geometry reasonably if the alignment between two source coordinates is unknown. Also, we can tweak the parameters of end-to-end program to generate expected model depends on the project requirement. For example, if alignment is not a requirement and we want to address the LiDAR shadow issue from point cloud, we would need to tweak the configurations with the lowest *performance*.

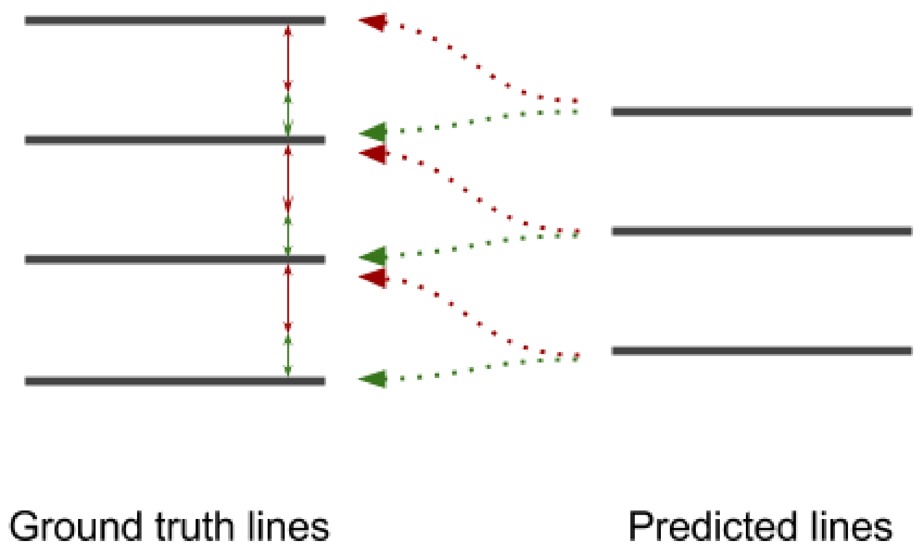


Figure 3.12. Ground truth lane boundary lines (left) and predicted lane boundary lines (road) pairing.

3.4.2. Patch level accuracy

To find the best patch level classifier, we crop tile image with given ground truth data as described in Section 3.2.1, with configurable variables such as as patch size $(8, 12, 16, 24)$, patch feature (pixel representation and gradient based features such as Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP)), and numerous machine learning techniques (Random Forest (RF), Support Vector Machine (SVM), Artificial Neural Network (ANN) and Convolutional Neural Network (CNN)) to evaluate their precision and recall. Figure 3.13 shows the performances of different training configurations of 10-fold Cross Validation.

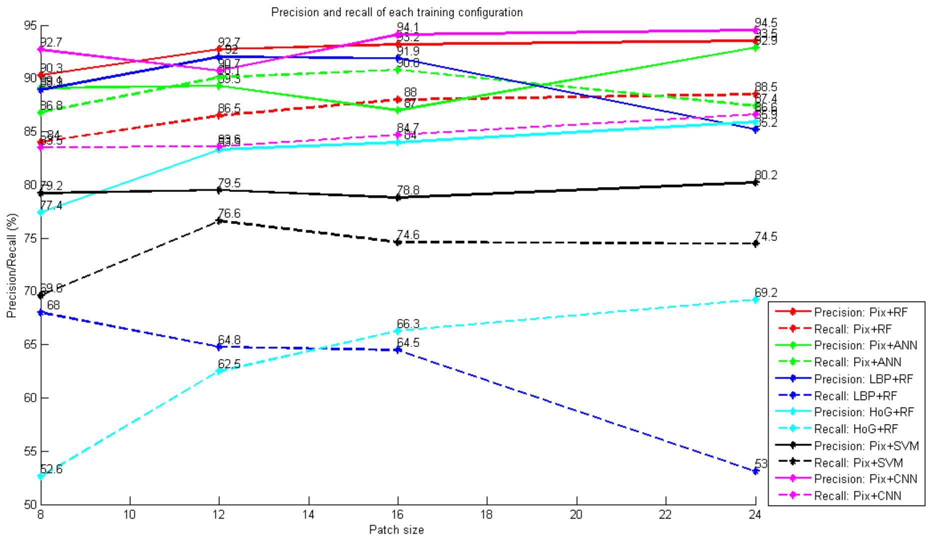


Figure 3.13. Patch level precision and recall of each training configuration.

According to our evaluation result, the change in patch size does not have a big impact on patch level performance. Considering the computational cost, a patch size of 12 is used in our final patch level classifier to generate a dense/smooth lane marking probability map shown in Figure 3.3 (b).

3.4.3. Lane Boundary Geometry Accuracy

With the pre-trained classifier and our end-to-end solution, we tweaked parameters and thresholds as mentioned in Section 3.2.3 to evaluate the performance of our approach on the testing set. The function level and geometry performances are shown in Table 3.1 below.

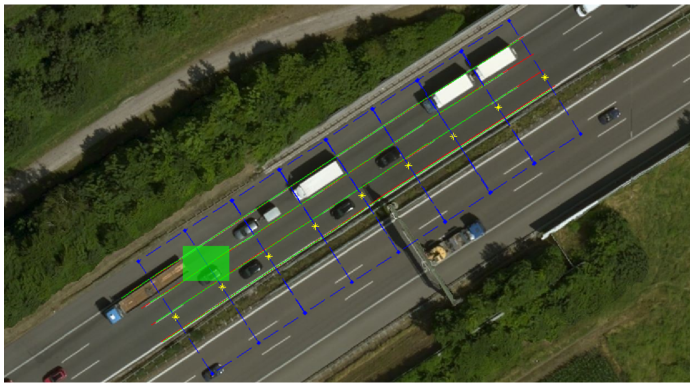
Table 3.1. Model level performance with different conditions.

Stage	Model Performance				
	Function Level		Geometry Level		
	Precision (%)	Recall (%)	Performance	Shift (cm)	Median Error (cm)
Before linking	84.96	95.58	76.01	0.37	4.26
After linking	95.79	95.79	74.38	0.87	4.24

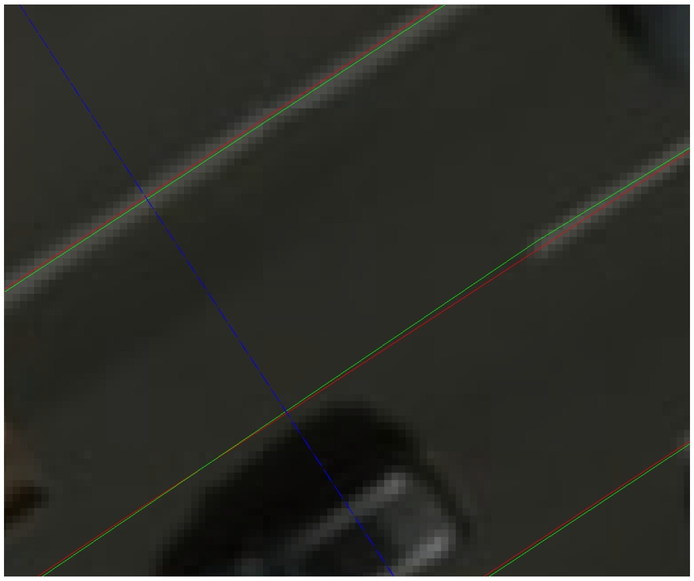
Sample of ground truth lane boundaries and final extracted lane boundaries rendered on satellite imagery is shown in Figure 3.14 (a). Each chunk is bounded by blue rectangle, yellow stars denote road trajectory, ground truth model is rendered in red, and the resulting lane boundary geometry is rendered in green.

The *shift* from experiment result shows the misalignment between ground truth coordinate (from point cloud) and testing data coordinate (from satellite imagery) is negligible on testing data, which was also validated by our observation. Benefited from the above-average conditions of the testing road (good painting quality, light traffic), the extracted lane boundaries achieved impressive results when measured against the ground truth before and after the linking stage at geometry level. Function level precision improves 10.83% by the linking stage while geometry level performance only dropped 1.63% due to the interpolated, synthetic lane boundaries. Even though the median error of the results is lower than 5 cm, limited by original satellite imagery resolution, we can claim that our lane boundary geometry accuracy is 30 cm [8].

[8]`shorturl.at/giwD9`



(a)



(b)

Figure 3.14. Result visualization overview (top) and zoom view of the green box (bottom).

CHAPTER 4

Accurate Vehicle Self-Localization in High Definition Map Dataset

4.1. Challenges

A self-localization system should return an accurate location longitudinally and laterally in the HD map's coordinate system, to enable the vehicle to accurately understand its surroundings and plan its motion. A desirable accurate vehicle self-localization has to address many challenges, including:

(1) The occurrence of unique objects (e.g. traffic signs, overhead structures, etc.) is sparse and random, and the occurrence of reference objects (e.g. end points of dashed line painting, light poles, etc.) is even and repetitive. If we have one unique object and one reference object like in Figure 1.3(a), we can easily triangulate/estimate the vehicle's location as shown in Figure 1.3(c). If we only have the end of the dashed line paint shown in Figure 1.3(b), which is the most common scenario on highways, it is hard to localize the vehicle shown in Figure 1.3(d).
(2) Image-based features, especially for image-to-image matching approaches, suffer from environmental factors such as varying illumination (i.e. shadows) and random objects such as nearby vehicles that either cause confusion

or occlude important features. At the same time, non-road objects always introduce more feature points. Figure 4.1(a) illustrates the average dash camera image across the entire dataset and Figure 4.1(b) illustrates Scale Invariant Feature Transform (SIFT) feature probability map of all images. This also influences Structure from Motion (SFM) based approaches, because fewer feature points are necessary to reconstruct 3D information.

(3) Computational speed and storage are key requirements for a real-time vehicle self-localization system. Ideally, the processing speed should reach 10 Hz to plan the motion in real time. Image-based data (especially perspective image data) is extremely large and slow to compute/display – however, it offers the richest information. On the other hand, abstract data (poly-points/vertices, text, occupancy grids, etc.) has the smallest size, but creates the greatest uncertainty and inaccuracy.

(4) Random error of consumer grade GPS is not consistent and they may have unexpected spikes. For example, around dash images #*301* and #*655* shown in Figure 4.7, upsurges of GPS error over *50* meters due to complicated terrain and obstruction are obtained, which makes raw GPS data too unreliable to be referenced “frame by frame”. Considering the first challenge, it is very difficult to localize a vehicle in real-time in a featureless scenario with imprecise GPS.

A feasible self-localization solution framework should focus on two primary environments: “feature-full” scenarios and featureless scenarios. A typical “feature-full”

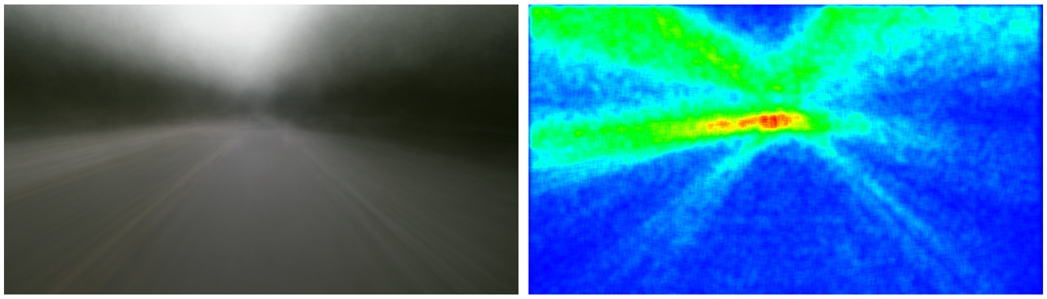

(a) (b)

Figure 4.1. a) Average color image of all *3583* dash camera images and b) SIFT feature probability map. (b) shows more features are obtained near vanishing point (center of the image) of road and non-road objects (upper part of the image), while fewer feature points are obtained on road surface (bottom part of the image).

scenario contains at least one unique object and one reference object, which allows the vehicle to triangulate its position by using either an image based approach or a point cloud/SLAM based approach. For featureless scenarios occurring between “feature-full” scenarios, feature-based tracking approaches like image-based lane/line tracking [**63, 91**] and SLAM-based tracking [**51**] or more traditional probabilistic-based approach like lane-level map matching [**189, 190**] are applied to correct the vehicle’s location on the fly.

4.2. Dataset Description

In this section, we provide the details of the dataset contributed in this work.

4.2.1. Vehicle and Sensor Configuration

A HERE True car (shown in Figure 4.2) was sent to acquire data for this dataset. The acquisition vehicle was equipped with a well calibrated (intrinsic and extrinsic)

HERE True platform that includes a Velodyne 32 LiDAR unit, high resolution cameras, and high precision positioning unit[1], which captures well-registered point clouds and street view imagery in world coordinates for high precision mapping purposes. Additionally, a consumer-grade dash camera was mounted on top of the windshield (above the roof of the vehicle) and roughly on the vehicle's major axis. A consumer-grade GPS receiver, was mounted closely behind the camera (longitudinally), which leads to a potential longitudinal error and slight negligible altitudinal and lateral errors. Relative positions of the dash camera and GPS receiver are not given in the dataset. We believe this configuration simulates a realistic case to let researchers design a generalized solution to tackle this task.

The vehicle drove and acquired two rounds of data on California State Route 13, the Warren Freeway, from $[37.8511°, -122.232°]$ to $[37.7774°, -122.1668°]$. The data was captured under reasonable traffic and weather conditions around 14:00 on January 20th, 2016. We reviewed the acquired data and selected one segment with the best data quality to build our HD Map, and another random segment was chosen to be the challenge segment, to make map coordinate and vehicle coordinate "relatively independent" even though the capture system is well calibrated. Also, the beginning and end of the test portion, which are transitional sections(i.e. entrance and exit ramps) from local roads to highway, have been removed to simplify the problem.

[1]Here collection vehicle v2.0, Engadget: `shorturl.at/ruxFX`

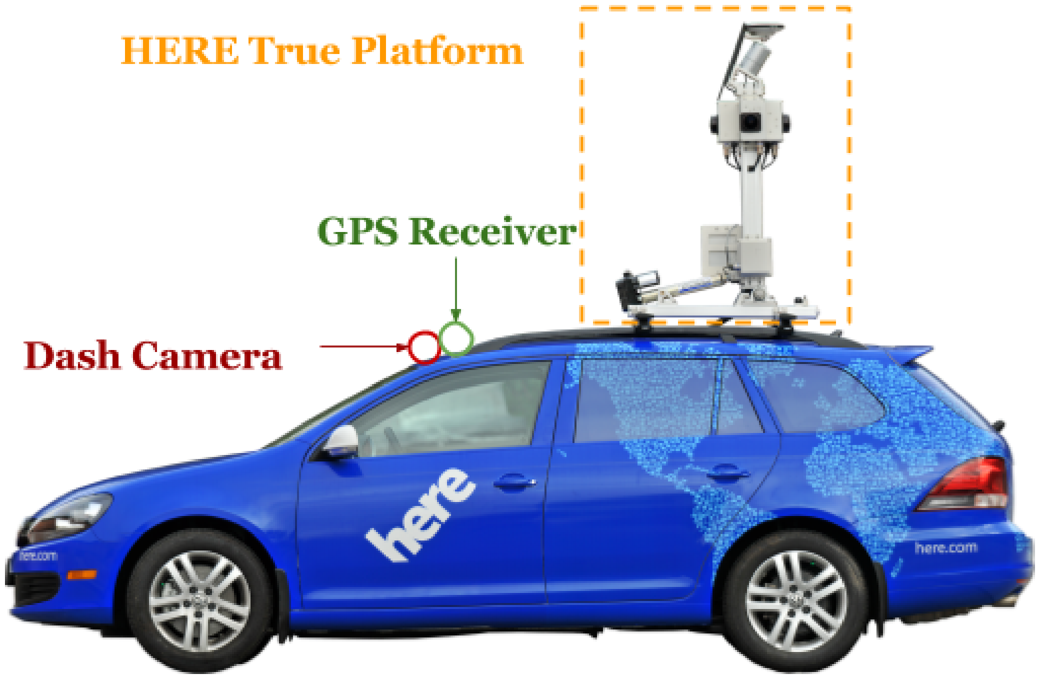


Figure 4.2. Side view of HERE True vehicle, with HERE True platform, consumer grade dash camera and GPS receiver.

The dash camera, consumer-grade GPS, and ground truth data are clock-synchronized and their sampling frequencies are *10* HZ, *5* HZ, and *10* HZ, respectively, as illustrated in Figure 4.4. Their timestamps are represented in UNIX time format, with time differences being 0.031, $7e^{-6}$, and 0.030 UNIX seconds, respectively. Also note that the fifth sample of each consumer-grade GPS sampling cycle is missing.

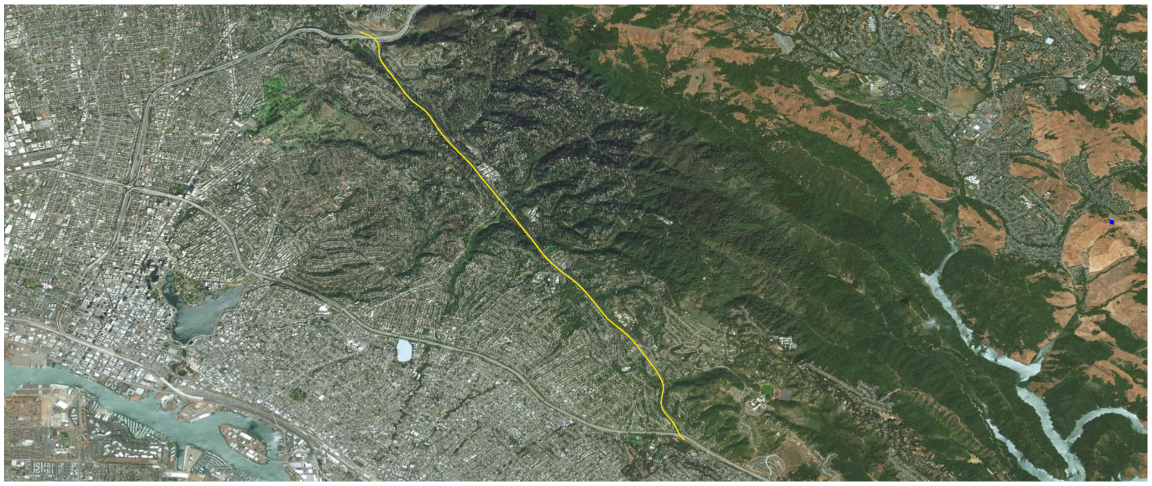


Figure 4.3. Illustration of our dataset trajectory.

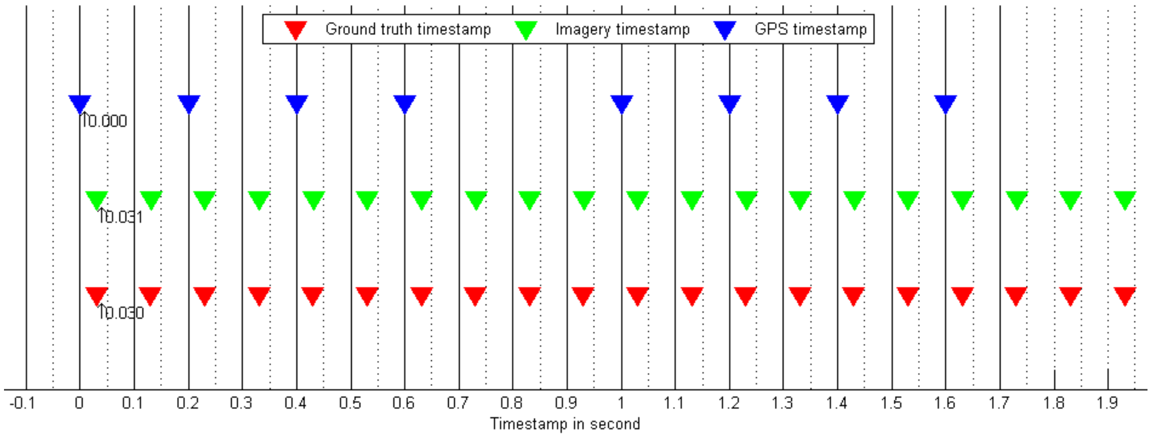


Figure 4.4. Illustration of two sampling periods of GPS ground truth, image sequence and consumer grade GPS timestamps. The horizontal axis is aligned with UNIX time.

4.2.2. High Precision Map Modeling

Navigation Data Standard (NDS)[2] is the leading data format for high precision maps in the industry, but due to its complicated data structure and current support features, we believe NDS is not "researcher friendly" enough for experiments with the self-localization problem. In this dataset, we manually modeled the road, divided the road into approximately 12-meter chunks along the vehicle's direction of travel, and represented each piece in a JSON format. Our road model contains three components:

- **: Lane Boundary**: The term "lane" refers to either a drivable lane or non-drivable lane (also known as the road shoulder), which means a boundary can be the physical boundary of the road or it can be a lane marking. Each lane boundary is represented by its geometry (a set of control points), color, function, and type. The accuracy of each control point is $1e^{-8}$ in degrees (which is equivalent to millimeter-level[3]) latitudinal and longitudinal.
- **: Occupancy Grid**: Similar to the occupancy grid concept in robotics, our occupancy grid represents a virtual voxel occupied by any stationary object on or above the current road surface in 3D space. Some examples include guardrails, light poles, road signs, and overpass bridges, shown in yellow cubes in Row 2, Figure 4.4. The occupancy grid can be used to generate a depth map of any given camera position and orientation on the road surface.

[2]NDS standard: `http://www.nds-association.org/thestandard`
[3]This is an estimated correspondence between angular distance in World Geodetic System 1984 (WGS84) to Euclidean distance in Cartesian.

- **: Road Sign**: Road signs (e.g. speed limit, route confirmatory, mile maker) are considered unique objects that offer the information for self-localization. The spatial location of a sign is represented as a 4-vertex bounding box; the text and the background color on the sign are also provided.

The structure of our JSON file can be found in dataset README file. Some samples of ground truth images, visualization of a road model, overlay of a road model and ground truth images[4], nearest dash camera images, and reference satellite images overlaid with road model of three scenarios are illustrated in Figure 4.4. Notice that the road model is not necessarily aligned with the satellite imagery because it is modeled in LiDAR coordinates.

4.2.2.1. Latitudinal/Longitudinal Editing and Elevation Correction. We used a custom 2D lane boundary geometry editing tool to create the high precision map. The acquired LiDAR point cloud is first projected onto a local tangent plane to generate an orthogonal background for the tool. Then, we manually drew polylines (an ordered set of points) to represent each lane boundary's geometry. Line function and type are obtained from the LiDAR projection and line color is obtained from the corresponding front-view image. Only latitude and longitude of a point can be encoded using this tool. When the collection vehicle goes around a curved road section, a large bank angle leads to a large shift of lane boundary through the vehicle's front view. To solve this elevation misalignment issue, we extract a high precision Digital Elevation Model from the point cloud [**229**]. Then, we assign altitude information to

[4]The ground truth images are only used for validation and visualization purposes, so they are not included in the dataset.

each control point by orthogonally projecting the point onto the DEM. An example of large bank angle at curved road section is illustrated in Figure 4.5. The yellow line and blue line refer to the camera Y axis and horizon (perpendicular to travel direction on local tangent plane). The red polyline and green polyline represent the road model before and after elevation correction. A half-lane-width shift can be observed in the right-most lane in this example.

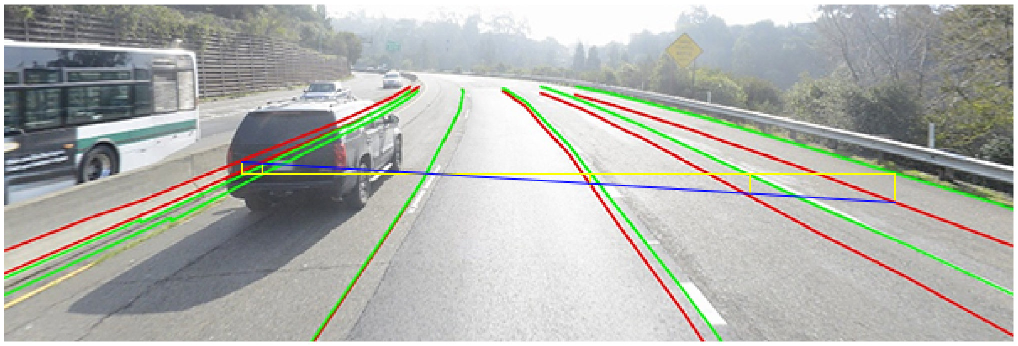


Figure 4.5. Illustration of lane model before elevation correction (red) and corrected elevation(green). Yellow line segment: camera y-axis. Blue line segment lies on current local tangent plane. An approximate 1-meter raising of the outer edge of the curved road above the inner edge can be observed. Note that there potentially exists a slight misalignment between the ground truth image and road model coordinates.

4.2.2.2. Occupancy Grids and Road Signs. An occupancy grid, with its z-axis orthogonal to the local tangent plane and y-axis oriented towards the current direction of travel, is extracted from the LiDAR point cloud and represented as voxels with 25-centimeter side-length. Compared to the original, unstructured point cloud, the occupancy grid represents the surrounding objects in less detail and therefore, requires less data storage[5]. The original occupancy grid includes all objects above the current road surface within the LiDAR range. However, for this dataset, we manually removed moving objects on both sides of the road and stationary objects on the adjacent road surface. Objects such as guardrails, light poles, traffic signs, trees, and overpasses on the current road surface, illustrated in yellow cubes in Row 2, Figure 4.4, were preserved. Road signs were also labeled, represented by four bounding box vertices, text information, and background color. One example is shown as the red polygon in Row 2 of Column c in Figure 4.4.

4.2.3. Dataset Statistics and Evaluation Metric

Different from the state-of-the-art self-location error evaluation metrics, which is represented in latitudinal/longitudinal angular distance or Euclidean distance in global coordinates, accurate in-map self-localization needs to be represented in "road/map" coordinates. Given a ground truth trajectory, which is a set of discrete locations with time stamps, and consumer-grade GPS points P^{lla}_{GPS} (lla denotes WGS84 coordinate), find the previous and the next (timestamp-wise) ground truth points $P^{lla}_{previous}$ and P^{lla}_{next}. Since GPS and ground truth sampling cycles are not necessarilly synchronized, a synthetic point $P^{ned}_{synthetic}$ (ned denotes local Cartesian North-East-Down (NED) coordinate) is interpolated in-between $P^{ned}_{previous}$ and P^{ned}_{next} as the ground truth

[5]Compression ratio is subject to many factors such as collection vehicle speed, LiDAR sensor density/angular speed/frequency and contour/shape of surrounding objects. In our case, the compression ratio is approximately 30:1

point of P_{GPS}^{ned}[6]. Notice that the projection of $P_{synthetic}^{ned}$ may not lie on line segment $\overline{P_{previous}^{ned} P_{next}^{ned}}$, due to either large GPS error or curved road section. Thus, we can define high-precision-map-based self-localization lateral error $e_{longitudinal}$ (meter) and $e_{lateral}$ (meter) as the distance from P_{GPS}^{ned} to $P_{synthetic}^{ned}$ along and perpendicular to $\overrightarrow{P_{previous}^{ned} P_{next}^{ned}}$, respectively. Altitudinal error $e_{altitudinal}$ calculated between $P_{GPS}^{lla} - P_{synthetic}^{lla}$ and $P_{GPS}^{ned} - P_{synthetic}^{ned}$ have negligible differences due to their tiny elevation value compared to earth's axis lengths.

[6]The transformation from P^{lla} to P^{ned} requires a reference point as local origin. In this case we use $P_{synthetic}$ as the origin. Different reference point selections can lead to slightly different $e_{lateral}$ and $e_{longitudinal}$ locally.

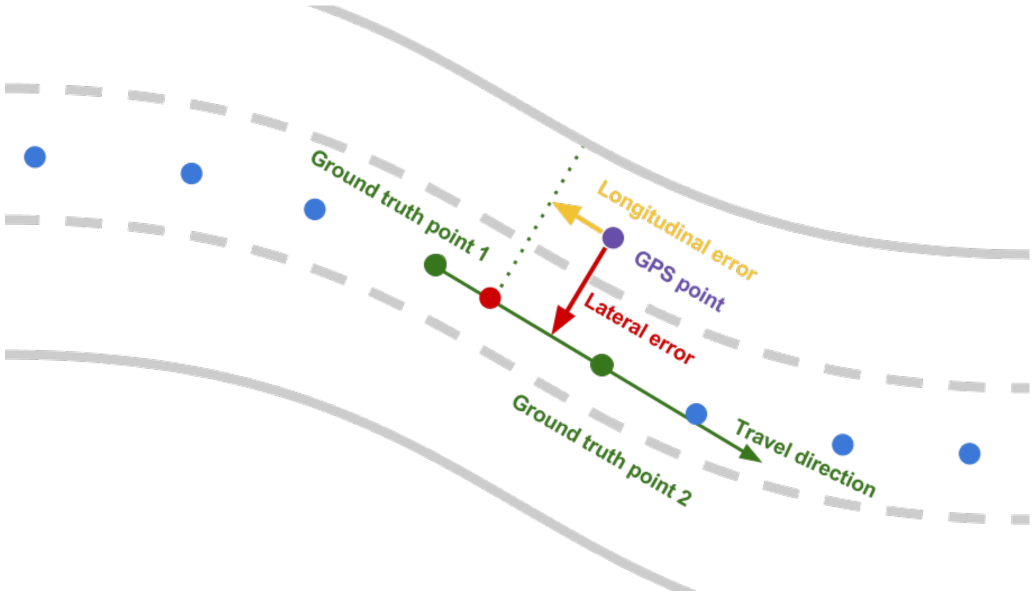


Figure 4.6. Illustration of vehicle self-localization accuracy definition. Blue point denotes ground truth trajectory pose point, two green points (ground truth point 1 and 2) denotes timestamp based previous and next ground truth points $P_{previous}$ and P_{next} of GPS point, red point denotes interpolated ground truth point $P_{synthetic}$, purple point denotes consumer grade GPS point P_{GPS}. Red arrow and yellow arrow denote vehicle self-localization lateral error $e_{lateral}$ and longitudinal error $e_{longitudinal}$ respectively.

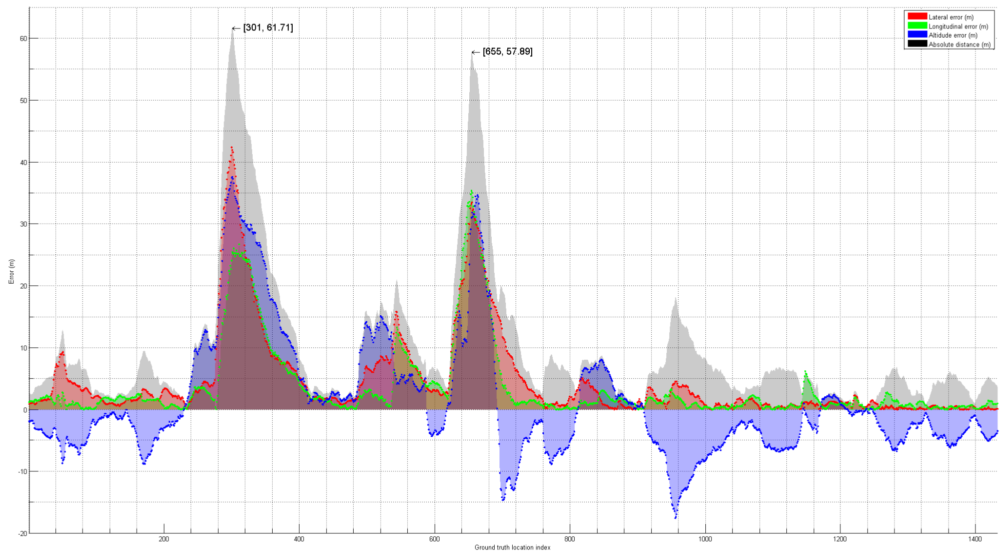


Figure 4.7. Error statistics of each consumer grade GPS location versus its corresponding ground truth location. Grey, red, green, and blue shades represent absolute, lateral, longitudinal, and altitudinal errors respectively. The horizontal axis represents consumer grade GPS sequence and the vertical axis represent error in meters. At image #301 and #655 the two largest errors can be observed due to complicated terrain.

With proposed self-localization evaluation metric, we calculated lateral, longitudinal and altitudinal errors of each GPS point and its corresponding ground truth point, illustrated in Figure 4.7. The statistics are listed in Table 4.1. Absolute distance statistics are also given. Notice that all five values are calculated from absolute value of e.

	$e_{lateral}$	$e_{longitudinal}$	$e_{altitudinal}$	$e_{absolute}$
Minimal (m)	*≈ 0*	*≈ 0*	*0.03*	*0.37*
Maximal (m)	*42.31*	*35.30*	*37.55*	*61.71*
Mean (m)	*4.74*	*3.67*	*6.97*	*9.88*
Median (m)	*1.74*	*1.33*	*4.70*	*5.93*
Standard deviation (m)	*7.5*	*6.42*	*7.32*	*11.75*

Table 4.1. Lateral, longitudinal and altitudinal error statistics along the entire dataset. $e_{absolute}$ denotes the L2 distance between each GPS point and its corresponding ground truth point in Cartesian coordinates.

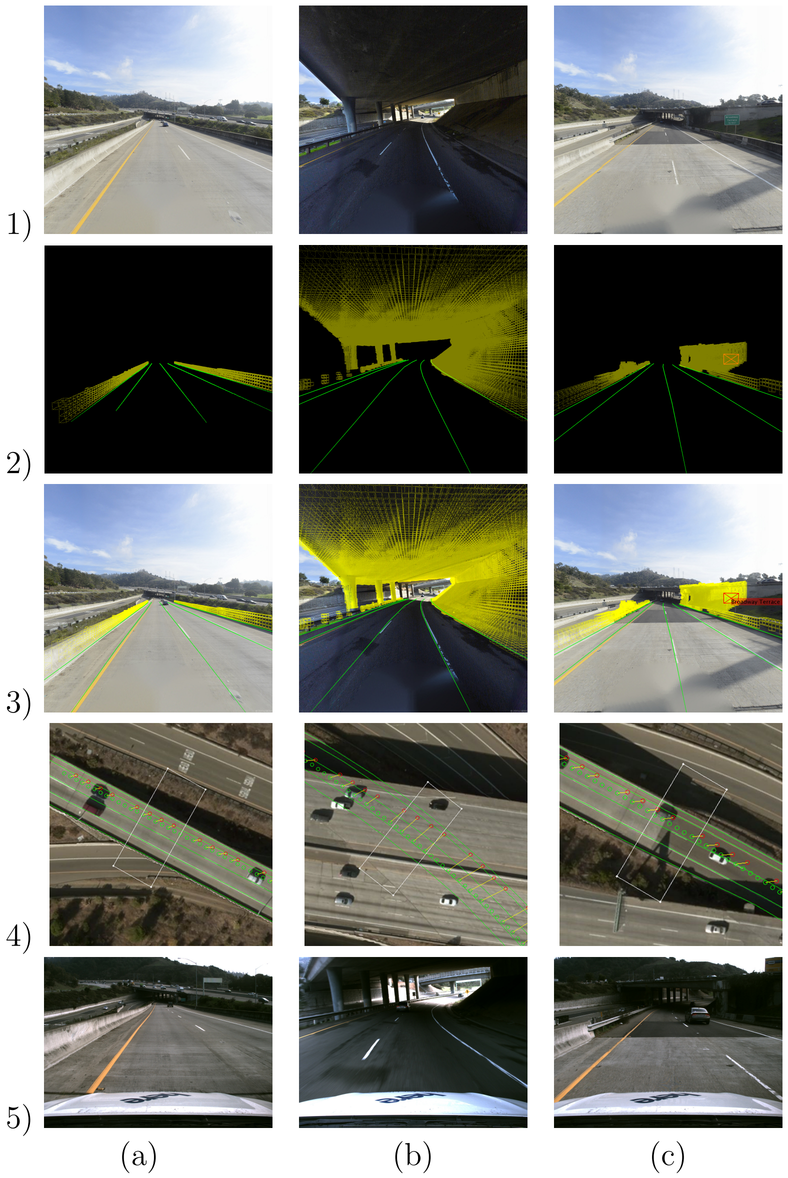


Figure 4.4. Row 1) Ground truth imagery, 2) road model visualizations, 3) overlays of imagery and road model, 4) corresponding aerial imagery overlaid with road model (note the slight misalignment), and 5) dash camera imagery. In each row, three scenarios are shown: Column a) a normal scenario, b) a scenario containing an overhead structure, and c) a scenario containing a road sign. In row 2) and 3), a green line denotes lane boundary geometry, a yellow cube indicates one occupancy grid voxel, and a red bounding box indicates road sign geometry. In Row 4), a green line denotes lane boundary geometry, a green circle indicates ground truth location, a red circle indicates consumer-grade GPS location in map coordinates, and a yellow line segment shows the matched pair.

CHAPTER 5

Towards Predicting Vehicular Data Consumption

5.1. Background and Motivation

Ensuring driving safety is a paramount in the autonomous driving industry, and the combining of on-board real-time sensing techniques and (external) knowledge based "verification" algorithms is a belt-and-braces approach to achieve the objective [**95, 165, 128**]. Low-level real-time perception systems involving cameras and LiDAR, accompanied with machine learning [**169**], have shown impressive performance in well-controlled environments and scenarios. However, with the help of HD maps [**86**], more functionalities can be realized, such as a higher level of assisted driving, improvement of fuel/energy consumption and driving experience/comfort.

Among other applications and systems, HD maps have been used in vehicle self-localization [**232**], however, their notable feature is that they are of enormous data size. They consist of road objects such as lane boundaries, pole-like objects, occupancy grids and other objects, which are also known as road "furniture". Nowadays, the HD maps can easily contain over a thousand voxels (highway scenario) or even tens of thousands of voxels (urban scenario) per road meter at a higher resolution, in contrast to dozens of points per road link in conventional maps [**233**]. Complementary to this, the map-based solutions – for example, self-localization [**232**], visualization and micro motion planning/adjustment – are computationally expensive. These two factors (data size and computational complexity) result in HD maps being the largest consumer of processing power and transmission bandwidth, from server end, through network, to vehicle/user end.

In recent years, numerous companies start sending out their experimental AVs (Autonomous Vehicles) on public roads. For example, in California, roughly 650 Avs have completed trips of cumulative length of $2,855,739$ and $1,955,201$ miles in 2019 and 2020 respectively [**131**]. Considering the market size of the "converntional" (without any assisted driving features) vehicles, increasing investment in [**96**] and the continuous growth of AVs [**197**], in the foreseeable future, vehicles with high level assisted driving functions are likely to dominate the market and, consequently, the occupancy of road networks. Currently, experimental/testing vehicles equipped with high performance on-board hardware can easily handle the load of both storage and computation – but with an overhead of high cost. When AVs are commercialized and operated as daily drivers, HD maps (and real-time/live maps) streaming will cause extremely heavy communication-load to the network, linearly increasing with the number of on-line vehicles.

Hence, optimizing the use of the HD maps data is of primary importance in many AV tasks. The most straightforward and widely used solution is to shrink/compress and improve/optimize the map data and data structure, in order to use the bandwidth more efficiently [**84, 29**]. In general, HD maps objects (i.e., furniture) are represented as voxels over an underlying grid [**179**], and are potentially compatible with hierarchical structures. Furniture can be downloaded on-demand and at different resolutions depending of the system configurations (e.g., hardware limitation and safety requirements) of particular use cases.

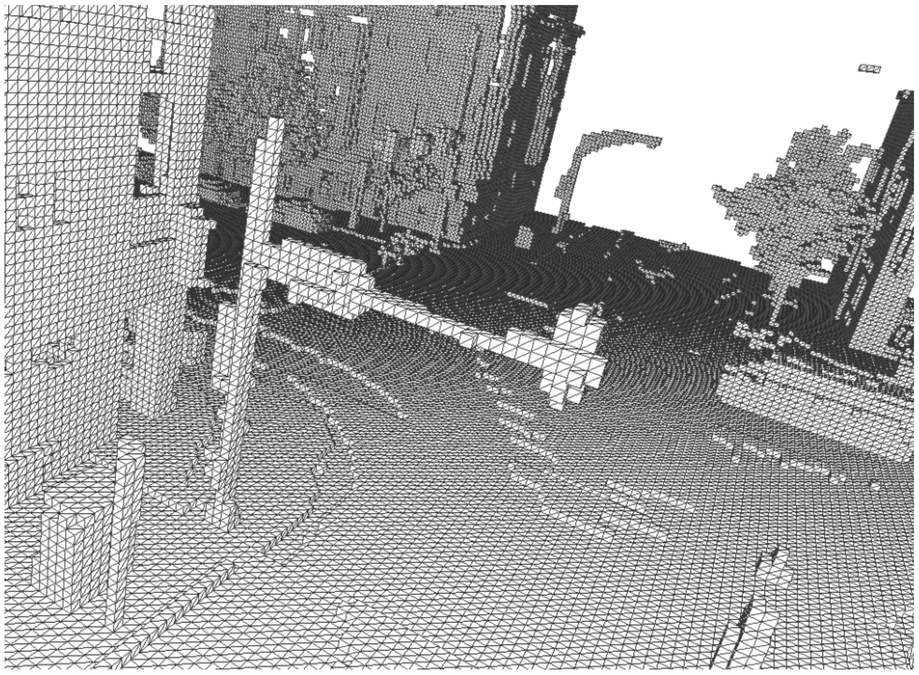
Brute-force downloading all the data, even with a high compression rate, does not solve the problem. In most cases, on-board hardware struggles with handling expensive algorithms which, in addition to delaying the progress of other applications, may also increase the decision response time. This, however, raises the issue of safety concern, along with the "side-effects" of bandwidth consumption and processing power on remote servers too.

If the MDC – in total, or even at each distinct time instants under current hardware constrains – can be predicted at the beginning of a trip, the system will have plenty of time to arrange the size of streaming data and, just as importantly, couple it with other external variables/factors (e.g., incorporating weather, traffic updates, etc.). This would enable designing a more reasonable/reliable maps data download strategy.

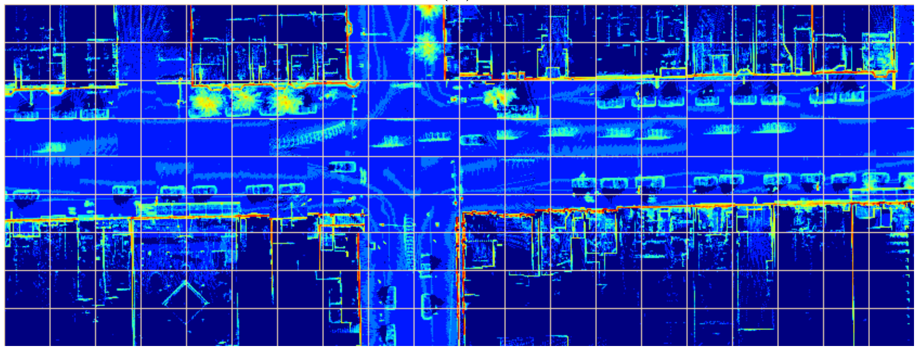
5.2. Related Works

We now overview the related works and position the dissertation in that context. We recognize that there are works which have addressed problems related to data use in mobile settings like, for example, data exchange in VANETS [**69**] and broadcast/indexing in air [**77**] – but we respectfully note that they are complementary to the current scope of the dissertation.

5.2.1. HD Maps



(a)



(b)

Figure 5.1. (a) A city intersection represented in an ocean of voxels at [41.896°, −87.670°] in Chicago, IL. The visualization is done by rasterizing millions of LiDAR points acquired for our previous work [**232**] into 20 centimeters voxels. (b) illustrates the tile bounding boxes at tile level 21 overlay on the city voxels heat map. (a) is the 3D view cropped from the center tiles in (b).

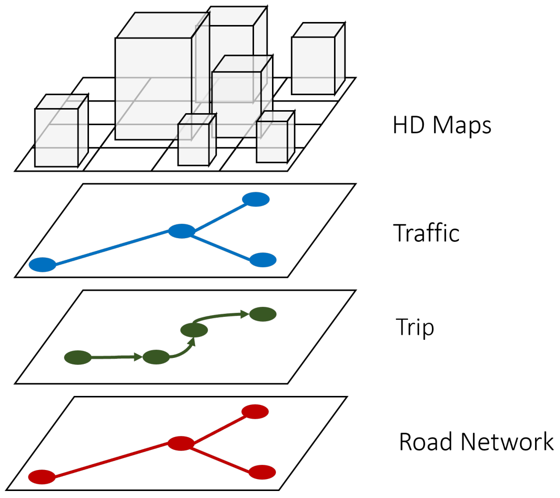
A quick rewind of HD maps and related applications. HD maps being used in AVs consist of at least lane boundary geometries, road signs, and other road furniture/objects, composed by points (for lane boundaries) and voxels (for other furniture), attached with other descriptive tags and information. Due to extremely high level of detail, their data size is much larger than conventional maps. A visualization of a city intersection composed by voxels is shown in Figure 5.1[1]. For references, in conventional maps, dozens of control points are sufficient to represent a hundred even a thousand meters long center line of a lane or road. In contrast, in HD maps, each urban lane meter can have over 1.2×10^4 and 4×10^3 voxels at resolutions of 10^{-1} meter and 2×10^{-1} meter respectively [**232**]. Voxels also have the advantage of potentially compatible with tree-like (i.e., quadtree and octree) hierarchy structures. There are mainly two types of containers that we can use to organize the road furniture which are distinct from the container dimensions: attach objects to one-dimensional road network and two-dimensional global tile [**214**]. One-dimensional data structure is rarely used [**192, 102**], because the objects do not have a global view, which may cause data redundancy (same furniture appears/obtained multiple times from different link/edge) and makes global optimization and alignment harder. Tile-like structure is the ideal container but inherently incompatible with road network (graph representation).

Similar to the definition of energy/fuel consumption in vehicle energy management study (i.e., miles per gallon or watt-hours per mile), MDC is based on the size of

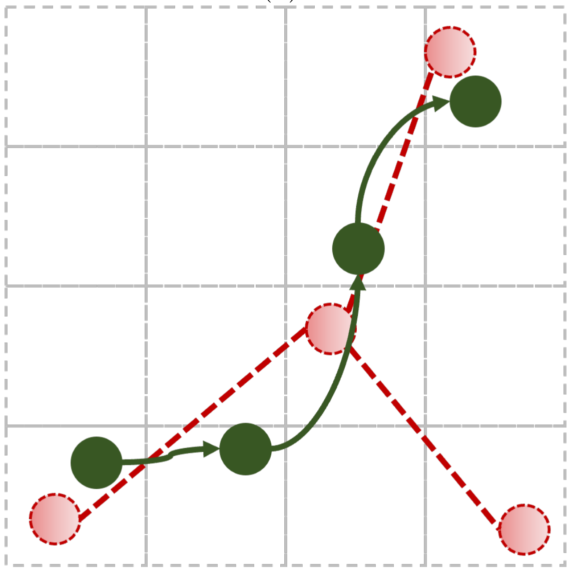
[1]Tile level, also known as the Bing Maps Tile System, is a quartered Mercator projection. Level 21 is equivalent to a 12.5 meters/tile ground resolution.

the maps data that a vehicle needs for its semi-autonomous or fully-autonomous driving function(s) [**233**]. One possibility to quantify the MDC is by the amount of data the vehicle needs to load for executing real-time applications. Specifically, the vehicle needs to load surrounding objects, represented in polygons/vertices or grids/voxels depending on the object representation. Even though representing objects in vectors can significantly reduce the size of the data and has invariance advantages such as scale and shift, raster representation is still more widely used in real-time autonomous driving applications since (cf. [**233**]): (1) sensors (e.g., LiDAR, depth/stereo cameras) equipped on vehicles acquire data that is either directly represented in raster format, or can straightforwardly converted into raster-like information. (2) most algorithms for autonomous (and assisted driving) applications, such as the ones used in vehicle self-localization, need to be fed with raster data [**222**].

Real-time vehicle self-localization is the most popular and critical application using HD maps in autonomous driving industry from the broad perspective of decision/action taking. A safe and reliable autonomous driving system should take an action (from observation to decision) no slower than human reaction time – 200 milliseconds to 1 second depends on the specifics of a particular task [**90**]. Increased computational expenses when capturing specific objects (e.g., pole-like objects, guardrails/curbs, road surface/pavement) or even entire surrounding environment have been addressed in [**205, 207, 101, 160**]. If there are higher safety requirements (under certain internal and external factors/constraints), the resolution of retrieved voxels needs to be increased, causing a substantial increase in the size



(a)



(b)

Figure 5.2. (a) The stacking of four types of data: HD maps, traffic, trip, and road network from top to bottom; (b) the overlay of tiles (gray grids), a trip (green trajectory) and road network (red graph).

of the map data to be downloaded. The size of objects information that the vehicle needs is also highly dependent on the sensor configuration, such as refresh rate, layout and orientation, angular/spatial resolution, sensing range, and even vehicle's motion. On-board acquisition is irrelevant to our work, we only focus on the data (size) retrieved from server.

5.2.2. Map Data Consumption

Similar to the definition of energy consumption in vehicle energy management study, MDC is based on the size of the maps data that a vehicle needs for its semi-autonomous or fully-autonomous driving function(s) at each moment [**233**]. We note that energy consumption is a well-studied field, and relevant values (e.g., Joule, the weight/volume of the petrol/gas) can be measured/quantified by on-vehicle sensors (and energy density charts) [**57**].

One possibility to quantify the MDC is by how much map data the vehicle needs to load for executing real time applications. Specifically, the vehicle needs to load surrounding objects, represented in polygons and grids/voxels – also known as vector and raster representations respectively. Even though representing objects in vectors significantly reduces the size of data and has invariance advantages in scale and shift, raster representation is still more in real-time autonomous driving applications since (cf. [**233**]):

(1) sensors (e.g., LIDAR, depth/stereo cameras) equipped on vehicles acquire raster-like information directly.

(2) most algorithms for autonomous (and assisted driving) applications, such as the ones used in vehicle self-localization, need to be fed in raster data.

If there are higher safety requirements, the data resolution of retrieved voxels per unit area needs to be increased, causing substantial increase in the size of the map data to be downloaded [**233**].

The size of object information the vehicle needs is also highly dependent on the sensors configuration, such as refresh rate, layout and orientation, angular/spatial resolution, sensing range, and even vehicle's motion.

5.2.3. Travel-time Estimation

Trip planning, ETA (Estimated Time of Arrival), as well as some related "derived" topics such as fuel consumption prediction [**208**] and electric vehicle energy management [**57**] are the closest ideas to our objective. Therein, the travel-time estimation – also known as ETA and Origin-Destination (OD) time estimation problem [**204, 41**] – is one of the widely used tasks in of high importance in location-based services/applications in both consumer market and industry. Given a pair of origin and designation locations (or the entire route), and prerequisite road network and other information/pattern (such as traffic, weather and accident), an accurate time predicting result not only benefits consumers' everyday life, but also the optimization of entire social system in the aspects such as logistic and ride-sharing [**12**].

The history of solving ETA problem can be traced back for decades, evolving from simple statistic model and regression [**154**] to modern convolutional neural network (CNN) based solutions [**202, 105**]. Numerous works using LSTM [**47, 140**], GNN [**203, 103, 104**], the hybrid of LSTM and GNN [**113**], and even image based [**56**] solutions have achieved impressive results. The concurrent works such as Curb-GAN [**237**] and DeepOD [**227**] integrate/embed external factors and historical data into the training process and show the strong correlation to prediction results.

5.2.4. Learning Approaches: LSTM and GNN

Linear models, such as Linear Discriminant Analysis (LDA) and Support Vector Machine (SVM), have been prediction tasks for decades [**176**]. In the fields of geospatial and transportation, the trajectories data is not only temporally continuous, but also spatially continuous. Machine Learning (ML) based approaches, such as Long Short-Term Memory (LSTM), RNN and their variants, have provided successful improvements in solutions to multiple mobility-related problems and applications: driver's behavior prediction [**58, 127**]; trajectory data mining/prediction [**59, 3, 223**], and vehicle speed/energy consumption prediction [**110, 57**]; and more broadly, Location Based Services (LBS) [**93, 238**].

The fast development of LSTM has drawn substantial attention due to its ability to model the long-term/historical dependencies of time-series data such as speed, ETA [**112**], fuel/energy consumption [**208, 57**] for a single vehicle. Meanwhile, the lacking of information from adjacent "samples" (such as links/edges in a graph) limits the LSTM from learning surrounding knowledge.

Complementary, GNN-like architectures are potentially compatible with road networks and have inherent advantages over LSTM-like solutions. Conventional

GNNs have disadvantages in handling the changes of the graph, not only the deletion/insertion of nodes, but also the ever-changing features of the nodes. Re-training the model is needed in order to represent this node [**175, 139**] once changes are obtained. Fortunately, being different from social network, maps and HD maps are relatively "stable"/staic on both topological and featurization aspects. Researchers working on dynamic graph representation learning [**151, 220**] are trying to eliminate such issues.

This dissertation takes a first step towards addressing the PMDC problem from the perspective of broader context awareness. As it has not been studied, one of the tasks was to generate and HD maps dataset along with trips and traffic datasets, since the existing HD maps are proprietary and, typically, confined to smaller geographic areas, and are not integrated with other (e.g., traffic, road-network) data sources. In this spirit, our solution also investigates the benefits of including additional data sources. What also separates our work from the related literature is that we combine data from multiple heterogeneous but spatially correlated sources, as illustrated in Figure 5.2. More specifically, we embed the HD maps – or, in a broader sense, the tile/raster based data – as a part of the heterogeneous input of a graph representation learning architecture, and design a unique framework to train MDC problem with other important (e.g., traffic) data sources, and use it to solve the PMDC of a trip.

5.3. Problem definition

In this section, we describe the specifics of the data sources, and present formal definitions of the concepts used in the rest of this chapter.

Maps and HD maps. We assume that an HD map is represented in a widely used 2D tile system for geo-regions, accompanied with a resolution value, denoted by $M \subseteq \mathbb{R}^{|P|\times|Q|\times|R|}$, where $|P|$ is the number of cells along x-coordinate; $|Q|$ is the number of cells along the y-coordinate of the suitably selected system; and R is the set of resolution values used among the cells. Typically, the values of R correspond to voxel-sizes (i.e., one can have different resolution levels for a given configuration of 2D tiles). The cell $(p, q, r) \in M$, denoted $m_{p,q,r}$, contains the voxels corresponding to single tile at tile coordinate $p \in P$, $q \in Q$ and resolution $r \in R$ in a respective grid of the geographical area of interest.

We assume a conventional road network represented as a directed graph $G = {<}V, E{>}$, where the elements of V (i.e., vertices) correspond to an intersection, and the elements of E (i.e., the edges) correspond to road segments. Each $v_i \in V$ has unique location, specified by its coordinates $(v_i.x, v_i.y)$. Similarly, each $e_k \in E$ is represented as the triplet ${<}u_k, v_k, w_k{>}$ where $u_k, v_k \in V$ are the start node and end node of e_k, while w_k denotes the "weight", which could stem from different context, such as: length, traffic travel index (TTI), or HD maps corresponding to e_k.

We further assume that M and G are specified in the same coordinate system. However, we note that an extra linear projection is still needed to calculate the conversion from G to M. To simplify the problem, we introduce $I_{({<}p,q{>},r)}$ to indicate the quantity of that data with respect to a tile location ${<}p, q{>}$ and resolution r, and $f_p({<}x, y{>}) = {<}p, q{>}$ to indicate the project from geolocation ${<}x, y{>}$ to tile coordinate ${<}p, q{>}$.

Given a vehicle at $<x, y>$ with a set of configuration c (a combination of internal and external factors, such as speed, acceleration, hardware configuration, weather and traffic), the vehicle needs the surrounding information not only consists of the current tile $f_p(<x, y>)$, but also nearby tiles, at certain resolution(s). We also define piecewise-defined functions $f_d(c) = d$ and $f_r(c) = r$ which use c to determine a pair of tile search size $d, d \leq 0$ and resolution $r \in R$. Therefore, the HD maps information of vehicle at $< x, y, c>$ can be represented by a set B consists of HD maps indices, where

$$\begin{aligned} B_{x,y,c} &= \{f_p(<x, y>) + <p, q>, r\}, \\ p &\in [-f_d(c), f_d(c)], q \in [-f_d(c), f_d(c)], r = f_r(c) \end{aligned} \tag{5.1}$$

and then the MDC at single moment $< x, y, c >$ can be formalized as:

$$MDC_{x,y,c} = f(M, B_{x,y,c}) = \sum_{b \in B_{x,y,c}} I_b. \tag{5.2}$$

Trip. A trip, or a trajectory, is a sequence of geospatial points represented as $L = \{l_i\}$, where $l_i =< x_i, y_i, t_i, c_i >$ or $l_i =< x_i, y_i, z_i, t_i, c_i >$[2] is the i^{th} point of L. $<x_i, y_i>$, t_i and c_i denote the geo-location, timestamp and configuration respectively.

In this study, a raw trip L needs to be converted into a graph G representation and then processed with GNN framework. Let $L_m = <x'_i, y'_i, t_i, c_i>$ denote the map-matched L to graph G, and $L_g = <u_j, \tau_j, \epsilon_j>$ denote the trip in graph representation,

[2]In this dissertation we only use 2D points/cells. Z-axis in coordinate system (e.g., altitude) should be considered in future 3D transportation, such as drone delivery.

where $<x'_i, y'_i>$ is the map-matched point of raw point $<x_i, y_i>$ at t_i, and $u_j \in V$ is a node between map-matched points $<x'_i, y'_i>$ and $<x'_{i+1}, y'_{i+1}>$, τ_j is the interpolated timestamp between t_i and t_{i+1}, and ϵ_j is the interpolated configuration of c_i and c_{i+1}, for the cases when $<x_i, y_i>$ and $<x_{i+1}, y_{i+1}>$ do not match to a same edge.

Define $dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>)$ is the in-graph distance between $<x_i, y_i>$ and $<x_{i+1}, y_{i+1}>$, in this example:

$$\begin{aligned} & dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>) \\ =& dist(<x'_i, y'_i>, <x'_{i+1}, y'_{i+1}>) \\ =& dist(<x_i, y_i>, u_j) + dist(u_j, <x_{i+1}, y_{i+1}>). \end{aligned} \tag{5.3}$$

Then we can define interpolated τ_j as

$$\tau_j = \frac{dist(u_j, <x_i, y_i>)}{dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>)} \times (t_{i+1} - t_i) + t_i. \tag{5.4}$$

as well as ϵ_j (assume c can be interpolated)

$$\epsilon_j = \frac{dist(u_j, <x_i, y_i>)}{dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>)} \times (c_{i+1} - c_i) + c_i. \tag{5.5}$$

Note that if the trajectory sampling rate is sparse or many intersections are clustered together, multiple nodes may occur between two adjacent $<x'_i, y'_i>$ and $<x'_{i+1}, y'_{i+1}>$.

MDC. Assume there is a trip $L_t = \{<\hat{x}_k, \hat{y}_k, \hat{t}_k, \hat{c}_k>\}$ converted from a map-matched trip L_m, has a perfect sampling rate which let there is one and only one trajectory point locates in each adjacent tile, where $f_p(\hat{x}_k, \hat{y}_k) = f_p(\hat{x}_{k+1}, \hat{y}_{k+1}) \pm <\{0,1\},\{0,1\}>$. To form such trajectory, if there is no such trajectory point locates in a tile from the map-matched trip L_m, a new point should be interpolated and the construction of its $\hat{c}$ follows Equation 5.4; if multiple trajectory points cluster in tile, the center trajectory point will be selected. Note that, L_m is unidirectional transferred from L, L_g and L_t are unidirectional transferred from L_m, and their relationship is shown in Figure 5.3. $|L| = |L_m|$, but not necessary equals to $|L_g|$ or $|L_t|$.

Thus, the total MDC of trip L can be formalized as

$$
\begin{aligned}
&MDC_L = MDC_{L_t} = \\
&MDC_{<\hat{x}_1,\hat{y}_1,\hat{c}_1>} \uplus \cdots \uplus MDC_{<\hat{x}_{|L_t|},\hat{y}_{|L_t|},\hat{c}_{|L_t|}>},
\end{aligned}
\tag{5.6}
$$

where $\uplus$ denotes a special MDC accumulation operation which unions $\cup$ of two sets of HD tiles inside function f. For instance,

$$
\begin{aligned}
&MDC_{<\hat{x}_1,\hat{y}_1,\hat{c}_1>} \uplus MDC_{<\hat{x}_2,\hat{y}_2,\hat{c}_2>} = \\
&f(M, B_{\hat{x}_1,\hat{y}_1,\hat{c}_1}) \uplus f(M, B_{\hat{x}_2,\hat{y}_2,\hat{c}_2}) = \\
&f(M, B_{\hat{x}_1,\hat{y}_1,\hat{c}_1} \cup B_{\hat{x}_2,\hat{y}_2,\hat{c}_2}) = \sum_{b \in B_{\hat{x}_1,\hat{y}_1,\hat{c}_1} \cup B_{\hat{x}_2,\hat{y}_2,\hat{c}_2}} I_b.
\end{aligned}
\tag{5.7}
$$

We note that, whenever there is no ambiguity, we will omit certain subscript(s) and/or superscripts. Thus, for example, to denote the MDC of a given trajectory L, we will use MDC_L when clear from the context.

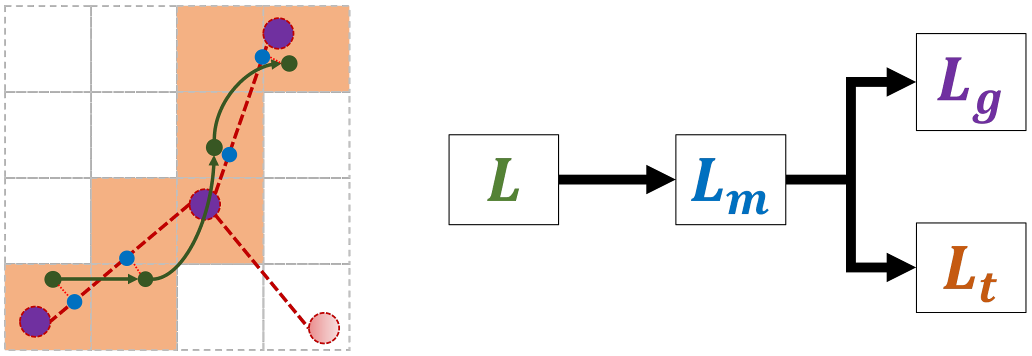


Figure 5.3. Illustration of L (green), L_m (blue), L_g (purple) and L_t, and their spatial correlation/transformation.

Portion MDC. Recall $L = \{l_i\}$ and MDC_L, the MDC of L can also be represented as $MDC_{l_1,l_{|L|}} = MDC_L$, where l_1 and $l_{|L|}$ are L's first point (start point) and destination (end point). Similarly, $MDC_{l_1,l_j}, j \in (1, |L|)$ denotes the MDC from l_1 to l_j. Hence, the $j-1^{th}$-to-j^{th} portion of L can be represented as

$$MDC_{l_{j-1},l_j} = MDC_{l_1,l_j} - MDC_{l_1,l_{j-1}}. \tag{5.8}$$

Given the MDC definition of a trip, the portion MDC of two adjacent trajectory

We note that, whenever there is no ambiguity, we will omit certain subscript(s) and/or superscripts. Thus, for example, to denote the MDC for a specific time instant t_i for a given trajectory l, we will use $MDC_{(l,t_i)}$, or even MDC_{t_i} when clear from the

Notation	Description
L, L_g, L_m, L_t	Trip in raw, graph, map-matched and tile representations
x, y, t, c	Raw locations, timestamp and configuration of L
x', y'	Map-matched locations of L_m
$\hat{x}$, $\hat{y}$	Processed locations of L_t
τ, $\hat{t}$	Processed timestamps of L_g and L_t
$\hat{c}$	Processed configuration of L_t
d, r	HD maps search range and resolution
f_p	Linear projection function from graph coordinate to tile coordinat
f_c, f_r	Range and resolution calculation function
f	HD maps data consumption calculation function

Table 5.1. A quick reference of easily-confused notations been used in this dissertation.

context. In addition, when we are dealing with consecutive time-stamps with respect to a known starting time of a given trajectory, we will simply use MDC_j to denote the j^{th} time unit after the initial time-stamp value of that trajectory. Similarly, we will use MDC'_{t_j} to denote the predicted value for time instant t_j (or, relative to a known starting point, simply MDC'_j).

For convenience, the symbols used throughout the chapter (along with their intended meanings) are summarized in Table 5.1.

5.4. Data Preparation: A Synthetic City

Since there is no existing dataset collected or built for this PMDC, we had to invest some efforts to build a dataset by/for ourselves. In this section, we describe which datasets we used for integration in creating the "synthetic city" dataset (SCD) used in the experiments.

5.4.1. Traces, Traffic and Maps

As we mentioned in the previous sections, HD maps (along with SD maps, which describe the basic topology and geometry of the road network) and traces/trips are the fundamental features of our model.

The traces need to have the following properties: dense and consistent sampling rate, and large number of traces acquired across a wider time window. Some public datasets, such as government released New York Taxi [**183**] consists of millions of trips collected during a decade. However, the sampling rate is extremely sparse – sometime only start/pick-up and destination time and locations are recorded. These may be good enough when used in applications focusing on ride sharing and traffic analysis, however, they are too sparse to be used in autonomous (or assisted) driving. Other trajectory datasets, such as Roma Taxi [**17**] and T-drive [**228**] collect taxi data during a long time period and consist of plenty of trips. Even though the sampling rate of these datasets is much higher, it is still at the level of a several minutes which again, is not high enough to satisfy real-time requirement. DiDi offers several datasets that meet the demanding requirements of real-time autonomous driving. The datasets are collected during 2016, in greater Xi'an city area and organized in months.

However, to enable experimenting with the other features, traffic information is also important. Open source providers, such OpenStreetMap (OSM), typically have a pre-allocated attribute called *Key:Flow* which denotes the traffic flow information – but does not cover too many cities. Leaders in the mapping industry - such as

HERE Maps [**184**], Google Maps [**116**] and Bing Maps [**115**] - do have their own traffic flow datasets, but they are not public accessible. Fortunately, DiDi also offers traffic information (travel time index, DiDi TTI) collected during certain months of 2017 [**39**].

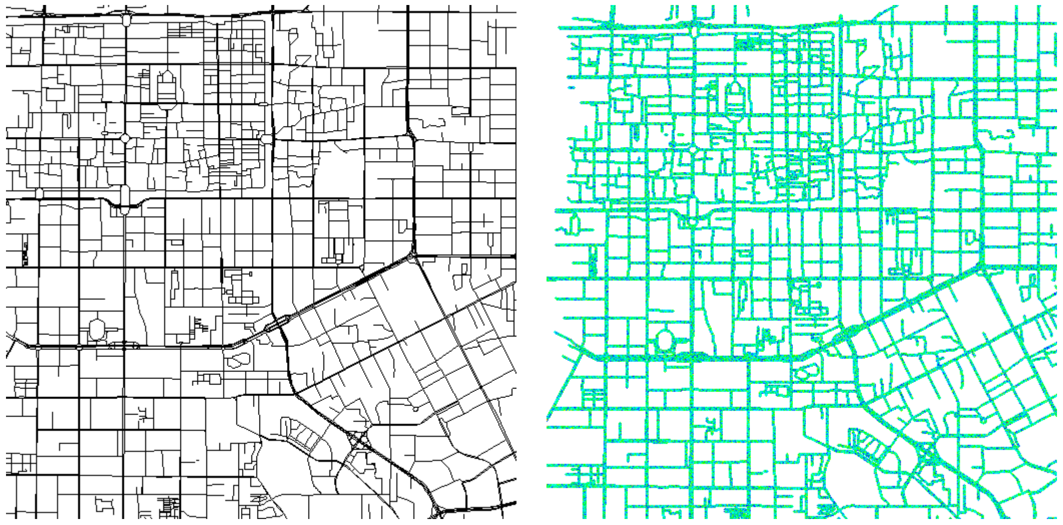


Figure 5.4. OSM road map of Xi'an downtown (left) and the heat map of (synthetic) HD map data size per small tile (12.25 meters per pixel) of Xi'an (right), within the region (34.207309°, 108.92185°), (34.279936°, 109.009348°).

5.4.2. Data Processing and Integration

We integrate heterogeneous data from various sources to create a synthetic city, which includes: road network (in graph representation) from OSM [**136**], traffic information (in plain text) from DiDi Open Dataset [**39**] and HD maps (in tile system) from previous work [**233**].

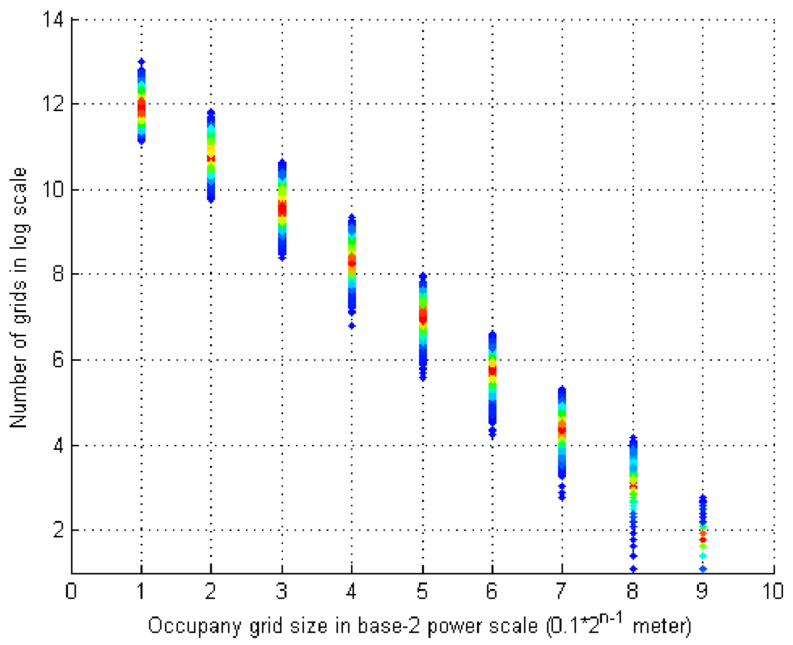


Figure 5.5. Illustration of the distribution of voxel densities at different resolutions, from 0.1 meter to 25.6 meters, acquired in Chicago area.

5.4.2.1. Road network. Road network is the skeleton of the dataset. First of all, a road map of Xi'an metropolitan area is retrieved from OSM (DiDi TTI also comes with a road network map but with missing too many road links, which let it unusable). The bounding box of this map is narrowed down to the downtown area $(34.279936°, 108.92185°, 34.207309°, 109.009348°$, an $8,060$ meters by $8,053$ meters (lateral) area) to aggregate the density of traces and road network. $1,814$ OSM links (roads) and $7,421$ OSM nodes (control points) are retrieved in this area, and its visualization is shown in Figure 5.4. Due to the data security policy, The data (both trips and traffic) provided by DiDi is enforced to be encoded in *GCJ-02* coordinate [**174**]. Which, in turn, a consistent (locally) misalignment[3] between DiDi's coordinate and OSM coordinate is obtained (by manually aligning several intersections). A graph with 4771 edges and 2140 nodes is generated when converting the raw OSM data into graph representations.

5.4.2.2. HD Maps. There are not many HD map datasets available in either academia or industry, and none of them has the ability to cover (larger regions of) an entire city. Our current work only needs the *size* of HD maps per unit (tile) area or road length.

In the context of maps and self-driving related applications, a voxel defines a cube in 3D Cartesian coordinate system (center and edge length), and its value can either indicates if this unit space is occupied by a road object (e.g., buildings, trees, pole-like objects, signs), or contain richer information (object type, color, etc.). There is a positive correlation between the size of HD maps per tile, and the number of voxel/occupancy-grids within a tile. Given an object (e.g., a cube), the number of voxels used to represent it grows cubically as a function of the decreasing of voxel size. Subject to the acquisition techniques, such as camera and LiDAR, only the exterior of an object can be recorded. To build an industry level HD maps, normally, acquisition vehicles mounted with LiDAR sensors are sent to acquire raw point clouds of the city. In post processing, point clouds (from either the same acquisition or different acquisitions) will be aligned, filtered (remove unwanted objects such as vehicles and

[3]The misalignment from OSM's coordinate to DiDi's coordinate is $[-0.0016°0.0047°]$ roughly equivalents to a 468.1 meters ground resolution.

pedestrians), and then converted to voxels. Although we cannot imitate the 3D world from one real city acquisition to another city, the number of voxels per tile can be transferred.

Define the voxel density as the number of voxels per surface unit area. The voxel density distribution can be calculated from spatially partitioning the acquisition (from a given/reference city) into unit areas. Then, similarly, we partition a new city (with no HD maps available) into the same size unit area as well and assign the same voxel density distribution as the reference city.

We learn the voxel density distribution (at different resolutions) relying on previous work acquired in Chicago area [**233**], and then apply this distribution to create a synthetic HD maps for Xi'an. The heat map shown in Figure 5.5 illustrates the distributions of number of voxels at different resolution, pertaining to a single tile, where the size of a tile 25.6×25.6 meters. We note that the number of voxels can exceed the ratio of the 2D-based vs. tile area, because voxels can be stacked vertically within a particular area. It turned out that these distributions follow normal distribution well at multiple levels of details. In our experiment, we choose the highest resolution at 0.1 meter, which has a normal distribution with $\sigma^2 = 4.78 \times 10^4$ and $\mu = 1.57 \times 10^5$, and 6.79×10^4 and 4.47×10^5 voxels as the hard-coded minimum and maximum bounds respectively. With this fitted seed of distribution, an HD map is generated, which is then attach to the road network. The value of each tile represents the number of voxels within this tile, to be used as the representation of its data size. The heat map of the size of the HD map is visualized in Figure 5.4 (right).

5.4.2.3. TTI integration. DiDi TTI dataset [**39**] offers the traffic information (travel time index (TTI) and speed) collected during a different time window from the trips dataset (2017 and 2016 respectively), it is impossible to fuse (assign the traffic information to each trajectory point) them together directly. To solve that, we select the traces [**38**] and traffic from the same month (but different years), and normalize/map/project the traffic information (of each link) during a month to a "week calendar" – which has a size of 7 days by 6×24 time sections (TTI information is recorded every 10 minutes for each link) per day, which = 1008 time slots in total, and it shows consistency across the month. Hence we project this traffic information to the traces to assume that the historical (even though it is collected after the trace data) TTI information can be used as the traffic information. We project this traffic information to the traces and we show the heat map of trip starting times in Figure 5.7.

Note, 53.20% of edges have traffic information recorded, and accounts 61.82% of the total length of the road network. If there is no traffic information attached to an edge, 0s will be assigned. Figure 5.6(a) visualizes the edge's length-to-MDC distribution.

5.4.2.4. Trips and Map-matching. The trips we select come from DiDi trips [**38**] which were acquired during October, 2016. This dataset has some features and characteristics such as second-level GPS sampling rate, intensive samples during a time window (number of vehicles and orders), which let it stands out from other popular datasets such as Roma Taxi [**17**] and T-drive [**228**]. Since DiDi includes

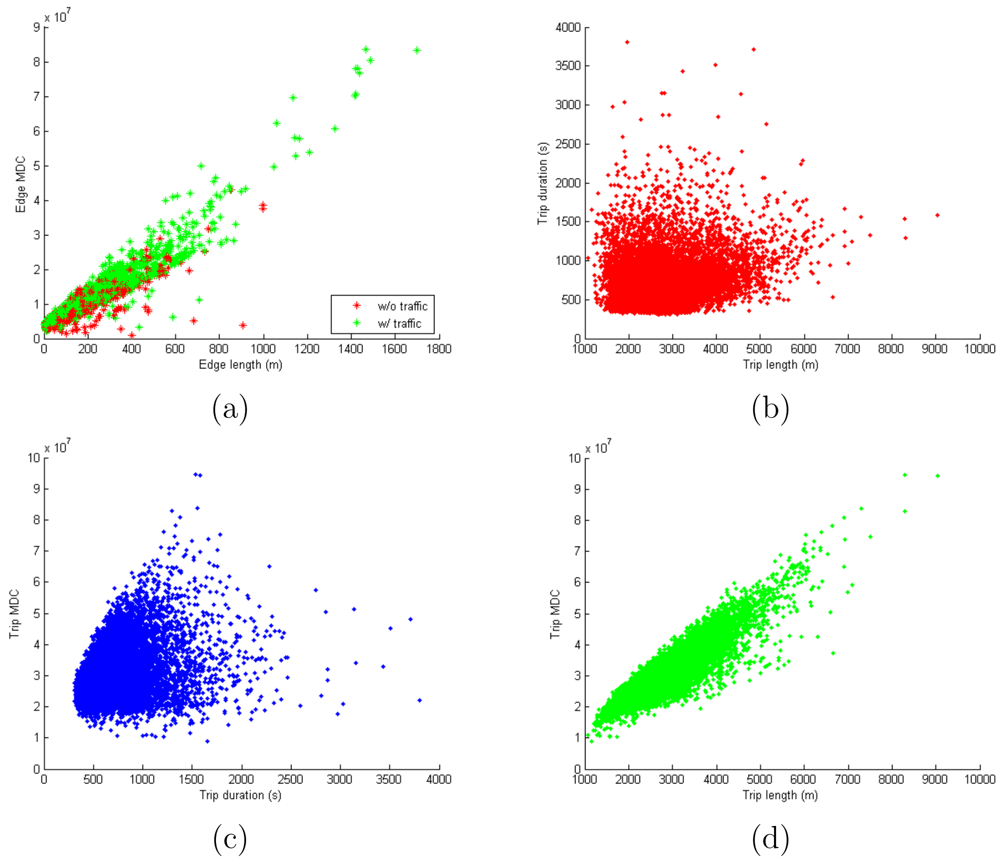


Figure 5.6. Illustration of the distributions of: (a) edge length-to-MDC, (b) trip length-to-duration, (c) trip duration-to-MDC and (d) trip length-to-MDC.

all the order/trip records in the dataset, a filtering process is applied to filter out short trips (both duration-wise and distance-wise), which the thresholds are set at 60 seconds and $1,000$ meters. Also, to reduce the load of map-matching, the trajectory points are down sampled at a minimum distance of 5 meters to reduce the redundant points. The heat map of pick-up time is illustrated in Figure 5.7, which shows the trips in this dataset temporally distribute as our empirical knowledge.

	# of drivers	# of trips	Median duration (s)	Median distance (s)	# of edges
Raw	523,881	3,069,317	414	2,052	NA
MM	NA	433,119	397	2,035	12
Filtered	NA	113,976	610	2,639	14

Table 5.2. Dataset statistics: number of drivers and trips, median trip duration, distance and MM edges of the raw dataset, map-matched (MM) set and the final filtered set. Note, driver information is discarded in processing stage.

Aerious map-matching is prerequisite. After testing out several solutions, we select Fast Map-Matching (FMM) [**225, 16**] – a hidden Markov model based solution with pre-computation of an upper bounded origin-destination hash table for acceleration purpose – due to its high accuracy, speed and accessibility. The configurations of FMM are 8 nearest neighbours (edges), $3 \times 10^{-3\circ}$ search radius (approximately 300 meters) and a GPS error of $5 \times 10^{-4\circ}$ (approximately 50 meters).

Table 5.2 shows the statistics of the trips at each step.

5.4.2.5. Trip MDC. To generate the simulated MDC for each trip, some configurations need to be "hard-coded" such as search range d and resolution $r \in R$ at different vehicle internal parameters (motion) and external factors combinations at a certain moment. Ideally, sensors (and hardware) are inclined to keep a high/consistent acquisition rate and quality to ensure driving safety. Unfortunately, due to hardware limitations, a trade-off between sampling rate (maintaining sampling rate and lower the data quality/resolution) and data quality (keep data resolution and drop frames)

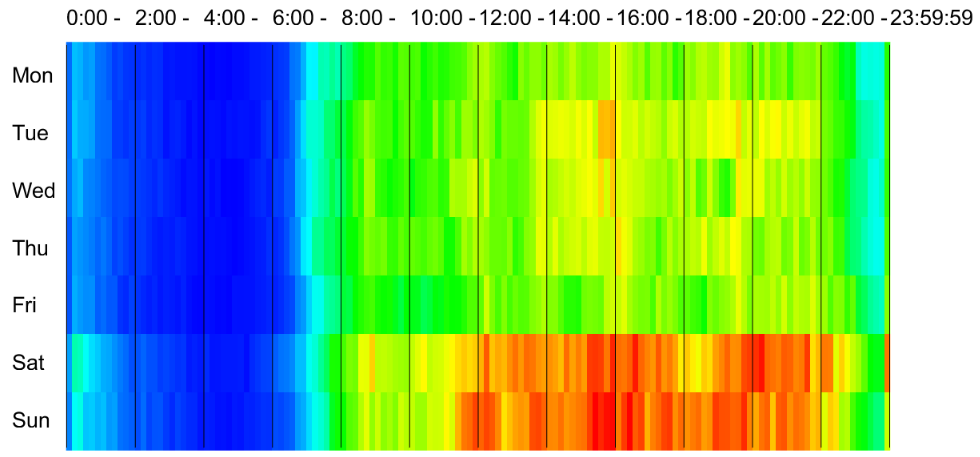


Figure 5.7. Illustration of trip starting times heat map. Each row represents day of the week, while each column represents the 2 hours time window from 0:00 to 23:59:59 (further split in 10min. intervals (shaded)).

rises [**221, 10**]. Most solutions tend to be the first solution to fulfill AV's reaction time requirement [**43**].

In DiDi's trip dataset [**38**], the only motion information recorded is the velocity of each trajectory point. Thus, for each trajectory point's velocity c, we define two thresholds $\Gamma_1 = 5m/s$ and $\Gamma_2 = 10m/s$ to determine different d, r combinations into three segments as searching criteria [4]. $d_{1,2,3} = 1, 3, 5$ and $r_{1,2,3} = 13, 12, 11$ [5].Given a map-matched trip L_t converted from dense enough trip L, the construction algorithm of trip MDC is shown in Algorithm 1. Figure 5.6(b), (c) and (d) illustrates three distributions of the trip duration, distance and MDC combinations. Note, due to

[4]The reason we use these two thresholds is the distribution of vehicle speed (at certain time interval) learned from the dataset follows normal distribution with $\mu \approx 8m/s$.
[5]Tile level 13 equivalents to a $\approx 10^{-1}$ meter voxel size, $12 \longrightarrow 2 \times 10^{-1}$, and so forth.

a filtering (to number of trajectory points) process applied, a hard cut-off at the bottom can be observed in both length-to-duration and length-to-MDC plots.

Algorithm 1: Trip MDC Generation Process

```
Input:
HD Maps: $M$;
Map-Matched Trip in tile representation: $L_t = \{< \hat{x}_k, \hat{y}_k, \hat{c}_k >\}, k \in [1, |L_t|]$;
Threshold: $\Gamma_1, \Gamma_2$;
Search window size and resolution: $d_{1,2,3}, r_{1,2,3}$;
Result:
MDC of a trip: $\mathbf{MDC}_L$
Initialize a set $B$ to store the HD maps tile IDs;
Initialize $MDC_L = 0$;
for $< \hat{x}_k, \hat{y}_k, \hat{c}_k > in\ L_t$ do
    if $c_k \leq \Gamma_1$ then
        $B_{temp} = \{f_p(< x, y >) + < p, q >, r\}$,
        $p \in [-d_1, d_1], q \in [-d_1, d_1], r = r_1$;
    end
    else if $\Gamma_1 < c_k \leq \Gamma_2$ then
        $B_{temp} = \{f_p(< x, y >) + < p, q >, r\}$,
        $p \in [-d_2, d_2], q \in [-d_2, d_2], r = r_2$;
    end
    else
        $B_{temp} = \{f_p(< x, y >) + < p, q >, r\}$,
        $p \in [-d_3, d_3], q \in [-d_3, d_3], r = r_3$;
    end
    for $b \in B_{temp}$ do
        if $b \notin B$ then
            $MDC_L += I_b$;
            $B$.append $b$;
        end
    end
end
Return $MDC_L$
```

5.4.3. Naive Feature embedding and variables

Embedding features in graph representation can easily cause the problem of the curse of dimensionality. Since our task focuses on the PMDC on a certain route, while the synthetic HD map data is generated and attached to every unit area on each link, we naturally concatenate the features of following route after local features. Theoretically, we can concatenate all the information of the rest of trajectory points to form an extremely long feature vector for each trajectory point. However, only a certain length of future information will used in our experiment, to ensure that the information is sufficient for the PMDC purpose (while not being redundant). In our dataset, the maximum time interval of two adjacent trajectory points is 6 seconds because, as mentioned in Section 5.4.2, this is a criteria of our raw data filtering process. This leads to the longest distance interval is equivalent to 18 tiles. For a safety margin, we hard code the number of tiles being used in our PMDC experiments to 20. Given the time interval $t_{j+1} - t_j$ from current point to its next point, and vehicle velocities (2D) and headings (2D) in both raw GPS data and MM data, we have our $\|a_j\| = 20 + 9 = 29$. Next, b_j consists of the traffic speed, TTI and link category information of the next 20 tiles, which makes $\|b_j\| = 3 \times 20 = 60$. We note that this is a naive road category representation: if the road links to an intersection in this tile, label it as 1; otherwise, label it as 0. Lastly, the trip global time-wise and distance-wise progress (in percentage) are added to form the g_j, where $\|g_j\| = 2$. Hence, three features combinations $v_j = \{b_j|\langle b_j, a_j\rangle|\langle b_j, a_j, g_j\rangle\}$ are used in our experiments have the lengths of 29, $29 + 60 = 89$, and $29 + 60 + 2 = 91$, respectively.

5.5. Naive LSTM Solution

We now proceed with explaining a simple (preliminary) solution.

Firstly, we note that in our case, the HD maps are represented in higher resolution tiles. When it comes to the auxiliary data, it includes: (a) Vehicle motion measurements, including vehicle velocity (2-D), accelerations (2-D) and location. (b) External and real-time information, not directly a factor to MDC size, including traffic speed and traffic indices of each link. (c) Trip global information, which records the progress (in percentage) of a trip in time-wise and distance-wise.

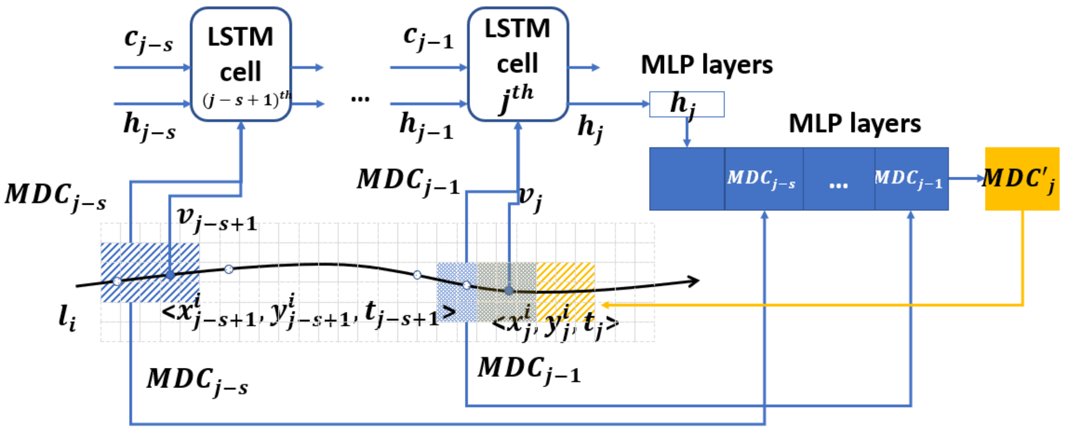


Figure 5.8. Vehicle features at j^{th} trajectory point (blue dot) and MDC (blue tiles), combined with previous $j - s + 1$ historical features and MDCs are fed into the LSTM, to predict PMDC for the next distance (yellow tiles).

Let b_j, a_j, and g_j denote internal, external and global features at j^{th} trajectory point of trip L. More specifically, b_j is a combination of vehicle's motion and surrounding HD maps (B_{l_j}). One may argue that, in theory, using b_j only is sufficient to predict the MDC (i.e., MDC itself is a function of geo-locations and HD maps), which makes the problem roughly equivalent to a speed prediction or next-location prediction problem. Furthermore, a_j is introduced to describe the external and real-time features at j^{th} trajectory point, such as traffic indices, speed and link categories ahead of current geo-location. Last but not least, a trip global information g_j indicates the trip status/progress if the trip destination is known. In our experiments (cf. Section 5.7), three feature combinations will be used: $v_j = \{b_j | \langle b_j, a_j \rangle | \langle b_j, a_j, g_j \rangle \}$.

Given its effectiveness in summarizing the contextual information from sequential data, we utilize LSTM to encode the trajectory knowledge from historical points, and each trajectory point $j \in (0, |L|]$ of a trip L is an LSTM time step. Inspired by the usage of word embeddings and sliding windows in natural language processing studies, we generate the trajectory embeddings at each trajectory point using various addressed featurization techniques and use the embeddings in a moving sliding window to forecast the MDC at the future trajectory point continuously.

LSTM cell at step j of one trajectory path, taking trajectory embeddings v_j $LSTM(h_{j-1}, c_{j-1}, v_j, y_{j-1}))$, is defined as follows:

$$
\begin{aligned}
e_j &= \sigma(W_e[h_{j-1}, c_{j-1}, v_j, y_{j-1}] + d_e) \\
f_j &= \sigma(W_f[h_{j-1}, c_{j-1}, v_j, y_{j-1}] + d_f) \\
o_j &= \sigma(W_o[h_{j-1}, c_{j-1}, v_j, y_{j-1}] + d_o) \\
\tilde{c_j} &= tanh(W_c[h_{j-1}, c_{j-1}, v_j, y_{j-1}] + d_c) \\
c_j &= f_j * c_{j-1} + e_j * \tilde{c_j} \\
h_j &= o_j * tanh(c_j)
\end{aligned} \tag{5.9}
$$

The input, forget, and output gates are e_j, f_j, and o_j respectively; the hidden state h_j indicates the sequential embeddings and c_j represents the contextual embeddings. $\tilde{c_j}$ denotes the intermediate embeddings carried out from input contexts. Weight matrices W_e, W_f, W_o, W_c and bias vectors d_e, d_f, d_o, d_c are shared across different trajectories $\{l_i\}$.

Suppose that the sliding window is of size s. The hidden state of our vanilla multivariate LSTM with sliding windows $LSTM(j)$ is obtained by the following recursive function:

$$h_j = LSTM(h_{n-1}, c_{n-1}, v_n, y_{n-1}), n \in [j - s + 1, j] \tag{5.10}$$

The initial state h_0 is a tensor padded with 0s. Similarly, when $j < s$ the feature embeddings are padded with 0s. We connect h_j with a stack of fully connected layers $\phi(j)$, for generating the predicted MDC value MDC_j' (noting that the objective function is the l_2 loss between MDC'_j and MDC_j):

$$MDC'_j = ([h_j, \{MDC_n\}]) \tag{5.11}$$

5.6. Methodology

In this section, we elaborate each module of the framework. As shown in Figure 5.9, our framework mainly consists of three components: **HD maps encoder**, which encodes the HD maps information of each edge into a fixed-length (d_m) vector; **Road segment encoder** converts each road segment into a d_r-length vector; **MDC simulator** generates MDC for each trip training purpose.

Note a trip will be firstly map-matched and converted into an $|L_g|$-length vector in graph representation (cf. Section 5.3). Traffic information, i.e., road conditions and traffic flow, has already been processed into a graph (for each edge at a specific timestamp, there is a d_t-dimensional vector), hence we do not further encode the traffic information.

5.6.1. MDC Simulator

Since there is no real MDC data acquired by either experimental or consumer vehicles, we create an MDC for each trip follow Equation. 5.1, 5.2, 5.6 and 5.7, with using trip, road network, HD maps and relative hard-coded variables as input. The process is illustrated in Figure 5.9. To reduce redundant description of this process, we elaborate the construction in Section 5.4.2.5. The output of this module is a stand-along MDC value, which is denoted by MDC_L for trip L.

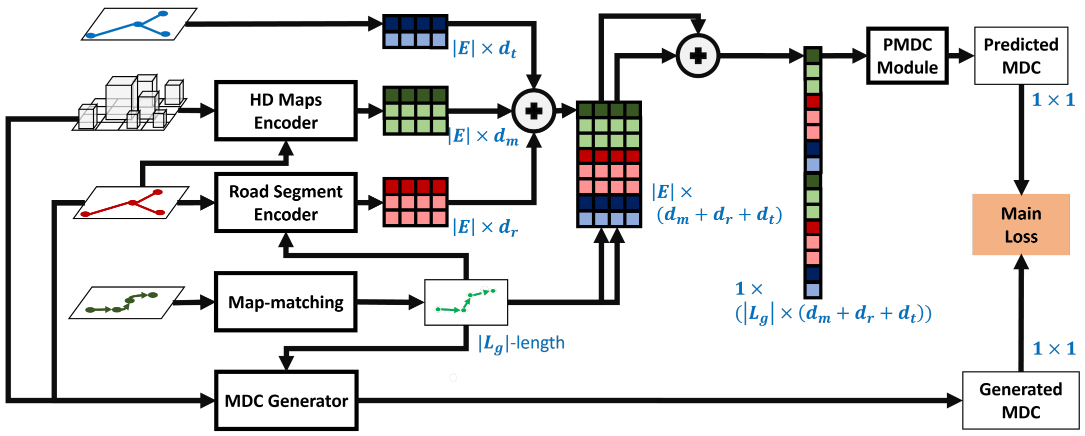


Figure 5.9. The architecture of all modules in our PMDC solution. The dimension (in cyan) is shown next to each output.

5.6.2. Road Segment Embedding

In general, the city road network consists of a set of interconnected road segments. Each of the segments denotes a sample of the physical connectivity. As for a specific trip, its map-matched trajectory can be split into a sequence of road segments sorted by time, and each segment is unique in the whole road network. Thus, road segment can be considered as the meta component of the trip and city road network. Since the explicit knowledge about the underlying interaction has been extracted and saved in the topological graph structure, implicit representations of road segments are necessary for resolving trips. To enhance and ensure the conciseness, we have established the city road network according to historical trip trajectories in Section III. In this section, we introduce an approach for learning the road segment representation and preserving the similarity of neighboring segments in the embedding space.

Given the weighted road graph network $G =< V, E >$, we first use a $|V|$-dimensional one-hot vector $\mathbf{o}_i$ as the initial feature of node $v_i (v_i \in V)$, which attaches a unique representation to each of the nodes. However, one-hot representation can not fully reflect the connectivity of the city road network. For example, the standard similarity calculation – Euclidean distance between any two one-hot embeddings is the same. Spatially adjacent nodes, e.g., neighboring nodes, should be given close embeddings. Inspired by the similarity-preserving network representation methods [**37**], we leverage GraphWave [**45**], i.e., an unsupervised node embedding method, to extract the topological road structure and represent nodes' network neighborhood via a low-dimensional embedding. The process can be defined as,

$$U = \text{GraphWave}(G, X = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{|V|}]), \tag{5.12}$$

where $X = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{|V|}]$ is the input one-hot embedding matrix, G denotes the topological structure of the road network and U represents the learned node embedding vectors.

However, in this study, road segments, i.e., edges E of the graph G, are the key components of each trip that we want to represent. For example, if $e_k \in E$ and $e_k =< u_k, v_k, w_k >$ is a part of a trip, the car will first pass through u_k. Thus, to take the direction of road segments into consideration, we splice two nodes' representation by order with learn node embedding and leverage the concatenation to determine the spatial feature of edge. Specifically, for a given edge e_k, its hidden representation can be defined as,

$$\mathbf{e}_k = \mathbf{u}_k \oplus \mathbf{v}_k, \tag{5.13}$$

where the *oplus* symbol $\oplus$ refers to the concatenation of embeddings. We note that the edge weight w_k has been considered in the node embedding process, i.e., the road network G is weighted, thus we omit it here.

5.6.3. HD maps Embedding

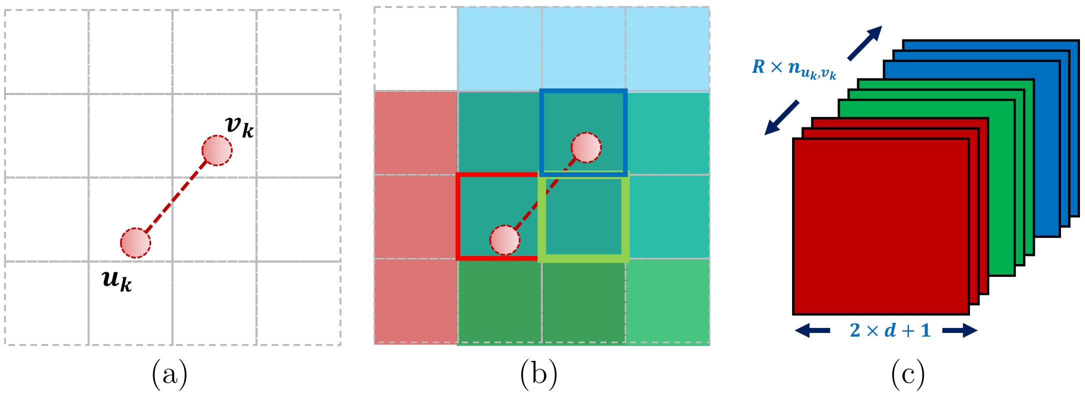


Figure 5.10. Illustration of HD maps information featurization: (a) one road segment $e_k = u_x \rightarrow v_x$, (b) three sequential tiles *red* $\rightarrow$ *green* $\rightarrow$ *blue* belong to e_k, and (c) HD maps information $\mathbf{HD}_{e_k}$ assigned to this edge.

Recall Section 5.3, at tile $< p, q >$, with given search window size d (tiles), the surrounding HD information can be represented as $[M_{p-d \rightarrow p+d, q-d \rightarrow q+d, 1 \rightarrow R}]$, which is a 3-dimensional vector. Assume an edge $e_k \in G$ with start and end points $u_k, v_k \in V$ (shown in Figure 5.10 (a)) has a n_{u_k,v_k}-length sequential footprint $\{< p_{u_k}, q_{u_k} >, \ldots, < p_{v_k}, q_{v_k} >\}$ in HD maps tile coordinate, where $< p_{u_k}, q_{u_k} >= f_p(u_k.x, u_k.y)$

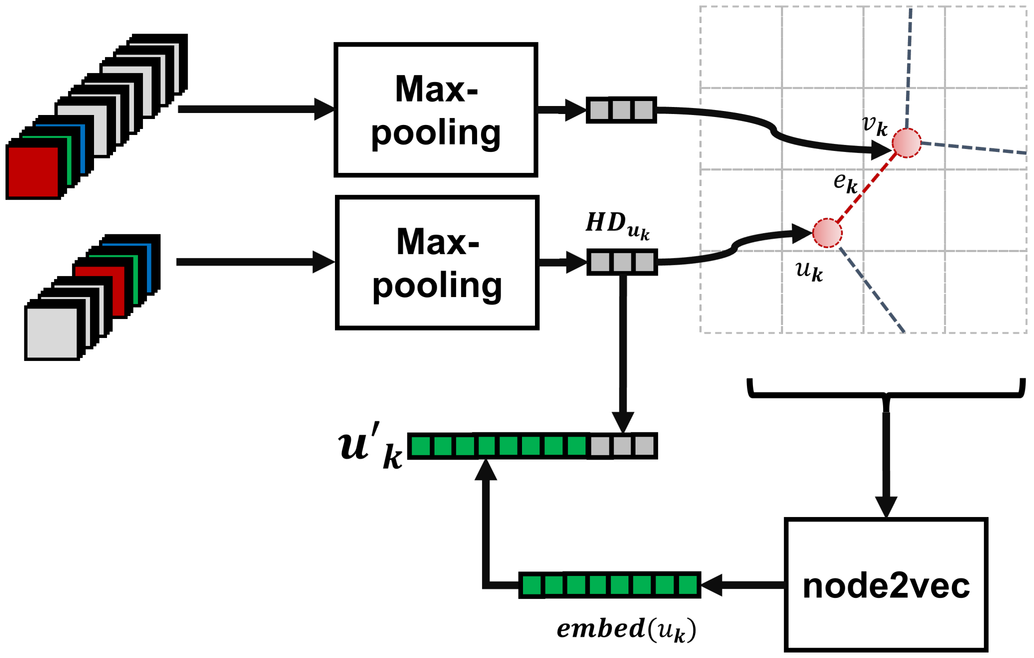


Figure 5.11. Architecture of HD maps encoder.

(illustrated in red → green → blue small boxes in Figure 5.10 (b)). Thus, the HD maps $\mathbf{HD}_{e_k}$ information of edge e_k can be represented as a 4-dimensional vector

$$\begin{aligned}
&\mathbf{HD}_{e_k} = \\
&[[M_{p_{u_k}-d\rightarrow p_{u_k}+d, q_{u_k}-d\rightarrow q_{u_k}+d, 1\rightarrow R}], \ldots, \\
&[M_{p_{v_k}-d\rightarrow p_{v_k}+d, q_{v_k}-d\rightarrow q_{v_k}+d, 1\rightarrow R}]], \\
&\mathbf{HD}_{e_k} \in \mathbb{R}^{(2d+1)\times(2d+1)\times R\times n_{u_k,v_k}}
\end{aligned} \tag{5.14}$$

Where n_{u_k,v_k} denotes the distance from u_k to v_k in tile coordinate.

The purpose is to extract a fix-sized feature vector for each edge using the associated HD maps features and the graph connections. We utilize the nodes and edges in all trips to build a graph neural network and apply max-pooling to the last dimension (across its batches) of the HD maps associated with a node to obtain the HD maps features for each node $\mathbf{HD}_{u_k}$. To be specific, given a set of edges $\{e_j\}$ connect to node u_k, $\mathbf{HD}_{u_k} = \mathbf{Maxpool}([\mathbf{HD}_{e_j}])$.

Inspired by the *word2vec* [**122**] work in natural language processing studies, we create a sliding window among nodes in our trips and maximize the probabilities of two connected nodes being on the same trip. We first generate node embeddings $embed(u_k)$ using skipgram and negative samplings in node2vec [**64**]; we then concatenate the node embeddings and the pooled HD maps features $\mathbf{u}' = [embed(u_k), \mathbf{HD}_{u_k}]$ via Hadamard transform. Similar to process being used in previous section (cf. Equation 5.13), the final HD maps embeddings $\mathbf{e}'_k$ for edge e_k can calculated as $\mathbf{e}'_k = \mathbf{u}'_k \oplus \mathbf{v}'_k$ with a dimension of d_m. The architecture of HD maps encoder is illustrated in Figure 5.11.

5.6.4. PMDC

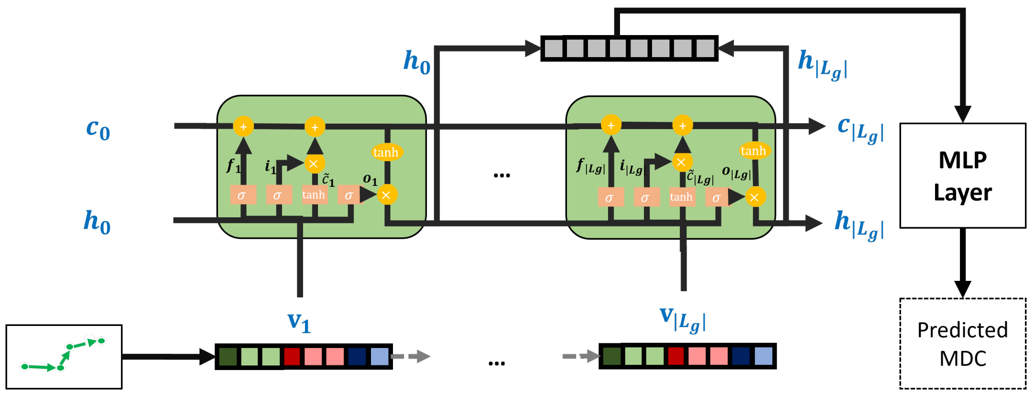


Figure 5.12. Architecture of PMDC module.

Aforementioned, a trip L has an equivalent representation in graph which is $L_g = \{<u_j, \tau_j, \epsilon_j>\}$, and this representation can be further converted in to a sequential sets of adjacent edges which denoted as $P = <<u_1, u_2>, \dots, <u_{J-1}, u_J> = <e_1, e_2, \dots e_{J-1}>>$, where $J = |L_g| - 1$. The output of **Road Network Encoder** and **HD Maps Encoder** are denoted as $\mathbf{e}$ and $\mathbf{e}'$, and the road network traffic information directly pulled from the dataset is $\mathbf{t}$. For each $e_i \in P$, we construct a fixed-length vector $\mathbf{v_i}$ by concatenating each embedding of this edge to get the final embedding $\mathbf{v_i} = [\mathbf{e}_i, \mathbf{e}'_i, \mathbf{t}_i]$.

Given its effectiveness in summarizing the contextual information from sequential data, we utilize LSTM to encode the trajectory knowledge into a fixed-length vector from historical segments, and each road segment e_j of a trip L_g is an LSTM time step $LSTM(i)$ defined by

$$\begin{aligned}
e_j &= \sigma(W_e[h_{j-1}, \mathbf{v}_j] + d_e),\\
f_j &= \sigma(W_f[h_{j-1}, \mathbf{v}_j] + d_f),\\
o_j &= \sigma(W_o[h_{j-1}, \mathbf{v}_j] + d_o),\\
\tilde{c_j} &= tanh(W_c[h_{j-1}, \mathbf{v}_j] + d_c),\\
c_j &= f_j \otimes c_{j-1} + e_j \otimes \tilde{c_j},\\
h_j &= o_j \otimes tanh(c_j).
\end{aligned} \tag{5.15}$$

The input, forget, and output gates are e_j, f_j, and o_j, respectively, which represent how much information we extract from the current input, save from the previous hidden state, and keep in current output. The hidden state h_j indicates the sequential embeddings, and c_j represents the contextual embeddings. $\tilde{c_j}$ denotes the intermediate embeddings carried out from input contexts. Weight matrices W_e, W_f, W_o, W_c and bias vectors d_e, d_f, d_o, d_c are shared across different trips. The initial hidden vector h_0 is a vector of zeros.

$$M\hat{D}C = W_f h_{J+1} + b_f. \tag{5.16}$$

We utilize the hidden state of the final step h_{J+1} to embed one complete trip and append one multilayer perceptron (MLP) layer to obtain the predicted MDC value $M\hat{D}C$; we use the standard mean squared error (MSE) as the loss function. Equation 5.16 shows the prediction function, where W_f and b_f are trainable weights for converting dimension $|h_{J+1}|$ to 1.

5.7. Experimental Results

5.7.1. Portion PMDC with Naive LSTM Solution

We now present the empirical results and discuss the impacts of different step lengths and features being used, as well as training models (LSTM, and RNN).

Note that, in the sequel we use a 2-character abbreviation to denote experiment setup combination. The first character is numerical and denotes the number of steps; the second character denotes the type of features being used: B for basic internal feature ($v_j = b_j$), A for external features ($v_j = \langle b_j, a_j \rangle$) and T for global features added ($v_j = \langle b_j, a_j, g_j \rangle$), where $\|b_j\|$, $\|a_j\|$ and $\|g_j\|$ are 29, 60 and 2 respectively. For example, **5A** stands for the experiment setup with using 5 steps and external features. The learning models used in our experiments share the same configurations: the dimentionality and batch size are 256 and 32, respectively; the dropout between fully connected layers is 0.2 with using ReLU as the activation function; and the learning rate is $5e^{-6}$.

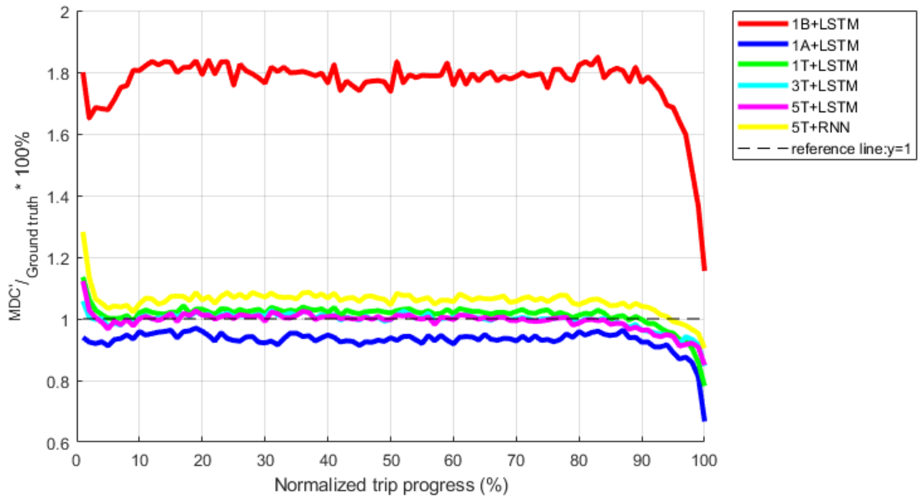


Figure 5.13. Prediction performances of selected training setups. Each plot is normalized to 100 poses and presenting the median performance of entire test set of each pose.

5.7.1.1. Empirical Results. In the sequel, we compare the performance among various features and model combinations, and show plots of results in naive metric in Figure 5.13. The plots are generated using the entire test set, and all trips are normalized into a 100-pose trip for a better visualization. We note that in our trip processing step, we have already filtered out all the trips with fewer than 100 poses.

LSTM						**RNN**		
	MAPE	**MSE**		**MAPE**	**MSE**		**MAPE**	**MSE**
1B	1.07	1.53	**5B**	0.318	0.379	**5B**	0.326	0.381
1A	0.38	0.43	**5A**	0.294	0.372	**5A**	0.331	0.376
1T	0.33	0.399	**5T**	0.288	0.369	**5T**	0.335	0.376

Table 5.3. Selected MAPEs and MSEs of different feature, step and model combinations.

As mentioned in Section 5.4, the MDC generated from our SCD is a function of vehicle's location and HD maps (only). Using the most basic features is sufficient to predict the MDC of each pose. According to Figure 5.13, with adding external features (a_j), the prediction performance is significantly improved. Another observation is, a huge performance drop appears at the end of a trip, no matter which combination of $v_j = b_j$ or $v_j = \langle b_j, a_j \rangle$, steps and learning model is being used. The feature ($v_j = \langle b_j, a_j, v_j \rangle$) introduced in previous sections can significantly fix the performance drop. The experiment also shows a longer historical steps leads to a higher prediction performance.

5.7.1.2. Metrics and Result Analysis. To systematically evaluate the prediction results of a time series regression problem, assessment criteria such as Mean Absolute Error (MAE), Percentage Error (PE), Scaled Error (SE) and their variants are widely used [**171**]. Compared to MAE (and its variants) – a scale-dependent error metric, which has advantages in evaluating single time series or multiple time series with the same units in similar orders of magnitudes, and SE-like evaluation metrics – the metrics show advantages in comparing accuracy across series with different units, PE-like evaluations are more suitable in our case [**201**]. Because the MDCs across the series are with same unit, but the orders of magnitudes are significantly affected by different locations, vehicle motion and configuration – sometime they cover over three orders of magnitude. Define error of the MDC prediction at j^{th} trajectory point (of a trip l_i) as:

$$e_j = MDC'_j - MDC_j, \tag{5.17}$$

where MDC'_j denotes the MDC prediction value at trajectory point $j \in [1, L_i - 1]$. We note that, ideally, at the end of a trip, $MDC'_j = 0$, and MDC_{j+1} does not exist, so we will ignore this term because we use PE-like measures. Given PE at $p_j = 100 \times e_j / MDC_j$, and a trip $l_i \in T$ where T is our test set, the Mean Absolute Percentage Error (MAPE) can be defined as $\mathbf{MAPE}_j = mean(|p_j|), j \in [1, L_i - 1]$. A detail result of selected experiment setups is presented in Table 5.3.

5.7.2. PMDC

In this section, we compare our result to several baseline approaches being used in similar tasks and discuss the impacts/effectiveness of each encoder in our framework. The full length of embedding is $d_v = |\mathbf{v}| = |[\mathbf{e}, \mathbf{e}', \mathbf{t}]|$, where each component has a length of $d_m = 100$, $d_r = 128$ and $d_t = 2$.

5.7.2.1. Baseline. Due to the novelty of the PMDC, there are no approaches that we are aware of that can be categorized as related ones. Hence, for complementary perspectives, we use the following approaches as baselines:

LSTM. ([**235**]) an LSTM based solution that concatenates all the sequential HD maps tile information along a given trip, combined with related internal and external features such as velocity and traffic. This pipeline is extremely expensive (both computation-wise and storage-wise) when encodes each trip since the HD maps are represented in their raw format. No neighbor (global) information is encoded.

Linear Regression. ([**217**]) a Linear Regression (LR) model is trained to minimize the loss (Euclidean distance) between predict MDC and true MDC to solve the PMDC. The complexity of building such feature vector (and normalized to a fixed-length vector) is the same as it in the LSTM pipeline. One of the significant drawbacks of this type of solution is the lack of representing both sequential spatial and temporal information.

DeepOD. ([**227**]) Deep Origin-Destination is a neural network based solution that learns and encodes both spatial and temporal properties from adjacent edges and all given trips to represent the current edge. The final travel estimation model is trained by concatenating each sequential edge from different trips and a Multilayer perceptron (MLP). Note this work aims to solve the ETA problem and is irrelevant to our case. Hence, to make a fair comparison, we add the HD maps information into the training process as a part of the embedding.

5.7.2.2. Evaluation Metrics. To systematically evaluate the performance of PMDC, popular assessment criteria such as Mean Absolute Error (MAE), Mean Absolute Percent Error (MAPE) are used in this chapter. Let MDC_L and $M\hat{D}C_L$ denote the ground truth MDC and predicted MDC of a trip $L \in$ dataset S, the MAE and

MAPE of the entire set can be computed as,

$$\begin{aligned} MAE_S &= \frac{1}{|S|}\sum_{L\in S}|MDC_L - \widehat{MDC}_L|, \\ MAPE_S &= \frac{1}{|S|}\sum_{L\in S}\frac{|MDC_L - \widehat{MDC}_L|}{MDC_L}. \end{aligned} \tag{5.18}$$

To reflect the performances on the entire dataset consisting of the varying length trips, we also bring in weighted MAPE (wMAPE) in the evaluation, where the weight is the travel distance of the trip:

$$wMAPE_S = \sum_{L\in S}\frac{\textit{length of } L}{\textit{total length of } S}\frac{|MDC_L - \widehat{MDC}_L|}{MDC_L} \tag{5.19}$$

5.7.2.3. Comparison with Baselines. Firstly, the MAEs, MAPEs and wMAPEs of the experiment results are shown in Table 5.4, and the Probability Density Function (PDF) of MAPEs is illustrated in Figure 5.14.

The first thing to catch our sight is the extremely poor performance of LSTM. Aforementioned, this LSTM framework is borrowed from our previous work, which is designed and optimized for the objective of giving a sequence of embeddings from previous trajectory points within a time window, predicting the MDC for the next time interval. We modify this work by simply expanding the size of time window to the entire trip for predicting the MDC.

Secondly, the LR's performance is impressive. Recall Section 5.4.2.1 and Section 5.4.2.5, the HD maps – or to be precise, I, the number of objects of an HD map tile – are generated using the normal distribution learned from real-world data and applied/attached to Xi'an's road network randomly. In real-world scenario, the distribution of "objects" is not only spatial-wise, but also graph-wise unique, varies from districts/blocks functionalities. For instance, central business districts consist of a higher volume of objects than park districts. Once the voxel distribution is applied to the entire city, the discrimination of number of voxels is eliminated, which can be observed in Figure 5.6 (a). At the same time, the generation of trip MDC follows a straightforward piecewise-defined function with several hard-coded variables, no regularization applied. This drawback is also indirectly reflected in the shape in Figure 5.6 (d).

Even though the synthesizing of HD maps and trip MDCs have such drawbacks, the performance of our proposed model is ahead of LR and the DeepOD. Considering the low margin of performance difference, a 4.17% improvement of MAPE shows the effectiveness of the integration of HD maps encoder and road network traffic information. Once real-world data is used, a dilated performance margin is expected.

The introduction of wMAPE shows whichever module is deployed, the trips with longer travel distance intending to have a worse prediction result in MDC.

5.7.2.4. Effectiveness of Embeddings. Since there are multiple modules integrated in our framework, we disable different modules to evaluate the effectiveness of each embedding. As aforementioned, we have three embeddings: road network, HD maps, and traffic information, which leads to 2^3 experiments. Note, if ether road network embedding or HD maps embedding is disabled, we will use one-hot vector

	MAE ("voxels")	MAPE %	wMAPE %
LR	$7.67e^6$	21.58	22.81
LSTM	$1.82e^7$	53.66	—-
DeepOD	$4.54e^6$	15.25	15.30
Ours	$3.41e^6$	11.08	11.19

Table 5.4. Experiment results on test set with different models. Note experiment LSTM has no wMAPE value assigned because in that experiment, all trip lengths are normalized.

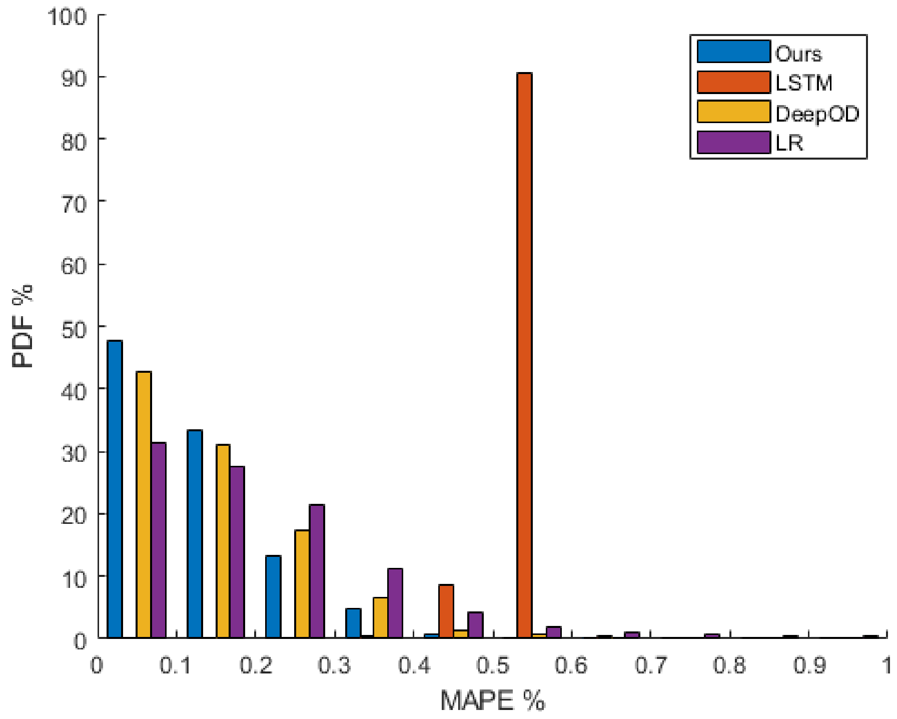


Figure 5.14. Probability Density Functions (PDFs) of MAPES on the test set using different methodologies.

to replace the graph embedding, and the experiments on some embedding combinations are meaningless, which we will ignore. Given the fixed order of "road network, HD maps, traffic", we use a 3-digit abbreviation – $\{111, 110, 101, 100, 010, 011\}$ – to denote each experiment combination. The MAPEs of experiments are shown in Table 5.5 and PDFs are shown in Figure 5.15, from which the following observations are made:

(1) Comparing all the controlled trials of traffic information (i.e., $\{111 \leftrightarrow 110\}$, $\{101 \leftrightarrow 100\}$ and $\{011 \leftrightarrow 010\}$), the integration of such information significantly improves the performance with the combination of HD maps encoding ($11.08\% = 13.34\% - 2.26\%$ and $11.58\% = 14.38\% - 2.8\%$). Once the traffic information stands alone, less improvement can be obtained ($14.22\% = 14.43\% - 0.21\%$);

(2) Both HD maps and road network encoders do not contribute the performance as well as the traffic information when being concatenated independently (i.e., $\{110 \leftrightarrow 100\}$ and $\{110 \leftrightarrow 010\}$). The performances only rise by 1.09% and 1.04% respectively;

(3) The higher wMAPEs (over MAPEs) and Figure 5.16 (a) both indicate the majority of longer (distance-wise) trips have worse prediction results than shorter trips. The common issue where a certain amount of "outliers" appear on the shorter-trip-end can also be found in similar trip property estimation problems [**235, 110**], which is caused by the amplification of features' volatility in a shorter sampling time. The quantified performance drops are $0.11\%, 0.16\%, 0.20\%, 0.35\%, 0.48\%, -0.07\%$ for six experiments respectively, and have positive correlations with their overall performances;

code	MAPE % (change %)	code	MAPE % (change %))
111	11.08 (0.00)	***100***	14.43 (3.35)
110	13.34 (2.26)	***010***	14.38 (3.30)
101	14.22 (3.14)	***011***	11.58 (0.50)
code	**wMAPE^1 % (change %)**	**code**	**wMAPE^1 % (change %))**
111	11.19 (0.00)	***100***	14.78 (3.59)
110	13.50 (2.31)	***010***	14.86 (3.67)
101	14.42 (3.23)	***011***	11.51 (0.32)
code	**wMAPE^2 % (change %)**	**code**	**wMAPE^2 % (change %))**
111	11.35 (0.00)	***100***	14.35 (3.00)
110	13.76 (2.41)	***010***	14.56 (3.21)
101	14.18 (2.83)	***011***	11.81 (0.46)

Table 5.5. MAPE and wMAPEs of effectiveness experiments. **wMAPE^1** and **wMAPE^2** denote the wMAPE use trip distance and duration as weights.

(4) When switch the weights from trip length to trip duration (i.e., $\frac{duration\ of\ L}{total\ duration\ of\ S}$), another observation (from Figure 5.16 (b) and Table 5.5) is the performance difference of each experiment has a negative correlation to its trip duration. The performance drops are $0.27\%, 0.42\%, -0.04\%, -0.08\%, 0.18\%, 0.23\%$ respectively. This result shows a better embedding (with temporal information integrated) is more effected by extending the trip duration.

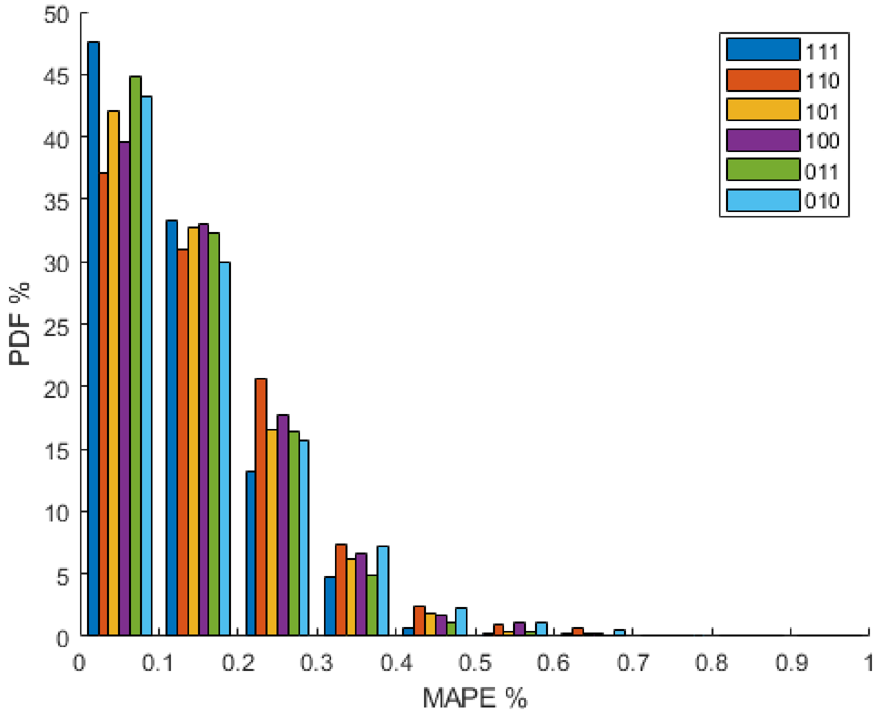


Figure 5.15. Probability Density Functions (PDFs) of MAPEs of effectiveness experiments.

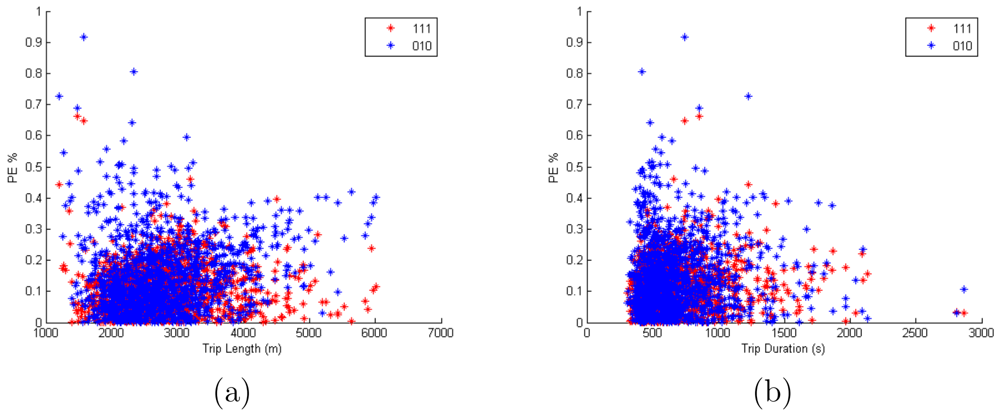


Figure 5.16. Distribution of Percentage Error (PE) to trip length (a) and trip duration (b) of experiments 111 (best performance) and 010 (worst case).

CHAPTER 6

Summary

This dissertation consists of four comprehensive studies in the domain of HD maps that covers the most fundamental knowledge to future data usage prediction problem. We next summarize the findings of each study, and remark on the future directions. The first study (cf. 2) summarizes the HD maps furniture, data structure, applications and challenges. As autonomous driving is becoming a reality, the demand for efficient systems to handle large volumes of complex data on the fly is a paramount. We reviewed the importance and roles of 3D objects – a type of data that comes from heterogeneous sources – in HD maps being used in autonomous driving applications, such as self-localization. We reviewed the state-of-the-art data structures, representations and coordinates used for storing 3D objects, and their potential improvements. We described an end-to-end pipeline of the system and emphasize the challenges and feasible solutions to each part of the pipeline. Lastly, we defined the evaluation metrics for each task and introduced the dataset we built for this objective. Further more, this work may help with any future application that needs to sense and process enormous amount of hierarchical data. Future work includes:

- Test state-of-the-art algorithms of the **3D Object Query** challenges, investigate the influence of each potential variable from a single input and certain vehicle configuration, such as moving speed, vehicle angular resolution, sensing range, failure tolerance, memory size, cpu speed and transmission bandwidth. Study the trade-off balancing problem and find the sweet point of the model.
- Design and implement solution that address different access problems that affect algorithms execution, along with comparative evaluation of our proposed system with with state-of-the-art solutions and investigate the optimization trade-off.
- With adding more input data, such as imagery, or making up synthetic input from reverse engineering the 3D objects in HD maps, investigate the problem of **Heterogeneous Data Fusion**, and the trade-offs of heterogeneous data sources on system performance and build a model for our objective.
- Combine **Heterogeneous Data Fusion** model and **3D Object Query** solution together to test out the overall performance of the system for vehicle self-localization.

The second study (cf. 3) focuses on high precision lane boundary geometries extraction technique from very unusual data source – satellite imagery. We demonstrate that our machine learning and pixel-wise segmentation hybrid prototype can achieve optimal performance in modeling HD Maps from reasonable high resolution overhead imagery. With such precise modeling result, our approach can be used as a good complementary data source for HD Maps modeling to either validate accuracy or complement the missing data caused by object occlusion. We designed a persuasive road model evaluation metric and published our HD road model dataset that can aid future research in this topic.

There are still many limitations to our solution. Thus, we are looking to improve in these areas:

- Unlike HD Map modeling from LiDAR point cloud, image based road modeling, especially from overhead imagery, has an apparent drawback: the lack of elevation data. This, however, can be solved by the High-Definition Digital Elevation Model (HD DEM) database [**230**].
- The second challenge is the fusion between image-based HD maps and point-cloud-based HD maps. As we mentioned in the introduction, point-cloud-based HD maps have an edge in accuracy, while the advantage of aerial-based HD maps is that it can remedy the LiDAR shadow problem. Thus, the alignment between these two map sources is very useful and important. Some feature points could identify the transformation, e.g. the end point of a dashed line and gore point.
- Aside from highways and expressways, there is still a huge number of road networks that need to be modeled for autonomous driving and other purposes. Most of the principal and minor arterial are not occluded by buildings and trees, which means our approach can be extended to model road networks in these regions.

In the third study (cf. 4), we build the first full HD maps dataset consists of lane boundaries, 3D objects and complimentary objects such as signs. The main purpose of this work is to describe this dataset that was collected and make it available for future studies. This first version contains a high precision map, consumer-grade GPS and imagery, and related ground truth data. Some natural extensions of this dataset include:

- Real-time 3D information. For example, a real-time LiDAR or depth image sequence can be added for better localization performance and flexibility.
- Change occupancy grid voxel representation to a compressed format such as the Sparse Voxel OcTree, which is a lossless representation with a significant reduction in data-size.
- Evaluation: a dataset toolbox which enables convenient data parsing, coordinate transformations, and visualization. A website which allows researcher download dataset directly, view benchmarks and logs. Finally, a real-time evaluation server which boosts evaluation cycle and increases the algorithm persuasion.
- Dataset coverage could be extended for large scale tests.

Our groundbreaking research is introduced in Chpter 5, which proposes a novel problem (PMDC) for future real-time HD maps application. We proposed a comprehensive deep learning approach and neural network architecture, which is able to not only exploit historical trips, but also encode the HD maps of each edge into a fixed-length embedding and trained with a graph-based neural network instead of a naive one-hot encoding or average pooling. As our experiments have demonstrated, our model achieved superior performance over the conventional LR, LSTM and modified GNN based solutions (designed for similar objectives). A detailed study on module effectiveness not only shows the contribution of each encoder, but also indicates the robustness/consistency to trip length and duration. A future model can tweak the embeddings based on its objective (e.g., prediction preferences).

Since, to our knowledge, this is a forerunner work for the MDC problem, we recognize certain, at this point unavoidable, limitations. One notable limitation is the lack of real-world data. Both SCD (especially HD maps) and the MDC of each trip are constructed by following straightforward algorithms. Specifically, to generate the MDC, vehicle speed is the only input/factor that affects two variables: search window size and resolution. We believe that numerous other factors (both internal and external) from multiple software and hardware component, would create a more sophisticated variant of the MDC problem and enable more realistic settings to be tackled. One feasible solution in the short run is generating the map data and acquiring real-time vehicle configurations from 3D AV simulation platforms.

A specific variant of the problem that we want to address in the future is the one which would enable incorporation of the impact of (partial) HD maps updates. Such cases occur in the settings in which a new, possibly long-lasting, construction project has started, affecting larger metropolitan area. We are planning to investigate their efficient propagation into the corresponding embedding.

In summary, we are eager to see more achievements and accomplishments in domain of HD maps. We will keep our passion in this area to help the academia and industry to achieve high-level autonomous driving for the foreseeable future, and extend our knowledge to other related fields such as autonomous agriculture and drone logistics.

References

[1] Administration, F. H. Highway statistics 2013, 2013.

[2] Ahn, H.-K., Mamoulis, N., and Wong, H. M. A survey on multidimensional access methods.

[3] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. Social lstm: Human trajectory prediction in crowded spaces. In *IEEE CVPR* (2016), pp. 961–971.

[4] Atanacio-Jiménez, G., González-Barbosa, J.-J., Hurtado-Ramos, J. B., Ornelas-Rodríguez, F. J., Jiménez-Hernández, H., García-Ramirez, T., and González-Barbosa, R. Lidar velodyne hdl-64e calibration using pattern planes. *International Journal of Advanced Robotic Systems 8*, 5 (2011), 59.

[5] Badino, H., Franke, U., and Mester, R. Free space computation using stochastic occupancy grids and dynamic programming. In *Workshop on Dynamical Vision, ICCV, Rio de Janeiro, Brazil* (2007), vol. 20, Citeseer.

[6] Bajcsy, R., and Tavakoli, M. Computer recognition of roads from satellite pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, 9 (1976), 623–637.

[7] Bansal, M., Daniilidis, K., and Sawhney, H. Ultrawide baseline facade matching for geo-localization. In *Large-Scale Visual Geo-Localization.* Springer, 2016, pp. 77–98.

[8] Bansal, M., Sawhney, H. S., Cheng, H., and Daniilidis, K. Geo-localization of street views with aerial image databases. In *Proceedings of the 19th ACM international conference on Multimedia* (2011), ACM, pp. 1125–1128.

[9] Barsan, I. A., Wang, S., Pokrovsky, A., and Urtasun, R. Learning to localize using a lidar intensity map. In *CoRL* (2018), pp. 605–616.

[10] Basu, C., Yang, Q., Hungerman, D., Sinahal, M., and Draqan, A. D. Do you want your autonomous car to drive like you? In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI* (2017), IEEE, pp. 417–425.

[11] Bauer, S., Alkhorshid, Y., and Wanielik, G. Using high-definition maps for precise urban vehicle localization. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on* (2016), IEEE, pp. 492–497.

[12] Bertsimas, D., Delarue, A., Jaillet, P., and Martin, S. Travel time estimation in the age of big data. *Operations Research 67*, 2 (2019), 498–515.

[13] Bierlaire, M., Chen, J., and Newman, J. A probabilistic map matching method for smartphone gps data. *Transportation Research Part C: Emerging Technologies 26* (2013), 78–98.

[14] Bing. Bing maps tile system, 2012.

[15] Bittel, S., Kaiser, V., Teichmann, M., and Thoma, M. Pixel-wise segmentation of street with neural networks. *arXiv preprint arXiv:1511.00513* (2015).

[16] Boeing, G. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems 65* (2017), 126–139.

[17] Bracciale, L., Bonola, M., Loreti, P., Bianchi, G., Amici, R., and Rabuffi, A. CRAWDAD dataset roma/taxi (v. 2014-07-17). `https://crawdad.org/roma/taxi/20140717`, July 2014.

[18] Brenner, C. Extraction of features from mobile laser scanning data for future driver assistance systems. In *Advances in GIScience*. Springer, 2009, pp. 25–42.

[19] Brenner, C. Global localization of vehicles using local pole patterns. In *Joint Pattern Recognition Symposium* (2009), Springer, pp. 61–70.

[20] Brenner, C. Vehicle localization using landmarks obtained by a lidar mobile mapping system. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences:[PCV 2010-Photogrammetric Computer Vision And Image Analysis, Pt I] 38 (2010), Nr. Part 3A 38*, Part 3A (2010), 139–144.

[21] Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles 2*, 3 (2017), 194–220.

[22] Bureau, U. S. C. Urban and rural, 2010.

[23] Byers, C., and Woo, A. 3d data visualization: The advantages of volume graphics and big data to support geologic interpretation. *Interpretation 3*, 3 (2015), SX29–SX39.

[24] Cham, T.-J., Ciptadi, A., Tan, W.-C., Pham, M.-T., and Chia, L.-T. Estimating camera pose from a single urban ground-view omnidirectional image and a 2d building outline map. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 366–373.

[25] Chaudhuri, D., Kushwaha, N., and Samal, A. Semi-automated road detection from high resolution satellite images by directional morphological enhancement and segmentation techniques. *IEEE journal of selected topics in applied earth observations and remote sensing 5*, 5 (2012), 1538–1544.

[26] Chen, C.-Y., and Grauman, K. Clues from the beaten path: Location estimation with bursty sequences of tourist photos. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 1569–1576.

[27] Chen, D. M., Baatz, G., K@oser, K., Tsai, S. S., Vedantham, R., Pylv@an@ainen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., et al. City-scale landmark identification on mobile devices. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 737–744.

[28] Chen, X. Detecting common geographic features in images based on invariant components, 2015. US Patent 9,129,163.

[29] Chen, X. Hd live maps for automated driving: an ai approach. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2018), pp. 1–1.

[30] Chen, X., Hui, R. B., and Alwar, N. Determining a geometric parameter from a single image, 2014. US Patent 8,896,686.

[31] Chen, X., and Yang, X. Detecting ground geographic features in images based on invariant components, 2015. US Patent 8,953,838.

[32] Chen, X., and ZHUKOV, V. Detecting road condition changes from probe data, 2016. US Patent 9,361,797.

[33] Coeurjolly, D., Miguet, S., and Tougne, L. 2d and 3d visibility in discrete geometry: an application to discrete geodesic paths. *Pattern Recognition Letters 25*, 5 (2004), 561–570.

[34] Cohen, J. D., Aliaga, D. G., and Zhang, W. Hybrid simplification: combining multi-resolution polygon and point rendering. In *Proceedings of the Conference on Visualization'01* (2001), IEEE Computer Society, pp. 37–44.

[35] Costea, D., and Leordeanu, M. Aerial image geolocalization from recognition and matching of roads and intersections. *arXiv preprint arXiv:1605.08323* (2016).

[36] Cui, D., Xue, J., and Zheng, N. Real-time global localization of robotic cars in lane level via lane marking detection and shape registration. *IEEE Transactions on Intelligent Transportation Systems 17*, 4 (2016), 1039–1050.

[37] Cui, P., Wang, X., Pei, J., and Zhu, W. A survey on network embedding. *IEEE Trans. Knowl. Data Eng. 31*, 5 (2019), 833–852.

[38] Dataset, D. C. G. O. Oct 2016, Xi'an City Second Ring Road Regional Trajectory Data Set. `https://gaia.didichuxing.com`.

[39] Dataset, D. C. G. O. Travel Time Index. `https://gaia.didichuxing.com`.

[40] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence 29*, 6 (2007), 1052–1067.

[41] Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Wiltshire, B., et al. Eta prediction with graph neural networks in google maps. *arXiv preprint arXiv:2108.11482* (2021).

[42] Deusch, H., Nuss, D., Konrad, P., Konrad, M., Fritzsche, M., and Dietmayer, K. Improving localization in digital maps with grid maps. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on* (2013), IEEE, pp. 1522–1527.

[43] Dixit, V. V., Chand, S., and Nair, D. J. Autonomous vehicles: disengagements, accidents and reaction times. *PLoS one 11*, 12 (2016), e0168054.

[44] Doersch, C., Singh, S., Gupta, A., Sivic, J., and Efros, A. What makes paris look like paris? *ACM Transactions on Graphics 31*, 4 (2012).

[45] Donnat, C., Zitnik, M., Hallac, D., and Leskovec, J. Learning structural node embeddings via diffusion wavelets. In *SIGKDD* (2018), pp. 1320–1329.

[46] Drost, B., Ulrich, M., Navab, N., and Ilic, S. Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), Ieee, pp. 998–1005.

[47] Duan, Y., Yisheng, L., and Wang, F.-Y. Travel time prediction with lstm neural network. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)* (2016), IEEE, pp. 1053–1058.

[48] DuenasArana, G., Joerger, M., and Spenko, M. Local nearest neighbor integrity risk evaluation for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 2328–2333.

[49] DuenasArana, G., Joerger, M., and Spenko, M. Local nearest neighbor integrity risk evaluation for robot navigation. pp. 2328–2333.

[50] Eckart, B. *Compact Generative Models of Point Cloud Data for 3D Perception.* PhD thesis, Carnegie Mellon University Pittsburgh, 2017.

[51] Einhorn, E., Schröter, C., and Gross, H.-M. Monocular scene reconstruction for reliable obstacle detection and robot navigation. In *ECMR* (2009), pp. 7–12.

[52] Enzweiler, M., Greiner, P., Kn@oppel, C., and Franke, U. Towards multi-cue urban curb recognition. In *Intelligent Vehicles Symposium (IV), 2013 IEEE* (2013), IEEE, pp. 902–907.

[53] Fang, Q., Sang, J., and Xu, C. Discovering geo-informative attributes for location recognition and exploration. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 11*, 1s (2014), 19.

[54] Filgueira, A., González-Jorge, H., Lagüela, S., Díaz-Vilariño, L., and Arias, P. Quantifying the influence of rain in lidar performance. *Measurement 95* (2017), 143–148.

[55] Freedman, D. *10 Breakthrough Technologies*. MIT Technology Review, 2017.

[56] Fu, T.-y., and Lee, W.-C. Deepist: Deep image-based spatio-temporal network for travel time estimation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019), pp. 69–78.

[57] Gaikwad, T. D., Asher, Z. D., Liu, K., Huang, M., and Kolmanovsky, I. Vehicle velocity prediction and energy management strategy part 2: Integration of machine learning vehicle velocity prediction with optimal energy management to improve fuel economy. Tech. rep., SAE Technical Paper, 2019.

[58] Gao, J., Murphey, Y. L., and Zhu, H. Multivariate time series prediction of lane changing behavior using deep neural network. *Applied Intelligence 48*, 10 (2018), 3523–3537.

[59] Gao, Q., Zhou, F., Zhang, K., Trajcevski, G., Luo, X., and Zhang, F. Identifying human mobility via trajectory embeddings. In *IJCAI* (2017), vol. 17, pp. 1689–1695.

[60] Gebhardt, S., Payzer, E., Salemann, L., Fettinger, A., Rotenberg, E., and Seher, C. Polygons, point-clouds and voxels: A comparison of high-fidelity terrain representations. In *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation—Processes, Standards, and Lessons Learned* (2009).

[61] Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 3354–3361.

[62] Ghallabi, F., Nashashibi, F., El-Haj-Shhade, G., and Mittet, M.-A. Lidar-based lane marking detection for vehicle positioning in an hd map. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), IEEE, pp. 2209–2214.

[63] Gopalan, R., Hong, T., Shneier, M., and Chellappa, R. A learning approach towards detection and tracking of lane markings. *IEEE Transactions on Intelligent Transportation Systems 13*, 3 (2012), 1088–1098.

[64] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), pp. 855–864.

[65] Gu, X., Zang, A., Huang, X., Tokuta, A., and Chen, X. Fusion of color images and lidar data for lane classification. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2015), ACM, p. 69.

[66] Guenter, B., Finch, M., Drucker, S., Tan, D., and Snyder, J. Foveated 3d graphics. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 164.

[67] Gutmann, J.-S., Burgard, W., Fox, D., and Konolige, K. An experimental comparison of localization methods. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on* (1998), vol. 2, IEEE, pp. 736–743.

[68] Gutmann, J.-S., Weigel, T., and Nebel, B. Fast, accurate, and robust self-localization in polygonal environments. In *IROS* (1999), pp. 1412–1419.

[69] Hartenstein, H., and Laberteaux, K. P. A tutorial survey on vehicular ad hoc networks. *IEEE Commun. Mag. 46*, 6 (2008), 164–171.

[70] Hata, A., and Wolf, D. Road marking detection using lidar reflective intensity data and its application to vehicle localization. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (2014), IEEE, pp. 584–589.

[71] Hata, A. Y., Osorio, F. S., and Wolf, D. F. Robust curb detection and vehicle localization in urban environments. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* (2014), IEEE, pp. 1257–1262.

[72] Hata, A. Y., and Wolf, D. F. Feature detection for vehicle localization in urban environments using a multilayer lidar. *IEEE Transactions on Intelligent Transportation Systems 17*, 2 (2015), 420–429.

[73] Heitz, G., and Koller, D. Learning spatial context: Using stuff to find things. In *European conference on computer vision* (2008), Springer, pp. 30–43.

[74] HERE. Here introduces hd maps for highly automated vehicle testing, 2015.

[75] Higgins, D. W., and Scott, D. M. System and method for synchronizing raster and vector map images, 2007. US Patent 7,161,604.

[76] Hu, J., Razdan, A., Femiani, J. C., Cui, M., and Wonka, P. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE Transactions on Geoscience and Remote Sensing 45*, 12 (2007), 4144–4157.

[77] Hu, Q., Lee, W., and Lee, D. L. A hybrid index technique for power efficient data broadcast. *Distributed Parallel Databases 9*, 2 (2001).

[78] Hugemann, W. Driver reaction times in road traffic. In *Proceedings of XI EVU (European Association for Accident Research and Accident Analysis) Annual Meeting. Portorož, Slovenija* (2002), vol. 32.

[79] Im, J.-H., Im, S.-H., and Jee, G.-I. Vertical corner feature based precise vehicle localization using 3d lidar in urban area. *Sensors 16*, 8 (2016), 1268.

[80] Javanmardi, E., Gu, Y., Javanmardi, M., and Kamijo, S. Autonomous vehicle self-localization based on abstract map and multi-channel lidar in urban area. *IATSS research 43*, 1 (2019), 1–13.

[81] Jeong, J., Cho, Y., and Kim, A. Road-slam: Road marking based slam with lane-level accuracy. In *2017 IEEE Intelligent Vehicles Symposium (IV)* (2017), IEEE, pp. 1736–1473.

[82] Jian, B., and Vemuri, B. C. A robust algorithm for point set registration using mixture of gaussians. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* (2005), vol. 2, IEEE, pp. 1246–1251.

[83] Jian, B., and Vemuri, B. C. Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence 33*, 8 (2011), 1633–1645.

[84] Jiang, K., Yang, D., Liu, C., Zhang, T., and Xiao, Z. A flexible multi-layer map model designed for lane-level route planning in autonomous vehicles. *Engineering 5*, 2 (2019), 305–318.

[85] Jo, K., Jo, Y., Suhr, J. K., Jung, H. G., and Sunwoo, M. Precise localization of an autonomous car based on probabilistic noise models of road surface marker features using multiple cameras. *IEEE Transactions on Intelligent Transportation Systems 16*, 6 (2015), 3377–3392.

[86] Jo, K., Kim, C., and Sunwoo, M. Simultaneous localization and map change update for the high definition map-based autonomous driving car. *Sensors 18*, 9 (2018).

[87] Johansson, G., and Rumar, K. Drivers' brake reaction times. *Human factors 13*, 1 (1971), 23–27.

[88] John, V., Xu, Y., Mita, S., Long, Q., and Liu, Z. Registration of gps and stereo vision for point cloud localization in intelligent vehicles using particle swarm optimization. In *International Conference in Swarm Intelligence* (2017), Springer, pp. 209–217.

[89] Johnson, A. E., and Hebert, M. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 5 (1999), 433–449.

[90] Kesting, A., and Treiber, M. How reaction time, update time, and adaptation time influence the stability of traffic flow. *Computer-Aided Civil and Infrastructure Engineering 23*, 2 (2008), 125–137.

[91] Kim, Z. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems 9*, 1 (2008), 16–26.

[92] Knopp, J., Sivic, J., and Pajdla, T. Avoiding confusing features in place recognition. *Computer Vision–ECCV 2010* (2010), 748–761.

[93] Kong, D., and Wu, F. HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction. In *IJCAI* (2018), vol. 18, pp. 2341–2347.

[94] Kothuri, R. K. V., Ravada, S., and Abugov, D. Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (2002), ACM, pp. 546–557.

[95] Kuru, K., and Khan, W. A framework for the synergistic integration of fully autonomous ground vehicles with smart city. *IEEE Access 9* (2020), 923–948.

[96] Laborda, J., and Moral, M. J. Automotive aftermarket forecast in a changing world: The stakeholders' perceptions boost! *Sustainability 12*, 18 (2020), 7817.

[97] Lategahn, H., and Stiller, C. Vision-only localization. *IEEE Transactions on Intelligent Transportation Systems 15*, 3 (2014), 1246–1257.

[98] Lehtomäki, M., Jaakkola, A., Hyyppä, J., Kukko, A., and Kaartinen, H. Detection of vertical pole-like objects in a road environment using vehicle-based laser scanning data. *Remote Sensing 2*, 3 (2010), 641–664.

[99] Levinson, J., Montemerlo, M., and Thrun, S. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems* (2007), vol. 4, p. 1.

[100] Levinson, J. S. *Automatic laser calibration, mapping, and localization for autonomous vehicles.* Stanford University, 2011.

[101] Li, L., Yang, M., Guo, L., Wang, C., and Wang, B. Precise and reliable localization of intelligent vehicles for safe driving. In *International Conference on Intelligent Autonomous Systems* (2016), Springer, pp. 1103–1115.

[102] Li, L., Yang, M., Wang, C., and Wang, B. Road dna based localization for autonomous vehicles. In *Intelligent Vehicles Symposium (IV), 2016 IEEE* (2016), IEEE, pp. 883–888.

[103] Li, R., Rose, G., and Sarvi, M. Evaluation of speed-based travel time estimation models. *Journal of transportation engineering 132*, 7 (2006), 540–547.

[104] Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., and Liu, Y. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 1695–1704.

[105] Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[106] Lin, T.-Y., Belongie, S., and Hays, J. Cross-view image geolocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 891–898.

[107] Lin, T.-Y., Cui, Y., Belongie, S., and Hays, J. Learning deep representations for ground-to-aerial geolocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 5007–5015.

[108] Linsen, L. *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek Technical Report, Faculty of Computer Science, University of Karlsruhe, 2001.

[109] Liu, H., Ye, Q., Wang, H., Chen, L., and Yang, J. A precise and robust segmentation-based lidar localization system for automated urban driving. *Remote Sensing 11*, 11 (2019), 1348.

[110] Liu, K., Asher, Z., Gong, X., Huang, M., and Kolmanovsky, I. Vehicle velocity prediction and energy management strategy part 1: Deterministic and stochastic vehicle velocity prediction using machine learning. Tech. rep., SAE Technical Paper, 2019.

[111] Liu, R., Wang, J., and Zhang, B. High definition map for automated driving: Overview and analysis. *Journal of Navigation 73*, 2 (2020), 324–341.

[112] Lu, Z., Lv, W., Cao, Y., Xie, Z., Peng, H., and Du, B. Lstm variants meet graph neural networks for road speed prediction. *Neurocomputing 400* (2020), 34–45.

[113] Lu, Z., Lv, W., Xie, Z., Du, B., and Huang, R. Leveraging graph neural network with lstm for traffic speed prediction. In *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (2019), IEEE, pp. 74–81.

[114] Majdik, A. L., Albers-Schoenberg, Y., and Scaramuzza, D. Mav urban localization from google street view data. In *IROS* (2013), IEEE, pp. 3979–3986.

[115] Maps, B. Bing Maps Traffic API. `https://docs.microsoft.com/en-us/bingmaps/rest-services/traffic/`. Online.

[116] Maps, G. Google Maps JavaScript API: Traffic Layer. `https://developers.google.com/maps/documentation/javascript/examples/layer-traffic`. Online.

[117] Mattern, N., Schubert, R., and Wanielik, G. High-accurate vehicle localization using digital maps and coherency images. In *Intelligent Vehicles Symposium (IV), 2010 IEEE* (2010), IEEE, pp. 462–469.

[118] Matthaei, R., Bagschik, G., and Maurer, M. Map-relative localization in lane-level maps for adas and autonomous driving. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* (2014), IEEE, pp. 49–55.

[119] Mattyus, G., Wang, S., Fidler, S., and Urtasun, R. Enhancing road maps by parsing aerial images around the world. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1689–1697.

[120] Máttyus, G., Wang, S., Fidler, S., and Urtasun, R. Hd maps: Fine-grained road segmentation by parsing ground and aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3611–3619.

[121] Mendes, E., Koch, P., and Lacroix, S. Icp-based pose-graph slam. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (2016), IEEE, pp. 195–200.

[122] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[123] Mnih, V., and Hinton, G. E. Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision* (2010), Springer, pp. 210–223.

[124] Mokhtarzade, M., and Zoej, M. V. Road detection from high-resolution satellite images using artificial neural networks. *International journal of applied earth observation and geoinformation 9*, 1 (2007), 32–40.

[125] Monnier, F., Vallet, B., and Soheilian, B. Trees detection from laser point clouds acquired in dense urban areas by a mobile mapping system. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci 3* (2012), 245–250.

[126] Morrell, J. J. Estimated service life of wood poles. *Technical Bulletin, North American Wood Pole Council, http://www. woodpoles. org/documents/TechBulletin_EstimatedServiceLifeofWoodPole_12-08. pdf (Last accessed 5 April 2013)* (2008).

[127] Morton, J., Wheeler, T. A., and Kochenderfer, M. J. Analysis of recurrent neural networks for probabilistic modeling of driver behavior. *IEEE Transactions on Intelligent Transportation Systems 18*, 5 (2016).

[128] Neil, J., Cosart, L., and Zampetti, G. Precise timing for vehicle navigation in the smart city: an overview. *IEEE Communications Magazine 58*, 4 (2020), 54–59.

[129] Nguyen, A., and Le, B. 3D point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)* (2013), IEEE, pp. 225–230.

[130] Noda, M., Takahashi, T., Deguchi, D., Ide, I., Murase, H., Kojima, Y., and Naito, T. Vehicle ego-localization by matching in-vehicle camera images to an aerial image. In *Asian Conference on Computer Vision* (2010), Springer, pp. 163–173.

[131] of California DMV, S. DISENGAGEMENT REPORTS. `https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/`, 2020. Online.

[132] of Transportation MN, D. Pavement markings on challenging surfaces. `https://www.dot.state.mn.us/trafficeng/safety/docs/pavementmarkings.pdf`, 2007.

[133] Office of Highway Policy Information, U. D. o. T. F. H. A. Public road length - 2013, miles by functional system, 2014.

[134] Olsen, M. J., Parrish, C., Che, E., Jung, J., Greenwood, J., et al. Lidar for maintenance of pavement reflective markings and retroreflective signs. Tech. rep., Oregon. Dept. of Transportation, 2018.

[135] Olson, C. F. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation 16*, 1 (2000), 55–66.

[136] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[137] Pazhayampallil, J., and Kuan, K. Y. Deep learning lane detection for autonomous vehicle localization.

[138] Peker, A. U., Tosun, O., and Acarman, T. Particle filter vehicle localization and map-matching using map topology. In *2011 IEEE Intelligent Vehicles Symposium (IV)* (2011), IEEE, pp. 248–253.

[139] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), pp. 701–710.

[140] Petersen, N. C., Rodrigues, F., and Pereira, F. C. Multi-output bus travel time prediction with convolutional lstm neural network. *Expert Systems with Applications 120* (2019), 426–435.

[141] Pink, O. Visual map matching and localization using a global feature map. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on* (2008), IEEE, pp. 1–7.

[142] Pink, O., Moosmann, F., and Bachmann, A. Visual features for vehicle localization and ego-motion estimation. In *Intelligent Vehicles Symposium, 2009 IEEE* (2009), IEEE, pp. 254–260.

[143] Pink, O., and Stiller, C. Automated map generation from aerial images for precise vehicle localization. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on* (2010), IEEE, pp. 1517–1522.

[144] Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. Fast plane detection and polygonalization in noisy 3d range images. In *IROS* (2008), IEEE.

[145] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE 1*, 2 (2017), 4.

[146] Qu, X., Soheilian, B., and Paparoditis, N. Vehicle localization using mono-camera and geo-referenced traffic signs. In *Intelligent Vehicles Symposium (IV), 2015 IEEE* (2015), IEEE, pp. 605–610.

[147] Quddus, M. A., Ochieng, W. Y., and Noland, R. B. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation research part c: Emerging technologies 15*, 5 (2007), 312–328.

[148] Ranganathan, A., Ilstrup, D., and Wu, T. Light-weight localization for vehicles using road markings. In *IROS* (2013), IEEE, pp. 921–927.

[149] Rehder, E., and Albrecht, A. Submap-based slam for road markings. In *2015 IEEE Intelligent Vehicles Symposium (IV)* (2015), IEEE, pp. 1393–1398.

[150] Remondino, F. From point cloud to surface: the modeling and visualization problem. *International Archives of photogrammetry, Remote Sensing and spatial information sciences 34* (2003).

[151] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[152] Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision 66*, 3 (2006), 231–259.

[153] Rusu, R. B., Blodow, N., and Beetz, M. Fast point feature histograms (fpfh) for 3D registration. In *2009 IEEE international conference on robotics and automation* (2009), IEEE, pp. 3212–3217.

[154] Sapankevych, N. I., and Sankar, R. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine 4*, 2 (2009), 24–38.

[155] Schaefer, A., Büscher, D., Vertens, J., Luft, L., and Burgard, W. Long-term urban vehicle localization using pole landmarks extracted from 3-d lidar scans, 2019.

[156] Scheiblauer, C., and Wimmer, M. Out-of-core selection and editing of huge point clouds. *Computers & Graphics 35*, 2 (2011), 342–351.

[157] Schindler, A. Vehicle self-localization with high-precision digital maps. In *Intelligent Vehicles Symposium (IV), 2013 IEEE* (2013), IEEE, pp. 141–146.

[158] Schindler, A., Maier, G., and Janda, F. Generation of high precision digital maps using circular arc splines. In *Intelligent Vehicles Symposium (IV), 2012 IEEE* (2012), IEEE, pp. 246–251.

[159] Schlichting, A., and Brenner, C. Localization using automotive laser scanners and local pattern matching. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* (2014), IEEE, pp. 414–419.

[160] Schönberger, J. L., Pollefeys, M., Geiger, A., and Sattler, T. Semantic visual localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 6896–6906.

[161] Schreiber, M., Kn@oppel, C., and Franke, U. Laneloc: Lane marking based localization using highly accurate maps. In *Intelligent Vehicles Symposium (IV), 2013 IEEE* (2013), IEEE, pp. 449–454.

[162] Schütz, M. Potree: Rendering large point clouds in web browsers. *Technische Universität Wien, Wiedeń* (2016).

[163] Schwarz, B. Lidar: Mapping the world in 3d. *Nature Photonics 4*, 7 (2010), 429.

[164] Sefati, M., Daum, M., Sondermann, B., Kreisköther, K. D., and Kampker, A. Improving vehicle localization using semantic and pole-like landmarks. In *2017 IEEE Intelligent Vehicles Symposium (IV)* (2017), IEEE, pp. 13–19.

[165] Seif, H. G., and Hu, X. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering 2*, 2 (2016), 159–162.

[166] Senlet, T., and Elgammal, A. A framework for global vehicle localization using stereo images and satellite and road maps. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* (2011), IEEE, pp. 2034–2041.

[167] Seo, Y.-W. *Augmenting cartographic resources and assessing roadway state for vehicle navigation.* Carnegie Mellon University, 2012.

[168] Seo, Y.-W., Urmson, C., and Wettergreen, D. Ortho-image analysis for producing lane-level highway maps. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems* (2012), ACM, pp. 506–509.

[169] Shalev-Shwartz, S., Shammah, S., and Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR abs/1610.03295* (2016).

[170] Shan, Q., Wu, C., Curless, B., Furukawa, Y., Hernandez, C., and Seitz, S. M. Accurate geo-registration by ground-to-aerial image matching. In *3D Vision (3DV), 2014 2nd International Conference on* (2014), vol. 1, IEEE, pp. 525–532.

[171] Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., and Kamaev, V. A. A survey of forecast error measures. *World Applied Sciences Journal 24*, 24 (2013), 171–176.

[172] Shiller, Z., and Gwo, Y.-R. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation 7*, 2 (1991), 241–249.

[173] Sinha, K. C., and Fwa, T. F. On the concept of total highway management. *Transportation Research Record 1229* (1989), 79–88.

[174] Site, P. N. W. Surveying and mapping law of the people's republic of china [eb/ol].

[175] Skardinga, J., Gabrys, B., and Musial, K. Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access* (2021).

[176] Smagulova, K., and James, A. P. A survey on LSTM memristive neural network architectures and applications. *The European Physical Journal Special Topics 228*, 10 (2019), 2313–2324.

[177] Socher, R., Huval, B., Bath, B., Manning, C. D., and Ng, A. Y. Convolutional-recursive deep learning for 3d object classification. In *Advances in neural information processing systems* (2012), pp. 656–664.

[178] Spangenberg, R., Goehring, D., and Rojas, R. Pole-based localization for autonomous vehicles in urban scenarios. In *IROS* (2016), IEEE, pp. 2161–2166.

[179] Standard, N. D. Navigation data standard - open lane model documentation. Tech. rep., Navigation Data Standard, 2016.

[180] Sun, X., Xie, Y., Luo, P., Wang, L., and Unit, B. A. D. B. A dataset for benchmarking image-based localization.

[181] Surmann, H., Nüchter, A., and Hertzberg, J. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems 45*, 3-4 (2003), 181–198.

[182] Tao, Z., Bonnifait, P., Fremont, V., and Ibanez-Guzman, J. Mapping and localization using gps, lane markings and proprioceptive sensors. In *IROS* (2013), IEEE, pp. 406–412.

[183] Taxi, and Commission, L. New York city taxi trip data, 2009-2018. Inter-university Cons. for Political and Social Research, 2019.

[184] Technologies, H. Traffic API: Flow using Quadkey. `https://developer.here.com/documentation/examples/rest/traffic/traffic-flow-quadkey`. Online.

[185] Technology, R. The world's biggest road networks, 2014.

[186] Thompson, P., Ford, K., Arman, M., Labi, S., Sinha, K., and Shirolé, A. Nchrp report 713: Estimating life expectancies of highway assets. *Transportation Research Board of the National Academies, Washington, DC* (2012).

[187] Thrun, S., Fox, D., Burgard, W., and Dellaert, F. Robust monte carlo localization for mobile robots. *Artificial intelligence 128*, 1-2 (2001), 99–141.

[188] TMTpost. Baidu driverless cars run in wuzhen, powered by four leading technologies, 2016.

[189] Toledo-Moreo, R., Bétaille, D., and Peyret, F. Lane-level integrity provision for navigation and map matching with gnss, dead reckoning, and enhanced maps. *IEEE Transactions on Intelligent Transportation Systems 11*, 1 (2010), 100–112.

[190] Toledo-Moreo, R., Bétaille, D., Peyret, F., and Laneurit, J. Fusing gnss, dead-reckoning, and enhanced maps for road vehicle lane-level navigation. *IEEE Journal of Selected Topics in Signal Processing 3*, 5 (2009), 798–809.

[191] TomTom. Hd map - highly accurate border-to-border model of the road, 2015.

[192] TomTom. Tomtom hd map with roaddna, 2017.

[193] Torii, A., Arandjelovic, R., Sivic, J., Okutomi, M., and Pajdla, T. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1808–1817.

[194] Torii, A., Sivic, J., Pajdla, T., and Okutomi, M. Visual place recognition with repetitive structures. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2013), pp. 883–890.

[195] Toriya, H., Kitahara, I., and Ohta, Y. Mobile camera localization using aerial-view images. *Information and Media Technologies 9*, 4 (2014), 896–904.

[196] Trinder, J. C., and Wang, Y. Automatic road extraction from aerial images. *Digital Signal Processing 8*, 4 (1998), 215–224.

[197] Trommer, S., Kröger, L., and Kuhnimhof, T. Potential fleet size of private autonomous vehicles in germany and the us. In *Road Vehicle Automation 4*. Springer, 2018, pp. 247–256.

[198] Uchiyama, H., Deguchi, D., Takahashi, T., Ide, I., and Murase, H. Ego-localization using streetscape image sequences from in-vehicle cameras. In *Intelligent Vehicles Symposium, 2009 IEEE* (2009), IEEE, pp. 185–190.

[199] Vaca-Castano, G., Zamir, A. R., and Shah, M. City scale geo-spatial trajectory estimation of a moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 1186–1193.

[200] Viswanathan, A., Pires, B. R., and Huber, D. Vision based robot localization by ground to satellite matching in gps-denied situations. In *IROS* (2014), IEEE, pp. 192–198.

[201] Wang, D., Yang, Y., and Ning, S. Deepstcl: A deep spatio-temporal convlstm for travel demand prediction. In *2018 international joint conference on neural networks (IJCNN)* (2018), IEEE.

[202] Wang, D., Zhang, J., Cao, W., Li, J., and Zheng, Y. When will you arrive? estimating travel time based on deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[203] Wang, Y., Zheng, Y., and Xue, Y. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), pp. 25–34.

[204] Wang, Z., Fu, K., and Ye, J. Learning to estimate the travel time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 858–866.

[205] Weiss, T., Kaempchen, N., and Dietmayer, K. Precise ego-localization in urban areas using laserscanner and high accuracy feature maps. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE* (2005), IEEE, pp. 284–289.

[206] Welzel, A., Reisdorf, P., and Wanielik, G. Improving urban vehicle localization with traffic sign recognition. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems* (2015), IEEE, pp. 2728–2732.

[207] Weng, L., Yang, M., Guo, L., Wang, B., and Wang, C. Pole-based real-time localization for autonomous driving in congested urban scenarios. In *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)* (2018), IEEE, pp. 96–101.

[208] Wickramanayake, S., and Bandara, H. D. Fuel consumption prediction of fleet vehicles using machine learning: A comparative study. In *2016 Moratuwa Engineering Research Conference (MERCon)* (2016), IEEE, pp. 90–95.

[209] Wiest, J., Deusch, H., Nuss, D., Reuter, S., Fritzsche, M., and Dietmayer, K. Localization based on region descriptors in grid maps. In *Intelligent Vehicles Symposium* (2014), IEEE.

[210] Wimmer, M., and Scheiblauer, C. Instant points: Fast rendering of unprocessed point clouds. In *SPBG* (2006), Citeseer, pp. 129–136.

[211] Wolcott, R. W., and Eustice, R. M. Visual localization within lidar maps for automated urban driving. In *IROS* (2014), IEEE, pp. 176–183.

[212] Wolff, M., Collins, R. T., and Liu, Y. Regularity-driven facade matching between aerial and street views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1591–1600.

[213] Wong, D., Deguchi, D., Ide, I., and Murase, H. Position interpolation using feature point scale for decimeter visual localization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2015), pp. 1–8.

[214] Wong, K., Gu, Y., and Kamijo, S. Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine 13*, 1 (2020), 91–106.

[215] Wonka, P., Wimmer, M., and Sillion, F. X. Instant visibility. In *Computer Graphics Forum* (2001), vol. 20, Wiley Online Library, pp. 411–421.

[216] Workman, S., Souvenir, R., and Jacobs, N. Wide-area image geolocalization with aerial reference imagery. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 3961–3969.

[217] Wu, C.-H., Ho, J.-M., and Lee, D.-T. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems 5*, 4 (2004), 276–281.

[218] Wu, T., and Ranganathan, A. Vehicle localization using road markings. In *Intelligent Vehicles Symposium (IV), 2013 IEEE* (2013), IEEE, pp. 1185–1190.

[219] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1912–1920.

[220] Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[221] Xu, X., Wang, X., Wu, X., Hassanin, O., and Chai, C. Calibration and evaluation of the responsibility-sensitive safety model of autonomous car-following maneuvers using naturalistic driving study data. *Transportation research part C: emerging technologies 123* (2021), 102988.

[222] Xu, Y., John, V., Mita, S., Tehrani, H., Ishimaru, K., and Nishino, S. 3d point cloud map based vehicle localization using stereo camera. In *2017 IEEE intelligent vehicles symposium (IV)* (2017), IEEE, pp. 487–492.

[223] Xue, H., Huynh, D. Q., and Reynolds, M. SS–LSTM: A hierarchical lstm model for pedestrian trajectory prediction. In *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018).

[224] Yan, W. Y., Morsy, S., Shaker, A., and Tulloch, M. Automatic extraction of highway light poles and towers from mobile lidar data. *Optics & Laser Technology 77* (2016), 162–168.

[225] Yang, C., and Gidofalvi, G. Fast map matching, an algorithm integrating hidden markov model with precomputation. *International Journal of Geographical Information Science 32*, 3 (2018), 547–570.

[226] Yoneda, K., Tehrani, H., Ogawa, T., Hukuyama, N., and Mita, S. Lidar scan feature for localization with highly precise 3-d map. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* (2014), IEEE, pp. 1345–1350.

[227] Yuan, H., Li, G., Bao, Z., and Feng, L. Effective travel time estimation: When historical trajectories over road networks matter. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020), pp. 2135–2149.

[228] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., and Huang, Y. T-drive: Driving directions based on taxi trajectories. In *Proceedings of 18th ACM SIGSPATIAL Conference on Advances in Geographical Information Systems* (2010).

[229] Zang, A., Chen, X., and Trajcevski, G. High-definition digital elevation model system vision paper. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management* (2017), ACM, p. 29.

[230] Zang, A., Chen, X., and Trajcevski, G. High-definition digital elevation model system (vision paper). In *Proceedings of International Conference on Scientific and Statistical Database Management* (2017), ACM.

[231] Zang, A., Chen, X., and Trajcevski, G. High definition maps in urban context. *SIGSPATIAL Special 10* (06 2018), 15–20.

[232] Zang, A., Li, Z., Doria, D., and Trajcevski, G. Accurate vehicle self-localization in high definition map dataset. In *Proceedings of the 1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles* (2017), ACM.

[233] Zang, A., Luo, S., Chen, X., and Trajcevski, G. Real-time applications using high resolution 3D objects in high definition maps (systems paper). In *ACM SIGSPATIAL* (2019), pp. 229–238.

[234] Zang, A., Xu, R., Li, Z., and Doria, D. Lane boundary extraction from satellite imagery. In *Proceedings of the 1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles* (2017), ACM.

[235] Zang, A., Zhu, X., Guo, Y., Zhou, F., and Trajcevski, G. Towards predicting vehicular data consumption. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)* (2021), IEEE, pp. 109–114.

[236] ZAVODNY, A., and Chen, X. Method and apparatus for processing and aligning data point clouds, 2016. US Patent 9,424,672.

[237] Zhang, Y., Li, Y., Zhou, X., Kong, X., and Luo, J. Curb-gan: Conditional urban traffic estimation through spatio-temporal generative adversarial networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 842–852.

[238] Zhao, P., Zhu, H., Liu, Y., Li, Z., Xu, J., and Sheng, V. S. Where to go next: A spatio-temporal LSTM model for next poi recommendation. *arXiv preprint arXiv:1806.06671* (2018).

[239] Zhou, B., Liu, L., Oliva, A., and Torralba, A. Recognizing city identity via attribute analysis of geo-tagged images. In *European conference on computer vision* (2014), Springer, pp. 519–534.

Vita

Andi Zang obtained his B.S. in Telecommunications at Beijing University of Posts and Telecommunications in 2012 and M.S. in Computer Science at Illinois Institute of Technology in 2014. During his Ph.D. program at Northwestern University, he published seven papers and was granted three patents as following:

Papers:

(1) **A. Zang**, X. Zhu, C. Li, F. Zhou, G Trajcevski, *"Integrating Heterogeneous Sources for Learned Prediction of Vehicular Data Consumption"*. Proceedings of the 23rd IEEE International Conference on Mobile Data Management, 2022.

(2) **A. Zang**, X. Zhu, Y. Guo, F. Zhou, G Trajcevski, *"Towards Predicting Vehicular Data Consumption"*. Proceedings of the 22nd IEEE International Conference on Mobile Data Management, 2021.

(3) **A. Zang**, S. Luo, X. Chen, G. Trajcevski, *"Real-Time Applications Using High Resolution 3D Objects in High Definition Maps (System Paper)"*. Proceedings of the 27th ACM International Conference on Advances in Geographic Information Systems, 2019.

(4) **A. Zang**, X. Chen, G. Trajcevski, *"High Definition Maps in Urban Context"*. SIGSPATIAL Special, 2018.

(5) **A. Zang**, Z. Li, D. Doria, G. Trajcevski, *"Accurate Vehicle Self-localization in High Definition Map Dataset"*. Proceedings of the 1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles, 2017.

(6) **A. Zang**, R. Xu, Z. Li, D. Doria, *"Lane Boundary Extraction from Satellite Imagery"*. Best paper. Proceedings of the 1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles, 2017.

(7) **A. Zang**, X. Chen, G. Trajcevski, *"High-definition Digital Elevation Model System (Vision Paper)"*. Proceedings of the 29th International Conference on Scientific and Statistical Database Management, 2017.

Patents:

(1) **A. Zang**, X. Chen, *"Method and apparatus for self localization"*. US Patent 16/830,772, 2021

(2) **A. Zang**, Z. Li, *"Road modeling from overhead imagery"*. US Patent 10,628,671, 2020.

(3) **A. Zang**, X. Chen, *"Method and apparatus for providing a tile-based digital elevation model"*. US Patent 10,482,655, 2019.