UNIVERSITY OF CALIFORNIA SAN DIEGO

Mapping and Planning for Autonomous Vehicles in Dynamic Urban Settings

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

David Paz Ruiz

2023

University of California San Diego

2023

DEDICATION

To my parents, Catalina Ruiz and Bertin Paz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am deeply grateful to my research advisor, Dr. Henrik I. Christensen, for his invaluable support and guidance throughout my undergraduate and graduate studies. His immeasurable backing and direction have played a vital role in shaping my abilities and understanding that were required to traverse my PhD. I express special gratitude for the thought-provoking conversations and valuable input he has generously shared with me over the past six years. His perspectives have contributed significantly to my growth as a scholar and as an individual.

I also want to acknowledge the members of my PhD committee for their teachings, support, and career advice over the years. I have learned invaluable insights from our exchanges. Thank you Dr. Nikolay Atanasov, Dr. Falko Kuester, Dr. Hao Su, and Dr. Rose Yu.

I would like to extend my heartfelt gratitude to the members of the Cognitive Robotics Lab and the Autonomous Vehicle Laboratory for granting me the opportunity to engage in collaborative endeavors with them across various projects. I express special appreciation to the following individuals: Hengyuan Zhang, Po-Jung Lai, Nathan Chan, Dominique Meyer, Sumukha Harish, Francis Joseph, Shawn Winston, Yuqing Jiang, Emily Le, Carlos Nieto, Shengye Wang, Ruffin White, Ploy Temiyasathit, Shixin Li, Qinru Li, Hao Xiang, Yunhai Han, Yuhan Liu, Jing-Yan Liao, Parth Kumar, Andrew Liang, Andres Gutierrez, Seth Farrell, Srinidhi Srinivas, Robin Wang, Parth Doshi, Narayanan Elavathur Ranganatha, Zihan Zhang, and Varun Vupparige. Your contributions have been instrumental in the success of our joint ventures.

I would also like to thank members of the Robotic Graduate Student Organization (Robo-Grads) for the great experiences and for helping build a stronger community in robotics at UCSD. Thank you Jacob Johnson, Sachiko Matsumoto, Pengcheng Cao, Anya Bouzida, Sander Tonkens, Nikhil Shinde, Shrutheesh Iyer, Pratyusha Ghosh, and Rabeya Jamshad.

–

Thank you.

The following describes co-author contributions and materials utilized in this dissertation.

Chapter 2 includes material from the paper published in Probabilistic Semantic Mapping for Urban Autonomous Driving Applications in International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, Oct 2020. David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik Christensen. The dissertation author was one of the primary co-authors and contributed to the implementation, design, and the dataset associated with the approach. Zhang contributed to the dataset, implementation, and evaluation of the semantic mapping module. Li contributed to the semantic segmentation design, training, and evaluation of the semantic mapping pipeline. Xiang contributed to the semantic mapping evaluation and formulation of the confusion matrix formulation.

Chapter 3 includes material from three papers. The first paper corresponds to TridentNet: A Conditional Generative Model for Dynamic Trajectory Generation in Intelligent Autonomous Systems-16, Singapore, June 2021. David Paz, Henry Zhang, and Henrik I Christensen. The second paper corresponds to Tridentnetv2: Lightweight Graphical Global Plan Representations for Dynamic Trajectory Generation in International Conference on Robotics and Automation, Philadelphia, May 2022. David Paz, Hao Xiang, Andrew Liang, and Henrik I Christensen. The third paper corresponds to Conditional Generative Models for Dynamic Trajectory Generation and Urban Driving in Sensors2023, July 2023. David Paz, Hengyuan Zhang, Hao Xiang, Andrew Liang, and Henrik I Christensen. The dissertation author was the primary co-author and contributed to the implementation, design, datasets, and evaluation of the approach. Zhang contributed to the dataset associated with the experiments. Xiang and Liang contributed to the global planner implementation.

Chapter 4 includes material being prepared for peer review. The dissertation author is the primary co-author and contributed to the design, implementation, dataset generation, and evaluations. Narayanan Elavathur Ranganatha contributed to the implementation and dataset generation process. Srinidhi Srinivas contributed to the implementation, and dataset generation.

Yunchao Yao assisted in the development of the datasets and data curation.

VITA

2015	A.A.S, Physics, San Diego Mesa College
2016	Undergraduate Research Assistant, i-Trek, Massachusetts Institute of Technology
2017	Undergraduate Research Assistant, San Diego Supercomputer Center
2017	Undergraduate Research Assistant, Computation Structures Group, Massachusetts Institute of Technology
2018	Bachelor of Science in Computer Engineering, University of California San Diego
2018-2023	Research Scientist/Project Lead, Autonomous Vehicle Laboratory, University of California San Diego
2018-2019	L4 - Verification Engineer, TuSimple, San Diego
2020	Master of Science in Electrical Engineering (Intelligent Systems, Robotics and Control), University of California San Diego
2021	Perception Engineer Intern, TuSimple, San Diego
2021-2022	Robotics Research Assistant, Qualcomm, San Diego
2023	Doctor of Philosophy in Computer Science, University of California San Diego
2023-Present	AI Research Scientist, Bosch Research, Sunnyvale

PUBLICATIONS

David Paz, Hengyuan Zhang, Hao Xiang, Andrew Liang, and Henrik I. Christensen. Conditional Generative Models for Dynamic Trajectory Generation and Urban Driving. Sensors2023, 23, 6764.

Hengyuan Zhang*, Shashank Venkatramani*, **David Paz**, Qinru Li, Hao Xiang, and Henrik I. Christensen. Probabilistic Semantic Mapping for Autonomous Driving in Urban Environments. Sensors 2023, 23, 6504.

Hengyuan Zhang, Jing-Yan Liao, **David Paz**, and Henrik I. Christensen. Robust human identity anonymization using pose estimation. In 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), pages 619–626, 2022

David Paz, Hao Xiang, Andrew Liang, and Henrik I Christensen. Tridentnetv2: Lightweight graphical global plan representations for dynamic trajectory generation. In International Conference on Robotics and Automation, Philadelphia, May 2022. IEEE

Henrik Christensen, **David Paz**, Hengyuan Zhang, Dominique Meyer, Hao Xiang, Yunhai Han, Yuhan Liu, Andrew Liang, Zheng Zhong, and Shiqi Tang. Autonomous vehicles for micro-mobility. In Autonomous Intelligent Systems, 2021

David Paz, Henry Zhang, and Henrik I Christensen. TridentNet: A conditional generative model for dynamic trajectory generation. In Intelligent Autonomous Systems-16, Singapore, June 2021. (Best paper)

Yunhai Han, Yuhan Liu, **David Paz**, and Henrik Christensen. Auto-calibration method using stop signs for urban autonomous driving applications. In International Conference on Robotics and Automation, Xian, May 2021. IEEE

David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik Christensen. Probabilistic semantic mapping for urban autonomous driving applications. In International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, Oct 2020. IEEE/RSJ

David Paz. Autonomous vehicles: Their capabilities and limitations. In UC San Diego Electronic Theses and Dissertations, 2020

David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit Binnani, and Henrik Christensen. Lessons learned from deploying autonomous vehicles at UC San Diego. In *Field and Service Robotics*, Tokyo, JP, August 2019

Emily Le and **David Paz**. Performance analysis of applications using singularity container on sdsccomet. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, PEARC17, pages 66:1–66:4, New York, NY, USA, 2017. ACM

ABSTRACT OF DISSERTATION

Mapping and Planning for Autonomous Vehicles in Dynamic Urban Settings

by

David Paz Ruiz

Doctor of Philosophy in Computer Science

University of California San Diego, 2023

Professor Henrik I Christensen, Chair

In highly dynamic urban environments, software stacks for autonomous driving applications must quickly adapt to fast changing environments. Examples of dynamic scenarios include construction sites, road closures, and lane level updates. Failure to adapt to changes in map definitions can result in catastrophic failures in the system that can lead to accidents or, at best, rule violations in shared public roads. This work focuses on identifying strategies that leverage automatically generated map representations to minimize human-in-the-loop efforts and explores methods for integrating nominal planners in the global planning task.

The first part of this dissertation covers multi-class semantic mapping for large scale urban driving applications. As part of this framework, sensor fusion based strategies are applied to provide robust depth and semantic estimates from the scene without making strong assumptions about the road topology. Secondly, rasterized and graphical representations are jointly leveraged to formulate a nominal global planning approach for lane-level navigation. This method utilizes the semantic maps introduced and employs a conditional generative model to explicitly model the multi-modal distribution of trajectories that are feasible when driving in an urban setting. We additionally provide details from real-world testing and the open-source data collected from the UC San Diego campus during 2020-2021. In the last chapter, 2D and 3D centerline prediction methods are introduced to reduce the gap in real-time scene understanding. This contribution outlines an automatic label generation process and additionally leverages an occlusion handling approach to reason about centerline prediction with varying degrees of occlusion. The methods proposed achieve robust performance in diverse driving scenarios with promising directions in autonomous driving architectures.

Chapter 1

Introduction

The evolution of autonomous driving technology has shown potential for a new era of transportation, promising increased safety, efficiency, and convenience. The pursuit of fully autonomous vehicles has spurred intensive research and innovation, leading to the development of cutting-edge techniques that push research boundaries. In recent years, the challenges of autonomous driving have shifted from mere navigation to encompass a deeper understanding of the environment and dynamic decision making. As vehicles navigate complex and dynamic urban landscapes and interact with a variety of agents, the need for sophisticated solutions becomes evident. However, building autonomous systems for operation in urban environments requires dynamic mapping, planning in the presence of dynamic objects, and generalization beyond fixed environments. Hence, we propose a model for mapping and planning based on the use of a minimal set of prior models and show that on-the-fly perception and interpretation allows design of systems for operation in dynamic urban settings.

The subsequent chapters provide a detailed account of the contributions presented within this dissertation. Chapter 2 introduces a framework for semantic map generation. Through the integration of sensor fusion, this approach lays the groundwork for a more comprehensive understanding of a surrounding environment.

Building upon the foundation laid by the previous chapter, Chapter 3 explores the concept of dynamic trajectory generation by leveraging the power of conditional generative models. This approach can enable planners to adapt to changing global plans while reducing dependence on expensive map priors. The result is a step towards reducing the reliance on HD maps–which present a considerable scalability and maintenance challenge for larger scale deployments.

Chapter 4 introduces real-time 2D and 3D centerline prediction models by leveraging automatic label generation and occlusion handling techniques. The methods explored aim to simplify the complexities in scene modeling while focusing on downstream applications such as global planning. A number of quantitative and qualitative evaluations are performed with various datasets. Finally, a number of conclusions and directions for future work are presented in Chapter 5.

In addition to the technical contributions presented in this dissertation, the datasets and code repositories associated with the different components are open and publicly available to facilitate the development of future research directions within the scientific community. These datasets and repositories can be accessed at `http://avl.ucsd.edu` and at `https://github.com/AutonomousVehicleLaboratory`. Additional details are provided in the different sections of the text.

Chapter 2

Scene Understanding

High-definition (HD) maps provide useful information for autonomous vehicles to understand the static parts of the scene. Due to the nature of the information encoded in HD maps–such as centimeter-level definitions for road networks, traffic signs, crosswalks, stop signs, traffic lights, and even speed limits–many of these maps become outdated during construction or road network changes. Given these fast-changing environments, manually annotated HD maps become obsolete and may cause vehicles to perform inadequate reference path tracking actions leading to unsafe scenarios. In the process of HD map generation, extracting semantics and attributes from data takes the most amount of the work [Jia18]. A model that automates this process could improve HD map generation, reduce labor costs, and increase driving safety.

Retrieval of centimeter-level semantic labels of the scene is a non-trivial task. Prior work such as [DFRDW11, SSLT12] adopted Conditional Random Fields (CRF) to assign semantic labels. The advancement of deep learning also provides promising results in terms of retrieving semantic information from images. State of the art semantic segmentation algorithms such as [LSD15, ZSQ+17, CZP+18] generate pixel-level semantic labels with greater accuracy. Researchers have also explored methods to create semantic maps of the environment: examples are given in [MCUS18, MLU17, HML+19]. Multi-sensor fusion has been used to improve the robustness of these algorithms; however, these approaches either use aerial imagery to extract road information or do not explicitly map the lane and crosswalk information, which are required for HD maps. Therefore, a detailed semantic map for urban autonomous vehicle applications is still of interest to explore.

To address these gaps, this work focuses on leveraging dense point maps built from a 16-channel LiDAR and state of the art semantically labeled images from deep neural networks, trained only on a publicly available dataset, to automatically generate dense probabilistic semantic maps in urban driving environments that provide robust labels for roads, lane marks, crosswalks, and sidewalks. We create a bird's eye view (BEV) semantic map of the environments by modeling the uncertainty of the semantic segmentation network with a confusion matrix formulation. Furthermore, the fusion of LiDAR intensity slightly improves the accuracy of lane mark segmentation on the road. The comparison with a ground truth HD map that has been tested in our autonomous vehicle for campus mail delivery tasks shows that the proposed model can identify semantic features on the road and localize them accurately.

2.1 Related Work

2.1.1 Semantic Segmentation

Semantic segmentation is the task of assigning each observed data point (e.g. pixel or voxel) to a class label that contains semantic meanings. Research in this field has made tremendous progress as large scale datasets like CityScapes [COR+16], CamVid [BFC08], Mapillary[NORBK17], become available. Given that HD maps require fine-grained labeling for each object, building such maps can significantly benefit from the pixel level information provided by semantic segmentation algorithms.

In 2D semantic segmentation, predominant works [LSD15, ZSQ+17, CPSA17] leverage pyramid like encoder-decoder architecture to capture both global and local information in the images. Trained on the large scale datasets aforementioned, these network architectures can easily detect objects on the road even when these objects only have textural differences like colors. In 3D semantic segmentation, work has also been done by modeling 3D LiDAR point clouds as range images and then feeding them to a classical CNN network to classify each point[WWYK18, HPSS20, WSY$^+$18]. While the results of these works seem promising, due to the nature of LiDAR sensors, these methods cannot distinguish objects with texture differences such as colors. Researchers have also proposed an alternative approach of segmentation on voxelized point clouds [TCA$^+$17]. However, such 3D convolutions are computationally expensive and usually require dense raw point cloud measurements (i.e 32 or 64-channel LiDARs), making it troublesome to operate in real time.

2.1.2 Semantic Mapping

Semantic mapping has a rich meaning in various literature [KG15]. We adopt the definition from [WS08], which is the process of building maps that represent not only an occupancy metric but also other properties of the environment. In the context of autonomous driving, the drivable areas and road features are often included.

Other methods like [SSLT12] propose CRF based methods in dense semantic mapping. They use associative hierarchical CRF for semantic segmentation and pairwise CRF for mapping. The pairwise potential minimization enforces the output smoothness. Sengupta et. al. [SGST13] utilize a stereo pair to estimate the depth robustly, but they do not explicitly map the lane and crosswalk information, which are required for HD maps.

Daniel et. al. [MCUS18] fuse semantic images from a camera with LiDAR point clouds, but they use real-time raw point clouds from a 64-channel LiDAR providing much denser real-time information whereas we can build map from a 16-channel LiDAR that has lower cost. Additionally, their focus is on off-road terrains, in contrast to our focus: the urban driving scenario. Urban driving scenes require special treatment of classes that contain specific traffic rules, such as lane marks and crosswalks.

2.1.3 Probabilistic Mapping

Probabilistic maps have been successfully applied to localization [LT10] [STGBS17] and pedestrian motion prediction [WRA18]. Probabilistic maps can capture the inherent distribution information in a discrete space while filtering the noise. In this work, we successfully apply these techniques to semantic map generation while leveraging the prior information in LiDAR's intensity channel to produce more stable semantic maps.

2.2 Semantic Segmentation for Mapping Applications

The framework consists of three parts: semantic segmentation, semantic association, and semantic mapping. The overall architecture is shown in Figure 2.1. We use semantic segmentation networks to predict semantic labels on 2D images and then associate semantic labels with densified 3D point clouds. Afterwards we apply probabilistic mapping to capture the distribution of labels assigned to each grid. In this section, we will describe each part in detail.

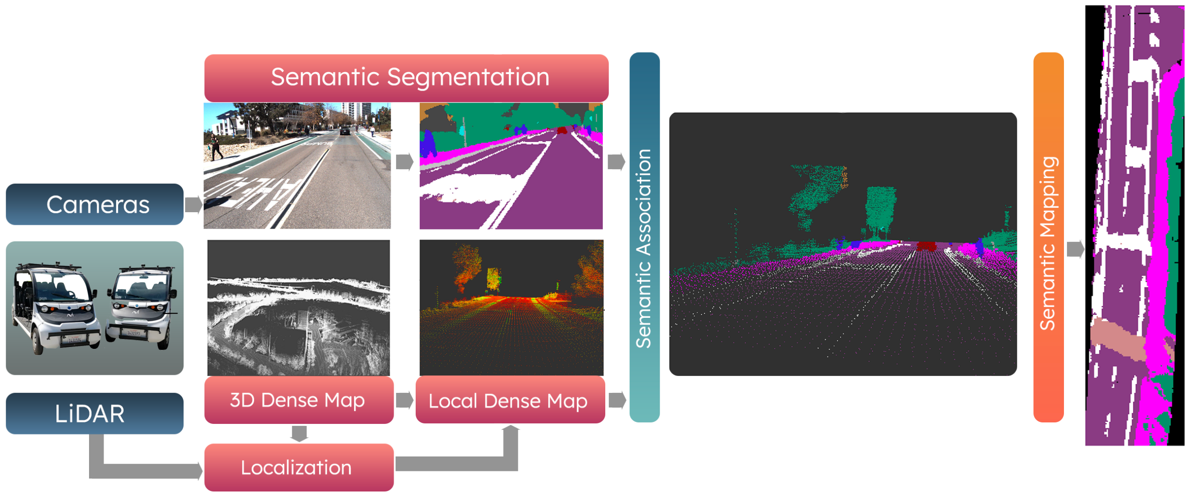


Figure 2.1: Our pipeline for generating probabilistic semantic maps for road feature extraction and HD mapping applications. Our model consists of three parts: *i*) Semantic Segmentation: predict semantic labels based on 2D images. *ii*) Semantic Association: associate point clouds with predicted semantic labels. *iii*) Semantic Mapping: use a probabilistic semantic mapping method to capture the latent distribution of each label and utilize LiDAR intensity to augment lane mark prediction.

2.2.1 Image Semantic Segmentation

A DeepLabV3Plus [CPSA17] network architecture is leveraged to extract the semantic segmentation from 2D images. A lightweight ResNeXt50 [XGD$^+$17] pre-trained on ImageNet [RDS$^+$15] is used as our feature extraction backbone. Compared with other backbones like ResNet101 [HZRS16a], it can achieve the same mean of intersection over union (mIoU) value with fewer parameters and faster inference times. We also adopt the depth-wise separable convolution inspired by [CPSA17, Cho17] in our spatial pyramid layers and decoder layers to further improve the inference time while preserving the same performance.

The semantic segmentation network is trained in the Mapillary Vistas dataset [NORBK17]. This dataset provides a large number of pixel-level semantic segmented images with 66 different kinds of labels in autonomous vehicle scenarios. We reduce the labels into 19 classes by removing labels that are not essential in our driving environment (e.g. snow) and merging labels with similar semantic meanings together (e.g. zebra line and crosswalk). This decision is made based on the observation that some classes are unlikely to appear in our test environment. The details of label merging can be found in Section 2.3. All the training labels and their associated colors are presented in Table 2.1.

Table 2.1: Training Labels and Their Associated Colors for the Semantic Segmentation Network

Training Labels			
curb	crosswalk	road	sidewalk
building	person	bicyclist	motorcyclist
lane marking (white)	sky	vegetation	manhole
pole	traffic-sign	bicycle	bus
car	motorcycle	truck	

2.2.2 Point Cloud Semantic Association

Given a semantic image, estimating the relative depth for the semantic pixel data can help us reconstruct the 3D scene with semantic labels. This information, however, is usually not available. Depth estimation from multi-view geometry requires salient features, which is prone to error on the road or in challenging lighting conditions. In contrast, LiDAR sensors can readily capture the depth information of the objects, but as they are often equipped with few optical channels (e.g. 16), it can be difficult to infer the underlying geometry in real time due to their sparse resolution. To alleviate this problem, our method leverages centimeter-level localization [PLH$^+$19] to extract small dense regions of a previously built dense point cloud map. These smaller regions are then projected into the semantically segmented image to retrieve depth information. Building such a dense point map can be automated and only requires driving through the area once and is thus less expensive than human labeling.

With localization in place, the intrinsic camera parameters and the relative transformation between the camera and LiDAR are used to project point cloud data into the 2D image space. The point cloud data is then associated semantically using the nearest neighbor search. The relative transformation between the camera and the LiDAR is estimated using the PnP method [LMNF09], and the intrinsic matrix of the camera is determined by a traditional chessboard method.

2.2.3 Semantic Mapping

While a point cloud with semantic labels naturally preserves the 3D geometry of the environment, such representation of the scene is subject to the sensor measurement noise and small semantic label fluctuations. To address this, we maintain a local or global probabilistic map, where the local map can provide direct dense semantic cues around the ego-vehicle and the global map can help automate the process of building HD maps. Semantic occupancy grids are used for both local and global semantic maps while the main difference is the reference frame. Our quantitative comparisons are performed in the global frame.

A local probabilistic map is a bird's eye view representation in the body frame (rear-axle) of the ego vehicle. We build a local map for a given frame i, with the origin defined by the corresponding pose, and update it by using the semantic point cloud. Only when the difference of our new pose and old pose is beyond a threshold do we construct a new map and transform the previous map to account for vehicle movement. In contrast, a global probabilistic map operates directly in the global frame without the need of map transformations. A side-by-side visual comparison is shown in Figure 2.3; where the top image corresponds to a local frame representation and the bottom image corresponds to a global frame representation.

The semantic occupancy grid has height H, width W, and channels C. Each channel corresponds to a semantic class of the scene. When the semantic point cloud is constructed, we project it onto the grid using the x and y components. The semantic label of the point will be regarded as the observed semantic label of its nearest cell c_{ij}. Every cell in the grid covers a $d \times d$ square area (in meters) of the physical world where the value d is a discretization factor.

The robustness of the semantic occupancy grid estimation is enhanced by a probabilistic model which leverages both the semantic and LiDAR intensity information from the point cloud to reduce the prediction error. We denote the semantic label distribution across all the channels as $\mathbf{S}_t$, the observed semantic labels as $\mathbf{z}_t$, and observed LiDAR intensity as I_t. Hence, the task is to estimate $\mathbf{S}_t$ from our past observations, i.e. the probability distribution of $P(\mathbf{S}_t|\mathbf{z}_{1:t}, I_{1:t})$. By following the Markov assumption and assuming that observed semantic labels and LiDAR intensity are conditionally independent given $\mathbf{S}_t$, the update rule of the semantic probability is

$$P(\mathbf{S}_t|\mathbf{z}_{1:t}, I_{1:t}) = \frac{1}{\mathbf{Z}} P(\mathbf{z}_t|\mathbf{S}_t) P(I_t|\mathbf{S}_t) P(\mathbf{S}_{t-1}|\mathbf{z}_{1:t-1}, I_{1:t-1})$$

where $\mathbf{Z}$ is a normalization factor. Here we also assume that $P(\mathbf{S}_t|\mathbf{z}_{1:t-1}, I_{1:t-1}) = P(\mathbf{S}_{t-1}|\mathbf{z}_{1:t-1}, I_{1:t-1})$. We model $P(\mathbf{z}_t|\mathbf{S}_t)$ with a 2D matrix $\mathbf{M}$ where an element in the ith row and jth column represents the likelihood of label i being predicted as label j. It characterizes our confidence of prediction to allow a more accurate probabilistic update. We model $P(I_t|\mathbf{S}_t)$ as a function of the reflectivity rate of each class in the scene.

The intensity from LiDAR sensors serves a strong cue for different materials on the scene. For example, the top image in Figure 2.6 illustrates a BEV intensity map on a road segment. Because lane marks are painted white, they can reflect light at higher intensity and thus be segmented out with a threshold k. We can use this as a prior to reason about the layout of the scene: it can help for the cases when semantic segmentation fails to capture the true label in poor lighting conditions.

2.3 Experiments and Data

The experimental data was collected by using a full scale autonomous vehicle platform [PLH$^+$19]. The vehicle is equipped with a 16-channel LiDAR and six cameras. The cameras are set up as two on the front, one on each side and two on the back as shown in Figure 2.2. Data from the front left camera, LiDAR, and vehicle position is recorded for experiments by driving along multiple areas at the UC San Diego campus. The camera data is streamed at approximately 13 Hz and the LiDAR scans at approximately 10 Hz. We drive through the campus to collect

Table 2.2: Quantitative evaluation on our labeled data for road, crosswalk and lane mark regions. Refer to section 2.3.2 for details.

Methods	IoU			mIoU	Accuracy		
	roads	crosswalks	lane marks		roads	crosswalks	lane marks
Vanilla	**0.715**	0.537	0.135	0.462	**0.797**	0.577	0.180
Vanilla+I	0.712	0.510	0.163	0.462	0.789	0.548	0.234
CFN	0.671	**0.605**	**0.301**	**0.526**	0.708	**0.696**	0.652
CFN+I	0.669	0.588	0.298	0.518	0.705	0.676	**0.657**

data for urban driving scenarios–including challenging scenarios like driving along steep hills, intersections and construction sites.

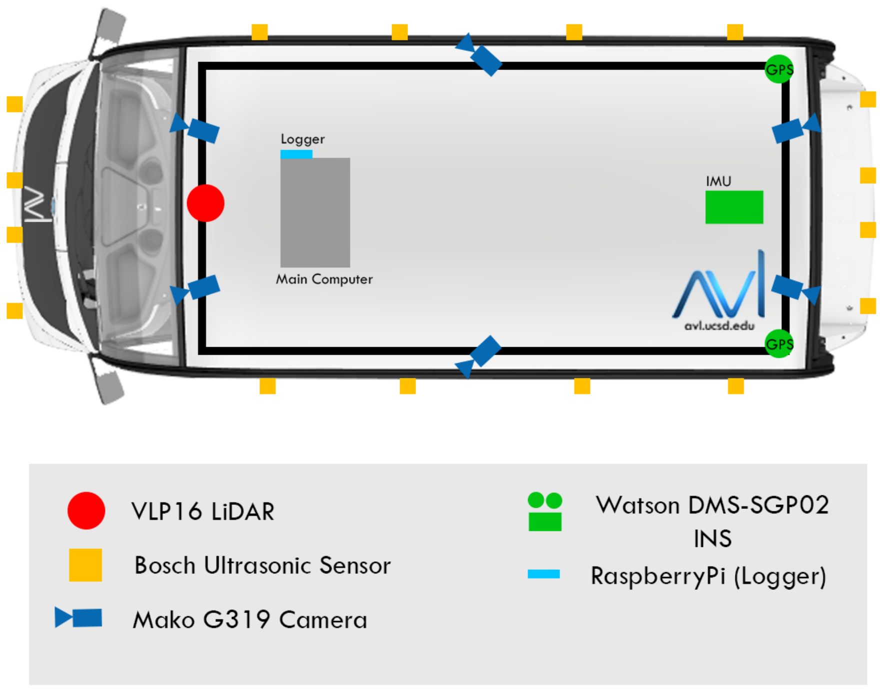


Figure 2.2: Vehicle Sensor Configuration

2.3.1 Image Semantic segmentation

Training Dataset

The Mapillary data set used comprises $18,000$ training images and $2,000$ validation images. We merge terrain into vegetation, different types of riders into the human category, traffic-sign-back and traffic-sign-front into traffic-sign, bridge into building, and different kinds of crosswalks into a single crosswalk class. The training dataset is augmented by random horizontal flips with 0.5 probability, random resize with the scale ranging from 0.5 to 2, and random crop. These images are also normalized to a distribution with mean of $(0.485, 0.456, 0.406)$ and standard deviation of $(0.229, 0.224, 0.225)$. Due to the similarity of the Mapillary dataset and our driving scenarios, as well as the intense data augmentation in the training process, we do not visually observe severe performance drop when testing it in the UC San Diego campus.

Hyperparameters

A batch size of 16 with synchronized batch normalization [ZSQ+17] is used to train the network for 200 epochs on eight 2080Ti GPUs with input image sizes of 640×640. The output stride of the network is eight. We use a SGD optimizer and employ a polynomial learning rate policy [CPSA17, ZSR+19] where the learning rate is $base\_lr \times (1 - \frac{epoch}{maxepoch})^{power}$ with 0.005 base learning rate and power=0.9. The momentum and weight decay are set to 0.9 and $4e^{-5}$, respectively.

Metrics

To quantify the performance of the network, the mean intersection-over-union (mIoU) metric is used. This metric measures the average perfomance at the pixel-level across the test samples. The mIoU of ResNeXt50 in the validation set is 68.32%. Compared with ResNet101, ResNeXt50's performance slightly decreases but requires much less memory (from $367MB$ to

210MB), which is preferable for our onboard hardware where the memory may be limited. The inference time of the network is approximately $0.2s$ per image.

2.3.2 Semantic Mapping

A region that spans 1.1 km of the UC San Diego campus is selected for evaluation in the map generation benchmark. An HD map has been manually annotated in this area. It contains road information such as crosswalks, sidewalks, and center of road lane definitions and has been tested in realistic environments. Based on these labels, we generate our semantic map with five channels: *road, crosswalk, lane marks, vegetation, and sidewalk* with resolution $d = 0.2$ meters. The labeling of HD map is not a trivial work. However, it exactly showcases the value of being able to automate the entire process.

The mIoU and pixel accuracy are used as the metrics of our evaluation. While mIoU reflects our model's segmentation recall and precision, it is worth noting these metrics are influenced by the sparsity of the LiDAR point cloud: the model output may be very accurate but may contain a lot of unclassified cells (holes). We address this problem by using a smoothing kernel to interpolate the missing labels on our map.

For all the experiments in Table 2.2, we clip the local dense point maps extracted up to 15m along the longitudinal axis and -10 to 10 m along the lateral axis of the vehicle as the semantic segmentation performance drops significantly for a longer range.

Modeling of Observation Uncertainty

We first validate the design of matrix $\mathbf{M}$ that models the observation uncertainty of the semantic label $\mathbf{z}_t$ given $\mathbf{S}_t$. For numerical stability reasons, we represent the elements of $\mathbf{M}$ logarithmically. We investigate two approaches: one is defined by $\mu(\mathbf{I} + \lambda\mathbf{I})$ where λ is a hyper-parameter and μ is a normalization factor: we refer to this model as Vanilla. Another approach is the confusion matrix of the semantic segmentation network in the Mapillary validation data set:

we define this as CFN. During the inference time, we assign each cell to the label with the highest probability. Quantitative results are shown in Table 2.2. Compared with Vanilla version, CFN has a significant improvement in IoU and pixel accuracy on the classes of crosswalks and lane marks. This indicates the advantage of utilizing the confusion matrix of the network to model the prediction error in the semantic segmentation and thus provides a better map generation result.

Integration with LiDAR Intensity

In order to utilize the fact that different materials on the road have different reflectivity, we first remove all the intensity data that is less than $k = 14$ (e.g. Figure 2.6); this normalized threshold was manually calibrated for a Velodyne VLP-16 LiDAR. Then during the semantic mapping step, if a label is predicted as lane marks, we increase its logarithmic probability by a constant factor γ. This essentially suppresses our prediction of other classes and increases our prediction confidence for lane marks. The models that contain the integration of intensity have a ”+I” in Table 2.2. For Vanilla+I, we observe better results with respect to Vanilla in terms of accuracy and IoU of the lane marks but slightly decreases for roads and crosswalks, indicating the benefit of intensity integration for lane mark prediction. This tendency, however, does not replicate for CFN+I with respect to CFN. To achieve further improvement, a more sophisticated function may be needed to model the LiDAR intensity.

An example of the global map generated by our CFN+I model for the entire test region is shown in Figure 2.7. A region of the map has been amplified in the figure, showing that our model can capture the static elements on the road clearly. Additional maps generated for wider regions are shown in Appendix A.1; these maps are utilized in Chapter 3.

2.3.3 Comparison to Sparse LiDAR Scan

One possible alternative for associating semantic images with depth information is to use the point cloud data generated by the LiDAR in real time. By following a similar mapping

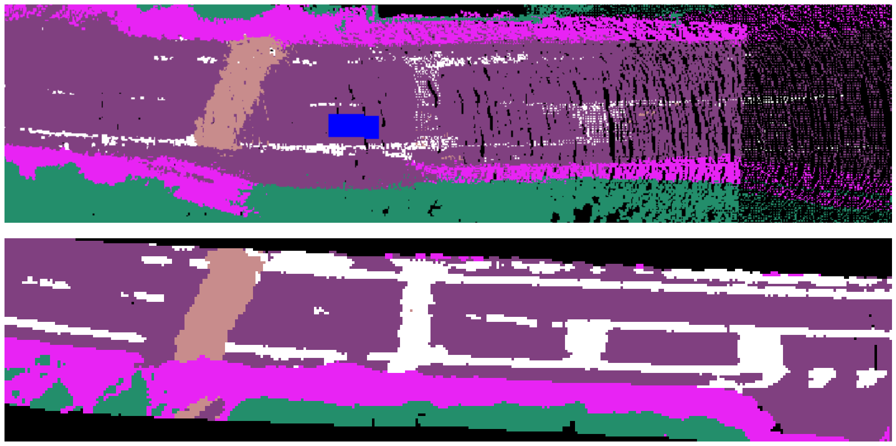


Figure 2.3: The map generation result of our algorithm. Top image is the local map and the blue car indicates the position of ego vehicles. Bottom image is the same region in global map.

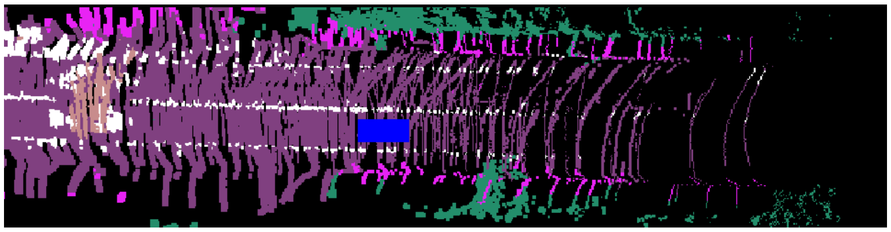


Figure 2.4: The semantic map generated from real time LiDAR scan. The black region on the scene indicates unknown area which is not detected by the LiDAR sensor.

approach, we project the point cloud onto the semantic image frame and build the semantic map. Figure 2.4 shows that this approach gives real-time performance. However, for the 16-channel LiDAR used, the point cloud scans are too sparse to construct a semantic map at greater distances. This becomes worse when the car drives faster. Therefore, a pre-built dense point cloud map allows us to construct semantic maps for longer ranges.

2.3.4 Comparison to Planar Assumption

Another method we explored is back-projecting the 2D semantic image into the 3D space using a homography with the assumption that the ground is flat. The back-projection approach leaves no black holes. Nevertheless, the planar assumption fails along steep hills or

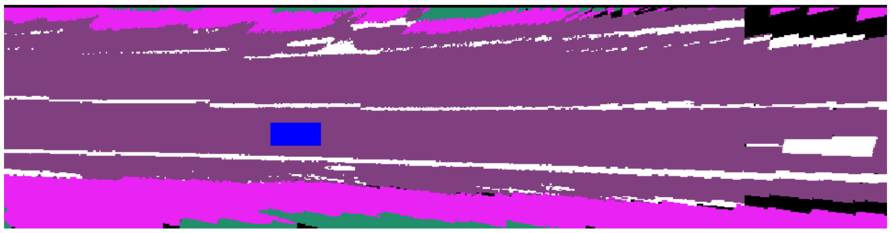


Figure 2.5: The semantic map generated by back-projecting 2D semantic image into the 3D space with planar assumption. The map presented in the vehicle frame. The distortion of lines indicates the failure case.

road intersections in urban driving scenarios as shown in Figure 2.5: this leads to considerable distortion at longer ranges.

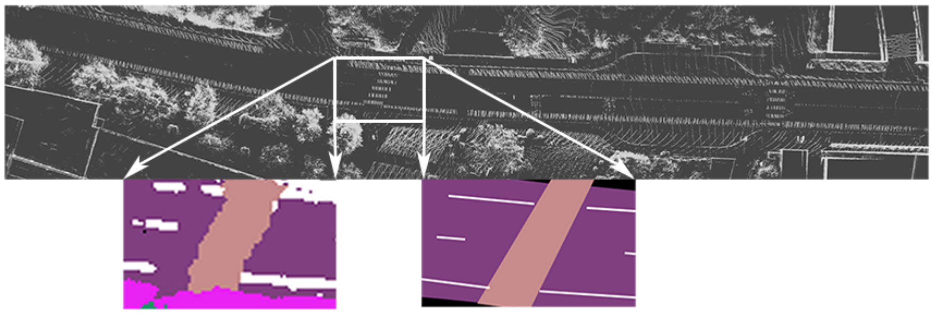


Figure 2.6: A visualization of our generated map (bottom left), the ground truth label (bottom right), and the LiDAR point cloud map (top). The point cloud map has been thresholded based on their intensity value. Any points that have intensity below a certain threshold k are discarded. With this threshold, we can clearly see the layout of the crosswalk and lane marks on the road.

2.3.5 Summary

By fusing the rich information from semantic labels on image frames, our comparisons to manually annotated maps indicate that this work effectively introduces a statistical method for identifying road features and localizing them in bird's eye view. This method can be extended for automating HD map annotation for crosswalks, lane markings, drivable surfaces, and sidewalks. These features can be incorporated for generating HD maps independently of predefined HD map formats with the additional extension of centerline identifications which are often used for path tracking algorithms. By accounting for the road network junctions and forks, future work involves the full automation of road network annotations that could leverage graphical methods. While a combination of the techniques proposed can potentially address the scalability drawbacks from HD maps, they also propose new areas of research on high-level dynamic planning. Currently, many autonomous driving architectures require dense point cloud maps for localization and come at a scalability and maintenance cost in a similar way that HD maps do. By dynamically estimating drivable surfaces, centerlines, lane markings and other road features, the notion of using centimeter-level localization could be removed as long as immediate actions can be extracted from a high-level planner. Initial work towards this direction is covered in Chapter 4–which focuses on real-time centerline detection in 2D and 3D without detailed scene maps.

Chapter 2 includes material from the paper published in Probabilistic Semantic Mapping for Urban Autonomous Driving Applications in International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, Oct 2020. David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik Christensen. The dissertation author was one of the primary co-authors and contributed to the implementation, design, and the dataset associated with the approach. Zhang contributed to the dataset, implementation, and evaluation of the semantic mapping module. Li contributed to the semantic segmentation design, training, and evaluation of the semantic mapping pipeline. Xiang contributed to the semantic mapping evaluation and formulation of the confusion matrix formulation.

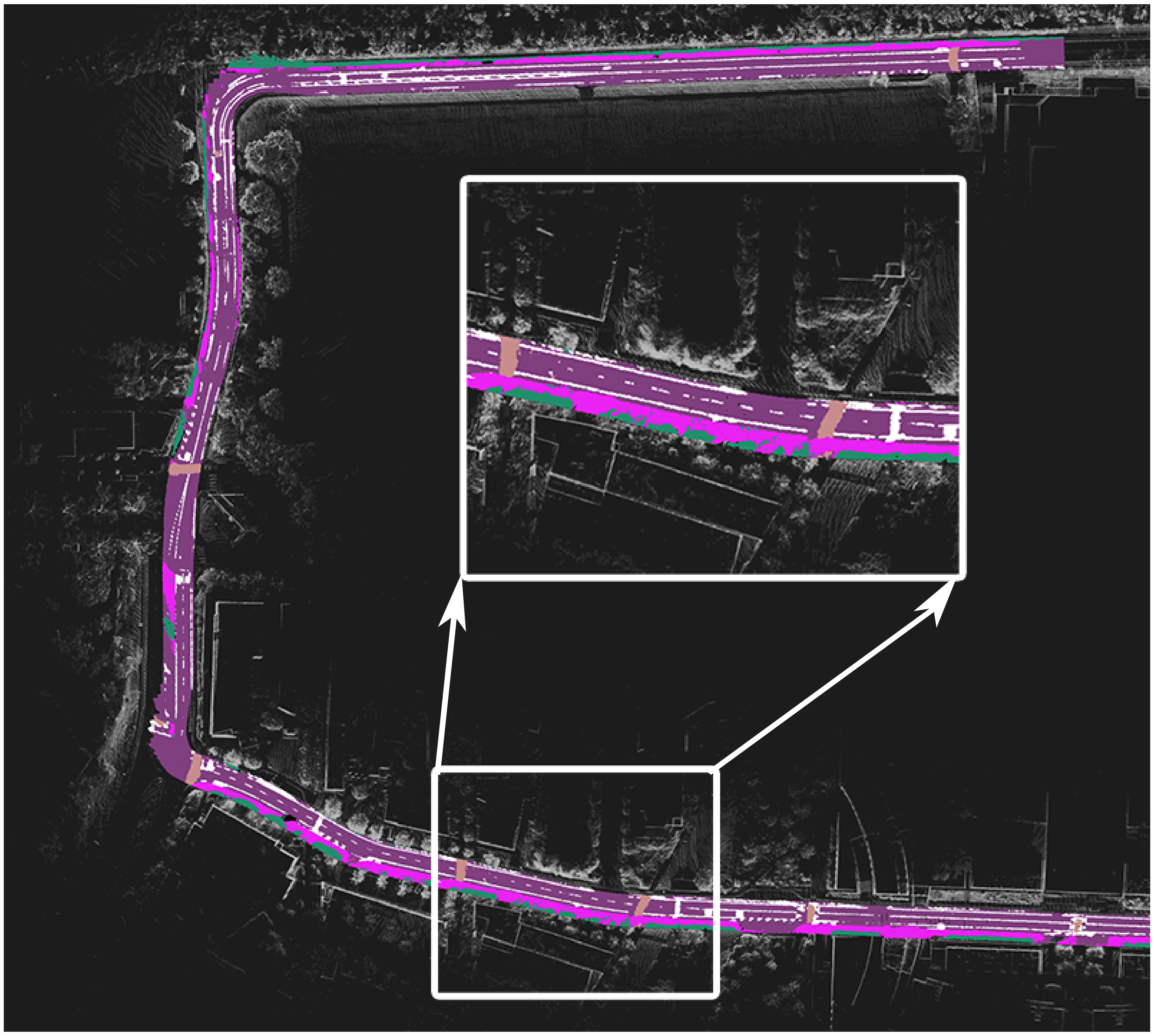


Figure 2.7: The BEV version of our generated map in the entire region of our testing data set. Our semantic map is displayed on top of the dense point cloud map. One segment of the map has been magnified to show the details of the map. Best view in color.

Chapter 3

Dynamic Trajectory Generation for Urban Environments

A fundamental component for path planning and navigation in autonomous driving systems involves mapping and localization. To generate sequences of control actions that can be executed by an autonomous system, global planners must formulate a navigation strategy that considers the position of the agent on the map and the reference paths needed to reach a goal. These strategies can be characterized by utilizing map information and standard graph search strategies such as A* [HNR68].

Global plans can be generated using different types of maps such as High Definition (HD) maps or coarse maps. The former are commonly used in modern autonomous driving architectures, providing detailed information about the road network, lane markings, crosswalks, and drivable areas. With complete road network and lane definitions, trajectory information is readily available during path planning, simplifying navigation, path tracking, and control. However, scalability is a challenge for large-scale deployments due to the manual or partially automated map labeling process, as well as the underlying maintenance process needed if construction displaces the original definitions.

In contrast, a coarse map representation for global planning can offer scalability advantages in dynamic environments over HD maps. Examples of coarse maps include open-source maps such as OpenStreetMaps (OSM) and proprietary maps like Google Maps. These lightweight maps generally provide road segment and intersection labels but lack lane and trajectory information. Therefore, a research objective entails dynamically identifying traversable trajectories by aligning coarse plan representations with real-time perception. These research directions can also enable autonomy applications with fewer priors that can generalize in areas that are constantly changing such as construction sites.

In this chapter, we introduce conditional generative models for generating dynamic ego-centric trajectories without manually annotated HD maps. Given a global plan $\mathbf{G}_t$ that represents the plan to reach a destination and a local scene representation $\mathbf{L}_t$ at time t that encodes the road features, lane markings and drivable areas, the task at hand becomes estimating a set of traversible waypoints $\mathbf{y} = \mathbf{y}_t, \mathbf{y}_{t+1}, \ldots, \mathbf{y}_{t+H}$ with respect to the ego-vehicle or a map frame– where H corresponds to the generation horizon. In the next three subsections, we describe the global plan approach and representation ($\mathbf{G}_t$), the local scene representation ($\mathbf{L}_t$), as well as the CVAE approach for modeling the distribution of feasible trajectories $p(\mathbf{y} \mid \mathbf{m})$, where $\mathbf{m} = (\mathbf{G}_t, \mathbf{L}_t)$. An important note to make here is that while obstacle detection, tracking, and motion planning are not covered in this work, the generated trajectory can facilitate the development of interfaces for these tasks. Hence, making it possible to estimate if an obstacle is blocking the road and if evasive maneuvers need to be taken. This is discussed in the conclusion and left for future work.

3.1 Related Work

In this section, we review related research on HD map generation. We then look at research related to two key components of our approach: lightweight maps and scene representations.

3.1.1 HD/Vector Maps

Over the years, many different methodologies have been developed for autonomous navigation in urban environments. Many of these methodologies use HD maps to facilitate the process of path planning. Examples of classical motion planning and control architectures that utilize HD maps include Autoware [KTI$^+$15] and Apollo [apo17]. These types of architectures have shown promising results for micro-mobility applications over relatively extended periods of time [CPZ$^+$21]. Similarly, end-to-end strategies [BKO18, ZLS$^+$, IER$^+$20, HYC$^+$23, ZLD$^+$21] have increasingly gained popularity in the research community. These methods generally rely on large training datasets [CKT$^+$21] or simulation environments [DRC$^+$17, RST$^+$20] for development and validation.

Recent work additionally explores building HD maps automatically to remove the limit of scalability imposed by human labeling. The data representations are often captured from aerial imagery or ground vehicles. Works from aerial imagery [BZT$^+$23] can be limited by the availability of images and will not be able to capture traffic signs; however, vehicles can naturally provide more effective coverage. Homayounfar et al. [HMLU18] and Zhou et al. [ZTTZ21] build lane-level HD maps in highway and urban areas, respectively. VectorMapNet [LYW$^+$23] proposes to predict vectorized representations directly, and MapTR [LCW$^+$23] improves prediction performance by using permutation invariant representations and loss. TopoNet [LCG$^+$23] further considers the association between lane-to-lane and lane-to-traffic elements (signs and traffic lights), which are essential elements to downstream tasks. However, in contrast to these efforts, our work focuses on the direct trajectory prediction task that is amenable for downstream applications instead of the mapping process itself.

3.1.2 Lightweight Maps

Methods that reduce the dependencies and priors on maps have also been an active area of research in recent years. The underlying representations that encode turn-by-turn directions for the global planning task are comparable to Google's proprietary maps or the publicly available OSM. For instance, generative models have been developed to encode the multimodal characteristics of driving implicitly [HDVG19] and explicitly [ARKR19] by utilizing coarse maps such as OSM [HDVG18]. An advantage of these representations is that the overhead associated with curating and updating maps can be reduced by relying more on raw sensor data such as camera image streams. Other related work involves utilizing a discrete action space such as "turn left," "turn right," or "go straight" to shift towards "mapless" strategies. This idea also focuses on minimizing the priors and reliance on maps; examples include Light Detection and Ranging (LiDAR) based methods [CSU21] and camera-based works [HSG+20] by using one-hot encoding to represent the desired action specifically for intersections.

3.1.3 Scene Representations

Recent developments in the semantic segmentation and sensor fusion literature have enabled methods for scene understanding that seek to build scene representations with highly detailed and accurate localization of road features without manual input. For example, [CLU+22, DMCD21] focuses on generating 2D bird's eye view semantic scene representations from monocular camera streams. While these present advantages for real-time scene understanding, they still present considerable limitations in terms of occlusions and localization errors of road futures. Alternative offline methods [PZL+] can address some of these limitations through a spatiotemporal fusion process that leverage camera and LiDAR fusion. Nevertheless, these efforts still require additional post-processing to extract lane-level trajectories that can be ingested by downstream modules. To address this challenge, our work seeks to align coarse representations (similar to those described in Chapter 2) with 2D bird's eye view semantic maps to estimate traversable lane-level trajectories. The motivation for utilizing coarse maps is centered around providing high-level global planning information with minimum priors and the use of semantic maps is to utilize scene representations that are accurate, up-to-scale, and that can be generated automatically.

3.1.4 Trajectory Forecasting

An important aspect of our work involves dynamic path generation to be capable of estimating feasible trajectories for the ego-vehicle. Our work is inspired by the latest developments on conditional generative models that have been applied for road user trajectory forecasting including pedestrians and vehicles. While prediction for other road users is out of the scope of this work, we leverage similar methods for trajectory generation. Examples of models utilized for prediction include LSTMs [AGR$^+$16], Conditional Variational Autoencoders (CVAEs) [ILSP20, SICP20], GANs [GJF$^+$18], Graph Convolution Networks (GCNs) [YYZ17], and most recently with Equivariant Continuous Convolutions (ECCO) [WLY21].

3.2 Representing Global Plans

Navigation requires choosing an optimal maneuver that aligns with a global plan and objective. An example of this involves intersection navigation: when the vehicle is approaching an intersection, there are multiple traversible trajectories but only one should be chosen and executed. This depends on global planning information that describes the actions that need to be taken to reach a destination given the current state of the vehicle. To capture this global plan information and context, we leverage OpenStreetMaps in a similar fashion as [ARKR19].

First, a global planner is implemented based on traditional graph search algorithms. This is performed by utilizing Global Navigation Satellite System (GNSS) traces to fetch and download open-source OSM data.[*] Each OSM is saved in an Extensible Markup Language (XML) format, which encodes map connectivity information. This format is parsed and post-processed to populate a graph with full road connectivity, where distance and road element information is preserved. In practice, we perform a projection from a geographic coordinate system into a Cartesian space by using the Haversine distance formula to approximate the relative distance on a sphere (earth). This approximation allows us to characterize a graph search strategy that is performed on a plane and assign weights based on units of meters, and also perform an egocentric transformation based on the position and orientation of the agent on the map. An example of this transformation can be seen in Figure 3.1, where we visualize the vehicle-centric rasterized (left) and vectorized (right) representations that are utilized in our work.

The rasterized representation is generated by defining a blank image canvas and drawing straight line segments that represent the road. In this representation, the road segments are represented by white lines and the planned road segments are denoted in green. The size of the image is $200px \times 200px$ with a $0.5m/px$ resolution. Given that this is a raster, the information provided in Cartesian space is discretized to convert the map information into an image and encoded using a Convolutional Neural Network (CNN) based encoder.

[*]`https://www.openstreetmap.org/`

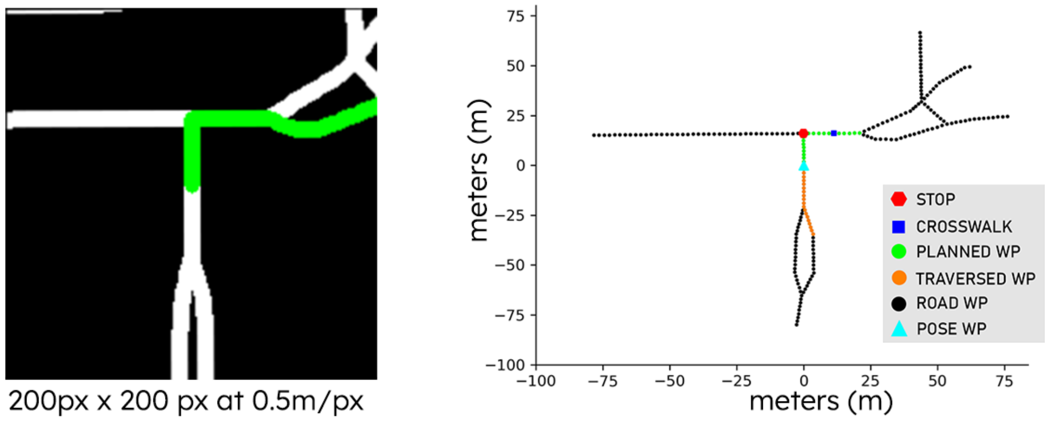


Figure 3.1: Global plan representations generated by our open-source navigation package. On the left, an image-based representation is shown and on the right, the graph-based representation is shown with various road features such as stop signs, crosswalks, and information about the history and planned trajectories.

In contrast, the graphical representation of the road network elements is directly extracted from the original graph by performing a local search within the vehicle location and state. The nearby elements are additionally decorated with various attributes from nearby features on a per-node basis such as stop signs, crosswalks, and traffic lights. This representation considers every node in the graph as a 3D vector–where the first two dimensions represent the 2D coordinates of the node element in an egocentric frame and the last dimension denotes if the node corresponds to a traffic signal, pedestrian crosswalk, or a stop sign. This compact global plan $g_m = \{(w_x^d, w_y^d, w_f^d)\}_{d=1}^{D}$ is represented by a 2D tensor of shape $D \times 3$, where $D = 40$ waypoints are utilized to represent the global plan. The first $\frac{D}{2}$ waypoints represent the trajectory directly behind the vehicle and the last $\frac{D}{2}$ waypoints represent the relative plan ahead of the vehicle. A self-attention mechanism [VSP$^+$17] is then applied on this input representation as shown in Equation 3.1; where $C = 3$ and $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are linear projections of the path. In Section 3.5.4, we perform an ablation study on the most relevant global planner features.

$$SA(\mathbf{g}_m) = \mathbf{g}_m + softmax\left(\frac{\mathbf{QK}^T}{\sqrt{C}}\right)\mathbf{V} \tag{3.1}$$

An important note in the design of our approach for the global planner is that the egocentric transformation is performed after a global plan is determined with respect to the map frame. This permits the egocentric representation to encode relative target information from a local perspective up to a fixed horizon, which applies to both representations: the rasterized and vectorized representations. Our implementation for the planner is open-source and implemented using the Robot Operating System (ROS) framework. [†]

3.3 Semantic Scene Representions

To estimate trajectories that a self-driving car can execute, sufficient contextual information must be gathered to represent the environment including lane markings, drivable areas and even side walks. To provide these contextual cues, we generate and utilize 2D semantic maps.

Semantic maps include information about road features that are critical for navigation like roads, lane markings, crosswalks and sidewalks. Our approach for generating semantic maps is based on the models proposed in our prior work [PZL$^+$] and Chapter 2. By fusing LiDAR and camera data, this approach is capable of automatically generating representations of the world with robust road features in a single pass of the scene without human annotation.

The algorithm consists of three parts, *i*) an image based semantic segmentation module, *ii*) a fusion and association module, and *iii*) a probabilistic mapping module as outlined in Figure **??**. The input camera data goes through a semantic segmentation neural network (Deeplab v3+ [CZP$^+$18]) with a light-weight backbone to generate labels for each pixel. LiDAR point clouds are then used to generate a dense point cloud map. We leverage localization to crop a portion of the point cloud map that is close to the vehicle. This cropped local dense map is then projected onto the semantic image to associate labels with each point. The point cloud with semantic labels is projected to 2D along the z-axis while performing a probabilistic update. A

[†]https://github.com/AutonomousVehicleLaboratory/gps_navigation

confusion matrix formulation is also applied (i.e. the error matrix associated with the semantic labels), to manage uncertainty and increase robustness.

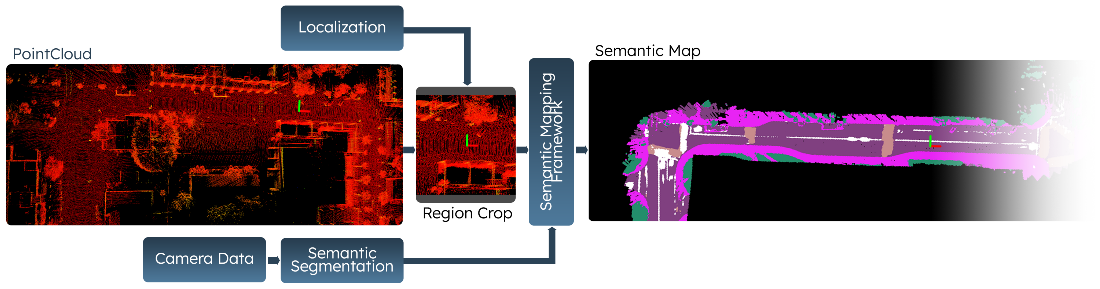


Figure 3.2: The figure represents the semantic mapping framework utilized to generate large 2D BEV semantic maps for the UC San Diego campus. It is composed of a camera-LiDAR fusion strategy that can aggregate multiple estimates across time and perform updates. The automatically generated maps can then be used with localization to provide egocentric representations for navigation applications.

To represent the 2D BEV semantic map, we use an occupancy grid-based representation with three dimensions: $(H \times W \times C)$. The height H and width W of the grid represent spatial dimensions, and the three channels C represent the color of different semantic classes, such as roads, sidewalks, and buildings. In practice, these maps are generated automatically offline and utilized during training with the known vehicle poses (estimated from localization). The position and orientation of the agent is used to extract local regions with respect to the rear-axle of the vehicle and perform a rotation such that the semantic features are aligned with the longitudinal axis of the vehicle, i.e. the front of the vehicle points up in the 2D BEV local semantic scene representation and the center of the image represents the rear-axle of the robot.

The map representation is scalable and can be continuously updated over time. At the same time the generated semantic map has labels that can be understood by humans. It is a challenge for the vehicle to automatically determine a path based on the pixel-level semantic map, thus we explore its capability of being potentially used for navigation.

Once a semantic map has been generated, localization can be leveraged to estimate the ego-vehicle pose ($\mathbf{T} \in SE(2)$) and extract local semantic representations that account for the translation and orientation of the vehicle. An example is shown on Figure 3.3, where the vehicle is represented by the red square and the front of the car always faces upward on the image frame.

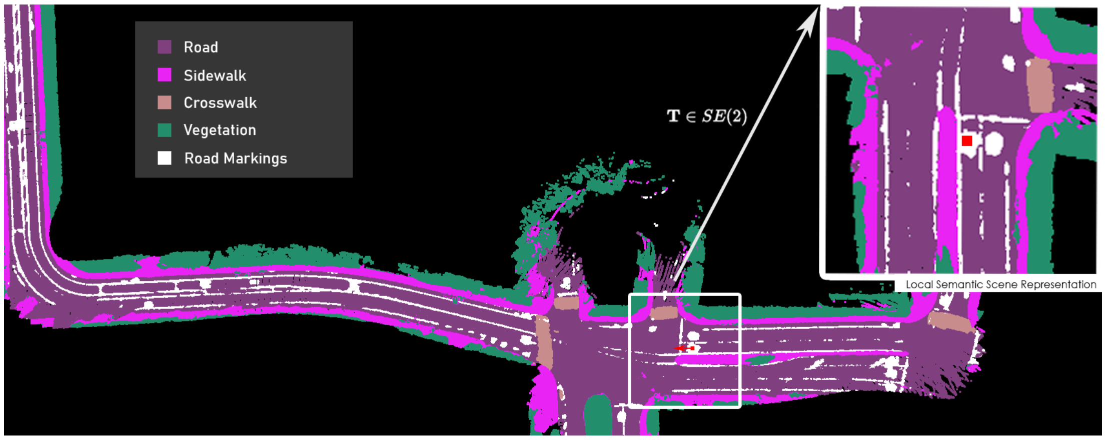


Figure 3.3: An example of the semantic map representation. An ego-centric representation is shown on the top right: red box denotes the rear-axle of the autonomous vehicle agent with an applied transformation **T**.

3.4 Generative Models for Dynamic Trajectory Generation

As previously introduced in Section 3.1.4, CVAEs have been used for estimating an external road user's future trajectory. One advantage of these models is that they allow us to explicitly encode the multi-modal aspects of trajectory prediction: based on a limited number of observations, a road user is likely to take multiple future trajectories. In other words, there is not a single best estimate in many cases.

In this section, we extend CVAEs for dynamic trajectory generation in the urban navigation task. Nevertheless, there are a few key differences between road user trajectory prediction and our approach. While trajectory generation is a multi-modal problem and the model must be capable of representing various modes for driving through intersections as an example, the intent and plan is known in advance and becomes deterministic when a high-level plan is given. In other words, the model should be capable of estimating the distribution of all possible modes and their corresponding trajectories but generate a single optimal trajectory given a high-level plan. In this context, we let $\mathbf{y} = \{(x^1, y^1), \ldots, (x^H, y^H)\}$ represent the lane-level trajectory of interest and $\mathbf{s}$ and $\mathbf{g}$ are the local semantic scene representation and the local global plan, respectively. To simplify notation and the derivation for a CVAE, we let $\mathbf{m} = \{\mathbf{s}, \mathbf{g}\}$ jointly represent the semantic and local plan information and rewrite the probability distribution as $p(\mathbf{y} \mid \mathbf{m})$.

A CVAE is a directed graphical model (Figure 3.4) that is formulated using an input variable $\mathbf{m}$, an output variable $\mathbf{y}$, and an approximation for the distribution $p(\mathbf{y} \mid \mathbf{m})$. To perform an approximation, a latent variable $\mathbf{z}$ is introduced and marginalized over all possible values as shown in (3.2), where $p(\mathbf{y} \mid \mathbf{m}, \mathbf{z})$ can be understood as a decoder and $p(\mathbf{z} \mid \mathbf{m})$ as an encoder. However, one problem arises while estimating $p(\mathbf{z} \mid \mathbf{m})$ as this requires inference of $p(\mathbf{m})$ as shown in (3.3).

To mitigate the intractable representation of the posterior $p(\mathbf{z} \mid \mathbf{m})$, a *recognition* model $q(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$ is introduced as an approximation that can be learned rather than treating it as a closed-form formulation.

$$p(\mathbf{y} \mid \mathbf{m}) = \sum_{\mathbf{z} \in \mathbf{Z}} p(\mathbf{y} \mid \mathbf{m}, \mathbf{z})\, p(\mathbf{z} \mid \mathbf{m}) \tag{3.2}$$

$$p(\mathbf{z} \mid \mathbf{m}) = p(\mathbf{m} \mid \mathbf{z}) \cdot \frac{p(\mathbf{z})}{p(\mathbf{m})} \tag{3.3}$$

Similar to [ILSP20] and [SICP20], importance sampling can be performed by multiplying (3.2) with $q(\mathbf{z} \mid \mathbf{m}, \mathbf{y}) / q(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$ and rewriting the sum as an expectation over $q(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$:

$$p(\mathbf{y} \mid \mathbf{m}) = \mathbb{E}_q \left[\frac{p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z})\, p_\theta(\mathbf{z} \mid \mathbf{m})}{q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})} \right] \tag{3.4}$$

For numerical stability reasons, during training the log-likelihood of $p(\mathbf{y} \mid \mathbf{m})$ is optimized as shown in (3.5) and due to the complexity of the right hand expression, Jensen's inequality is applied as shown in (3.6). Furthermore, the expression to be maximized can be rewritten using Kullback-Leibler divergence (3.7). The probability distributions are parameterized by different weights and are represented by subscripts ψ, ϕ, and θ.

$$\log p(\mathbf{y} \mid \mathbf{m}) = \log \mathbb{E}_q \left[\frac{p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z})\, p_\theta(\mathbf{z} \mid \mathbf{m})}{q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})} \right] \tag{3.5}$$

$$\log \mathbb{E}_q \left[\frac{p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z})\, p(\mathbf{z} \mid \mathbf{m})}{q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})} \right] \geq \mathbb{E}_q \left[\log \frac{p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z})\, p_\theta(\mathbf{z} \mid \mathbf{m})}{q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})} \right] \tag{3.6}$$

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{m}) &\geq \mathbb{E}_q \left[\log p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z}) \right] - \mathbb{E}_q \left[\log q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y}) - \log p_\theta(\mathbf{z} \mid \mathbf{m}) \right] \\ &= \mathbb{E}_q \left[\log p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z}) \right] - \mathbb{KL} \left[q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y}) \| p_\theta(\mathbf{z} \mid \mathbf{m}) \right] \end{aligned} \tag{3.7}$$

While training, $p_\theta(\mathbf{z} \mid \mathbf{m})$, and $q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$ are jointly optimized and the latent variables $\mathbf{z}$ are sampled from $q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$. As previously derived in (3.4)-(3.7), we seek to maximize (6) or in other words minimize $\mathcal{L}_{\text{CVAE}} = -\mathbb{E}_q \log p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z})] + \mathbb{KL} \left[q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y}) \| p_\theta(\mathbf{z} \mid \mathbf{m}) \right]$. While this formulation alone can capture the multi-modal aspects of intersection navigation, we further improve the performance by incorporating a Mean-Squared-Error (MSE) loss term. Hence, the loss is given by:

$$\mathcal{L} = -\mathbb{E}_q[\log p_\phi(\mathbf{y} \mid \mathbf{m}, \mathbf{z})] + \mathbb{KL} \left[q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y}) \| p_\theta(\mathbf{z} \mid \mathbf{m}) \right] + \frac{1}{H} \sum_{i=1}^{H} \| \mathbf{y}_i - \hat{\mathbf{y}}_i \|^2 \tag{3.8}$$

On the other hand, during testing the model can estimate the complete distribution $p(\mathbf{y} \mid \mathbf{m})$ by explicitly computing (3.2) or the distribution $p(\mathbf{y} \mid \mathbf{m}, \mathbf{z}^*)$ that corresponds to the latent variable with the highest probability:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} p_\theta(\mathbf{z} \mid \mathbf{m}) \tag{3.9}$$

Here $\mathbf{z}^*$ can be understood as the mode that best describes a deterministic trajectory.

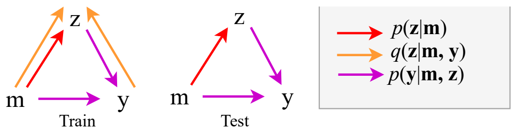

z
m
y
Train
z
m
y
Test
$p(\mathbf{z}|\mathbf{m})$
$q(\mathbf{z}|\mathbf{m}, \mathbf{y})$
$p(\mathbf{y}|\mathbf{m}, \mathbf{z})$

Figure 3.4: CVAE graphical model during training and testing.

This is intuitive from the perspective of having various traversible trajectories for intersection navigation; however, by conditioning **z** on **m** – the global plan and the local scene representation – a single output is generated.

With the global plan, the local scene representation and the CVAE formulation in place, we focus on the implementation details for model. First, the global plan and the local scene representation are processed using two different types of encoders as shown in Figure 3.5. The local scene representation utilizes a sequence of CNN layers, while the global planner can use a CNN or a Self-Attention encoder. The dimensionality of these representations is further reduced using fully connected layers.

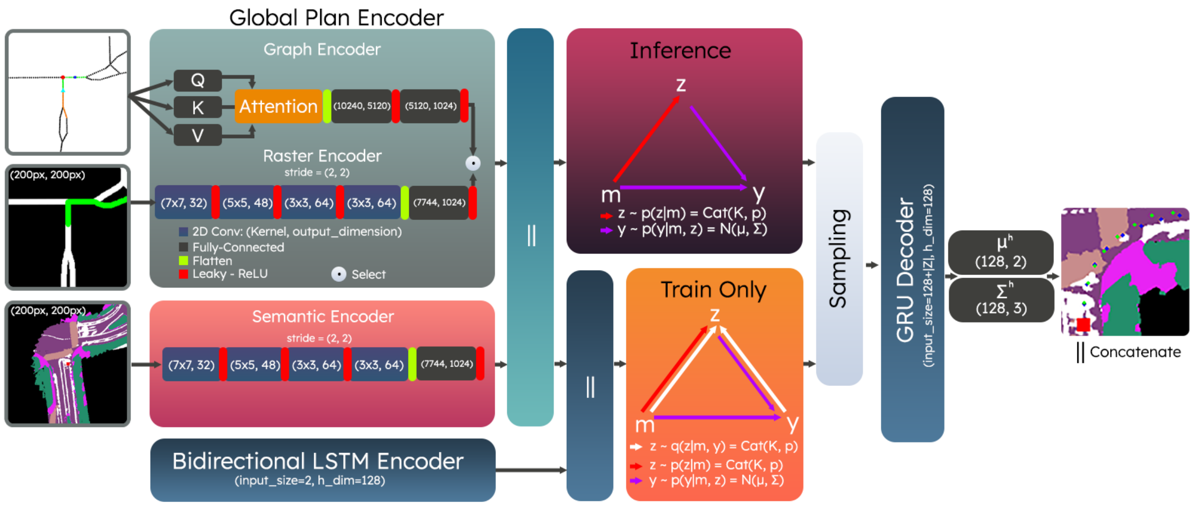


Figure 3.5: The CVAE network architecture is presented with the various components. The global plan encoders can process graph-based or rasterized representations. Similarly, the semantic scene representation is encoded using a sequence of CNN layers. Both representations are jointly used as conditionals in the training and test process.

Given that the *recognition* model $q_\Psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$ is dependent on the ground truth trajectories

($\mathbf{y}$) during training, a bidirectional LSTM [HS97] is used to encode the next H ground-truth positions. This embedding is then combined with the map representation embedding using another fully connected layer. The latent variable $\mathbf{z}$ that is sampled from $q_\psi(\mathbf{z} \mid \mathbf{m}, \mathbf{y})$ and $p_\theta(\mathbf{z} \mid \mathbf{m})$ is assumed to be drawn from a categorical distribution.

Once $\mathbf{z}^*$ is determined following (3.9), $p(\mathbf{y} \mid \mathbf{m}, \mathbf{z}^*)$ is assumed to be a multivariate normal that is parameterized with a gated recurrent unit (GRU) [CvMG$^+$14]: each multivariate normal is characterized by parameters (μ_t, Σ_t) that are decoded from $t \to t+H$. Both graphical models that correspond to the training and testing process are shown in Figure 3.4.

3.5 Experiments and Data

The approach utilizes synchronized global plan and local semantic scene representations to condition the CVAE. The synchronized data samples are generated as part of a data collection process and can be visualized in Figure 3.6. First, we set a destination using the OSM planner described in Section 3.2. This planner utilizes GNSS and inertial measurement unit (IMU) data to estimate and correct the position of the ego vehicle over time and perform the necessary egocentric transformations. The output generated from the planner consists of a 2D raster and the graph representation (Figure 3.1). Similarly, a local semantic scene representation is generated in an egocentric frame by utilizing a LiDAR-based localization method. These representations are extracted from full semantic maps. Both representations are synchronized based on nearest timestamps. The intrinsic and extrinsic parameters that are used for estimating egocentric transformations are estimated offline utilizing standard calibration methods such as checkerboards and plane fitting methods with feature matching strategies. Additional details on the process for calibration and localization can be found in [CPZ$^+$21].

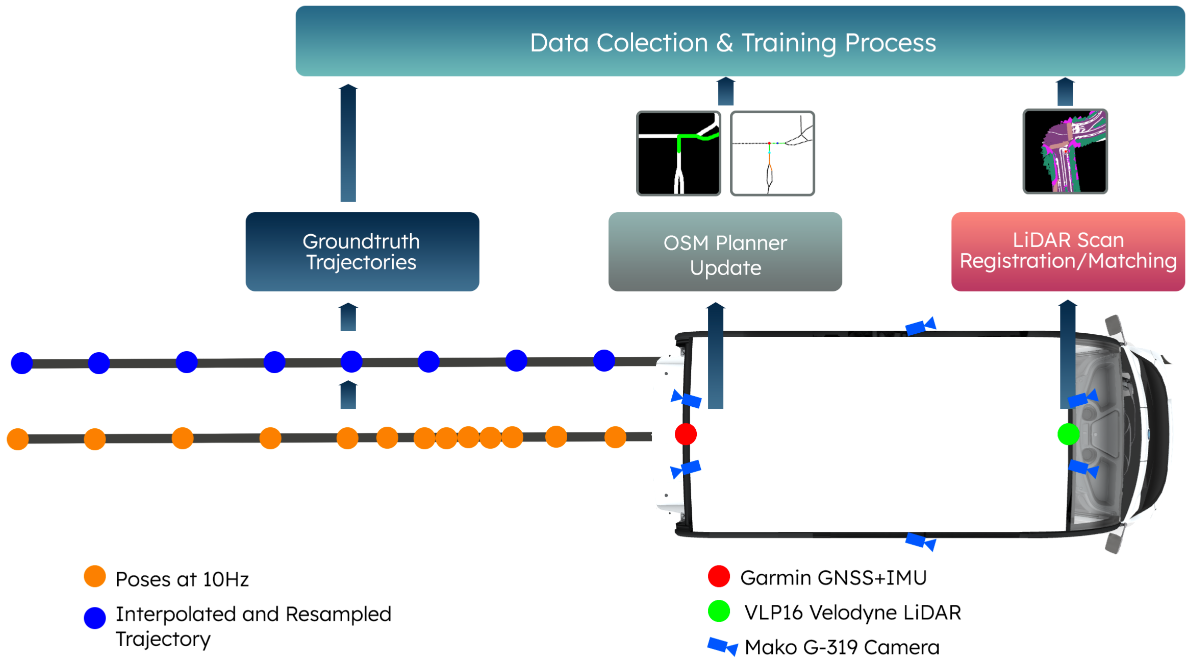


Figure 3.6: The figure outlines the data generation process for the global plan and semantic scene representations. The OSM planner generates egocentric global plans using a combination of GNSS and IMU. On the other hand, LiDAR-based scan matching is utilized to localize precisely and extract 2D egocentric semantic maps.

3.5.1 Datasets

With the data collection process described, various datasets are curated at the UC San Diego campus. The NominalScenes dataset [PZC22] includes urban driving data with global plan and semantic maps in various driving scenarios such as curved roads, loops, and intersections. On the other hand, the IntersectionScenes dataset [PXLC22] includes global plans and semantics for intersection-specific scenarios such as three-way and four-way intersections. We use these datasets to train and test the performance of our methods. We additionally make our implementation and benchmark publicly available.[‡] Visualizations of the point cloud maps used in the semantic mapping process are shown in Figure 3.7 alongside the output semantic maps.

[‡]`https://github.com/AutonomousVehicleLaboratory/coarse_av_cgm.git`

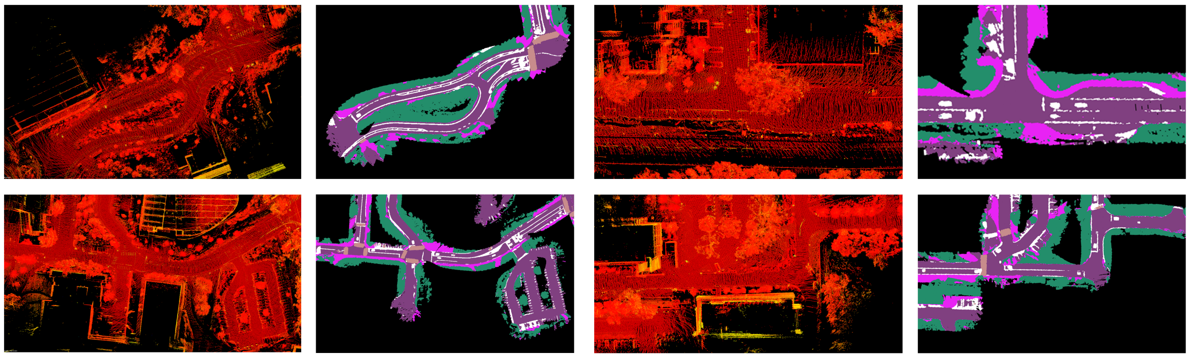


Figure 3.7: The visualizations in a map frame perspective correspond to the pointcloud maps used to generate 2D BEV semantic maps for various parts of a university campus. The pointcloud maps are shown in the first and third columns, and the associated 2D semantic maps are shown in the second and fourth columns. The camera-LiDAR fusion process is outlined in Chapter 2.

3.5.2 Experiments

To train and quantify the performance of the approach, we use the reported ego-vehicle poses to characterize trajectories. However, since the position and the orientation of the vehicle is reported at $10Hz$, the data points recorded during data collection can be inconsistently spaced apart. Hence, we interpolate as noted in Figure 3.6 and sample new trajectories during training and evaluation. This allows us to sample trajectories with varying waypoint density and analyze the limitations of each.

3.5.3 Metrics

We measure the quality of each trajectory produced based on two criteria: Driveable Area Compliance (DAC) and Displacement Error (DE). DAC evaluates the model's capacity to generate trajectories that stay within drivable regions. This measurement is derived by averaging the trajectories that coincide with drivable areas, including crosswalks, lane markings, and road surfaces as defined in the local semantic map. If any waypoint of a trajectory overlaps with a sidewalk or vegetation area, it is deemed non-compliant. We present the error associated with half of the trajectory (DAC$_{HALF}$) as well as the entire predicted trajectory (DAC$_{FULL}$); as shown in Equations 3.10- 3.16. Both metrics characterize compliance in a range between $[0,1]$ by utilizing an indicator function to evaluate if a waypoint prediction $\hat{y}_i^h$ lies within the semantic set of drivable regions C_d.

We additionally employ metrics commonly used in the field of road user prediction research to characterize the performance of trajectories: Average Displacement Error (ADE) and Final Displacement Error (FDE) [GJF$^+$18, FDSF19]. These metrics allow us to assess the average error of each trajectory across all H waypoints (ADE) and the average error specifically related to the last predicted waypoint (FDE). To provide a more comprehensive analysis, we extend the evaluation of ADE by measuring the error associated with half of the trajectory (ADE$_{HALF}$), considering that the waypoints closest to the autonomous agent are executed first during navigation. The worst-case errors are also captured by calculating the average maximum displacement error (MDE) along each predicted trajectory.

$$DAC_{\text{FULL}} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{H}\sum_{h=1}^{H} s(\hat{y}_i^h) \tag{3.10}$$

$$DAC_{\text{HALF}} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\left\lfloor\frac{H-1}{2}\right\rfloor}\sum_{h=1}^{\left\lfloor\frac{H-1}{2}\right\rfloor} s(\hat{y}_i^h) \tag{3.11}$$

$$s(\hat{y}_i^h) = \begin{cases} 1, & \text{if } \hat{y}_i^h \in C_d \\ 0, & \text{else} \end{cases} \tag{3.12}$$

$$ADE_{\text{FULL}} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{H}\sum_{h=1}^{H} \left\| y_i^h - \hat{y}_i^h \right\| \tag{3.13}$$

$$ADE_{\text{HALF}} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\left\lfloor\frac{H-1}{2}\right\rfloor}\sum_{h=1}^{\left\lfloor\frac{H-1}{2}\right\rfloor} \| y_i^h - \hat{y}_i^h \| \tag{3.14}$$

$$FDE = \frac{1}{n}\sum_{i=1}^{n} \left\| y_i^H - \hat{y}_i^H \right\| \tag{3.15}$$

$$MDE = \frac{1}{n}\sum_{i=1}^{n} \max_{h} \left\| y_i^h - \hat{y}_i^h \right\| \tag{3.16}$$

3.5.4 Results and Discussion

For all the experiments performed, we set the number of discrete states within the CVAE to 12, i.e. $|\mathbf{Z}| = 12$. The rationale for choosing 12 as the number of distribution modes is not only motivated by performance results but also by the various navigation behaviors including i) making left turns, ii) making right turns, iii) lane following, iv) driving straight across intersections, v) driving along curved roads, and vi) making u-turns. This equates to utilizing two modes to model each driving behavior explicitly.

Ablation Study: Waypoint Density

In our initial experiments, we compare trajectory predictions with varying waypoint densities up to a $30m$ horizon. The first type **H10** is defined by 10 waypoints spaced $3m$ apart and the second **H15** defines 15 waypoints spaced $2m$ apart. The performance of these two characterizations of the model are shown in Table 3.1. From the results, we observe that increasing the number of waypoints regressed also increases the likelihood of encountering compound errors. We hypothesize that this is due to the dependency of the next waypoint prediction on the previous cell state within the GRU decoder of the network. Therefore, for the remainder of the experiments, we utilize **H10** as the underlying trajectory representation.

Ablation Study: Rasterized and Graph Representations

To further evaluate the performance implications of the underlying global plan representations, we compare the rasterized and the graph based global plan representations discussed in Section 3.2. In Table 3.2, we evaluate four different models using the NominalScenes dataset and benchmark. The first method evaluated utilizes the rasterized global plan, and the last three leverage the graphs directly while aggregating various node features. For graphical models, we analyze the value of incorporating information about the road segments traversed (P), the planned road segments to be traversed (F), traffic signals and stop signs (S), and crosswalks (C). In these experiments, we observe that the attention mechanism that makes use of all node features except crosswalks outperforms the rasterized method and the methods that only make partial use of the node features. However, we note that not all the node features necessarily boost performance. For example, crosswalk features slightly deteriorate performance when we compare the two models that make use of the full history, planned trajectory, and traffic signals and stop sign features. While this may be unexpected, we find that OSM crosswalk information is not unique to intersections and as a result may present difficulties distinguishing between intersections with crosswalks and straight road segments with crosswalk features. On the other hand, this is not the case when we incorporate stop signs and traffic signals, since they are consistently defined at intersections.

Table 3.1: The comparison between the **H10** and **H15** models is presented. Rasterized global plan representations are used to evaluate the performance of the models using the NominalScenes dataset.

Method	**ADE**$_{FULL}$ $\downarrow$	**ADE**$_{HALF}$ $\downarrow$	**FDE** $\downarrow$	**MDE** $\downarrow$
OSM - **H10**	**1.056245**	**0.336941**	**2.447714**	**2.494614**
OSM - **H15**	2.341875	0.753802	6.127183	6.177646

Table 3.2: The results of an ablation study between rasterized and graph-based representations are presented. These results include a comparison of the graph-based models as a function of different types of node features. The experiments are performed with the NominalScenes dataset.

Method	**ADE**$_{FULL}$ ↓	**ADE**$_{HALF}$ ↓	**FDE** ↓	**MDE** ↓	**DAC** ↑	**DAC**$_{HALF}$ ↑
OSM Raster	1.056245	**0.336941**	2.447714	2.494614	0.849162	0.934218
OSM ATT w/ PF	1.365685	0.538815	2.852894	3.047669	0.892321	0.933054
OSM ATT w/ SPF	**0.969206**	0.353576	**2.316740**	**2.393168**	**0.914869**	0.944642
OSM ATT w/ SCPF	1.131581	0.388303	2.717636	2.795832	0.905864	0.942408

In a subsequent set of experiments, we evaluate the performance of our methods for intersection-specific scenarios using the IntersectionScenes dataset; these scenarios include left turns, right turns, and driving straight across intersections for three-way and four-way intersections. The results consistently show that explicitly utilizing the graphical representation can boost performance.

Table 3.3: The graph-based and rasterized plan representations are evaluated an intersection-specific dataset, IntersectionScenes. This includes three-way and four-way intersections for urban driving scenarios specifically.

Method	**ADE**$_{FULL}$ ↓	**ADE**$_{HALF}$ ↓	**FDE** ↓	**MDE** ↓	**DAC** ↑	**DAC**$_{HALF}$ ↑
OSM Raster	1.793062	0.673450	3.672231	3.728120	0.858367	0.913147
OSM ATT w/ SPF	**1.511415**	**0.619056**	**3.087722**	**3.226302**	**0.898473**	**0.924834**

On average, we observe an $1.5m$ error across the full trajectory generated, and an average error of $3.1m$ associated with the last waypoint of every trajectory predicted. We additionally find that the worse case predictions average to $3.2m$. Similarly, DAC indicates that 90% of the trajectories estimated overlap with a drivable region: including pavement, crosswalks, and lane marks.

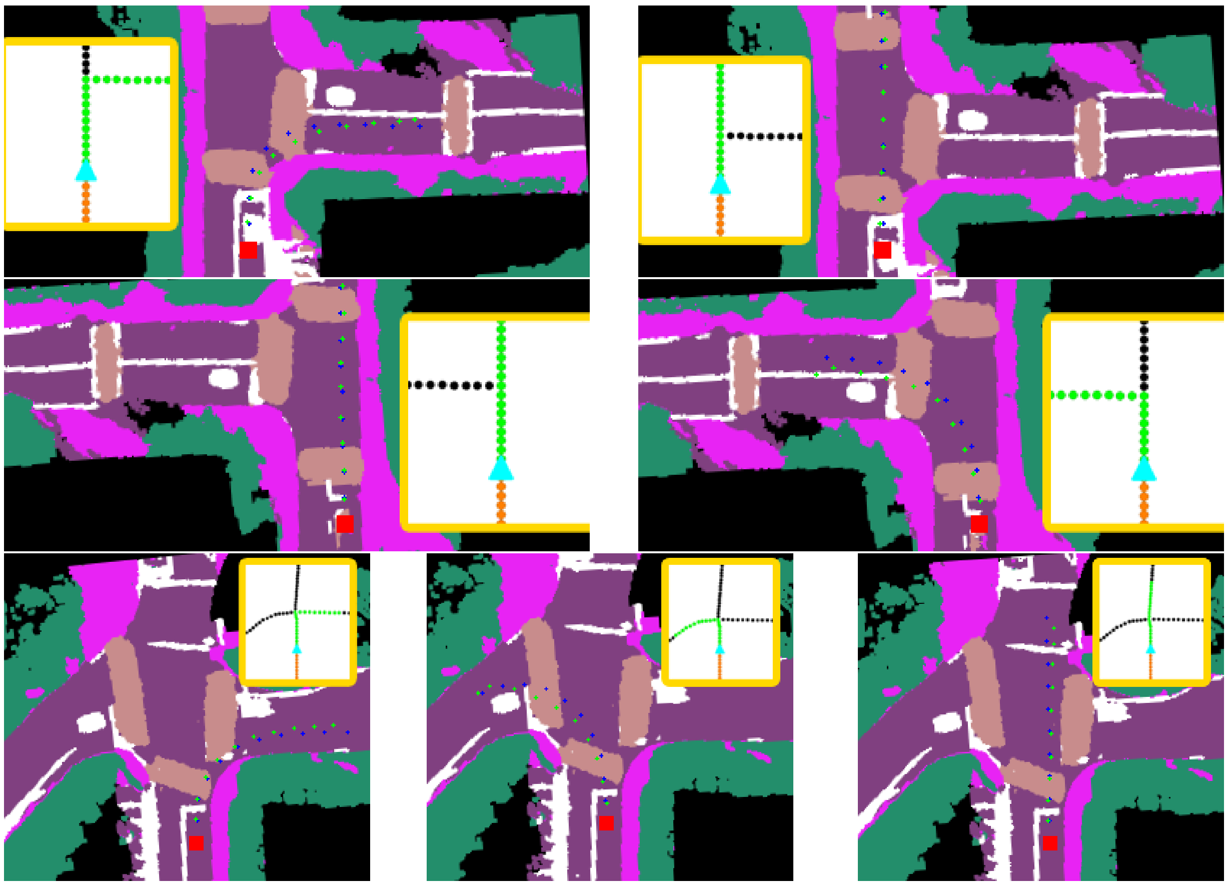


Figure 3.8: Various visualizations from the IntersectionScenes dataset using the SPF graph encoder. Each row represents an intersection with different global plans (yellow boxes). Groundtruth labels are denoted by blue waypoints and predictions by green waypoints.

In contrast, the errors associated with the first half of each trajectory predicted are considerably lower. These results are particularly more relevant given that a motion planner will utilize the nearby waypoints first. For instance, in an urban scenario with a $11m/s$ ($25mph$) speed limit, the ego-vehicle can utilize the first $H/2$ waypoints to formulate a trajectory that spans $15m$ without needing to replan or generate a new estimate before the $1s$ mark. Even though a motion planner would still execute collision checking as part of a downstream process, this illustrates the potential in real-time applications for urban driving. In fact, we find that our approach can keep up with real-time compute requirements by achieving an inference time of approximately 6.18*ms* using the attention-based encoder and 6.22*ms* using the CNN-based encoder. As an added benefit, the self-attention mechanism uses 31% fewer parameters than the CNN-based encoder, which comprises 16.8*M* learnable parameters.

A number of visualizations from various intersection scenarios is shown in Figure 3.8. These visualizations are generated using the SPF graph-based model and represent prediction outputs from three-way and four-way intersections that capture the multimodal properties of the CVAE.

3.5.5 Summary

In this work, a method for aligning coarse global plan representations with semantic scene models was explored. Various datasets and benchmarks with open implementations for the planner and CVAE formulation are made available. The contributions indicate potential use cases for urban driving navigation with fewer map priors, such as the widely used HD maps. This additionally presents directions for unifying dynamic path generation strategies with existing frameworks that leverage offline maps in combination with map detection methods. Towards this end, Chapter 4 introduces methods that aim to reduce the complexity of the semantic scene models by directly predicting centerlines in 2D and 3D.

Chapter 3 includes material from three papers. The first paper corresponds to TridentNet: A Conditional Generative Model for Dynamic Trajectory Generation in Intelligent Autonomous Systems-16, Singapore, June 2021. David Paz, Henry Zhang, and Henrik I Christensen. The second paper corresponds to Tridentnetv2: Lightweight Graphical Global Plan Representations for Dynamic Trajectory Generation in International Conference on Robotics and Automation, Philadelphia, May 2022. David Paz, Hao Xiang, Andrew Liang, and Henrik I Christensen. The third paper corresponds to Conditional Generative Models for Dynamic Trajectory Generation and Urban Driving in Sensors2023, July 2023. David Paz, Hengyuan Zhang, Hao Xiang, Andrew Liang, and Henrik I Christensen. The dissertation author was the primary co-author and contributed to the implementation, design, datasets, and evaluation of the approach. Zhang contributed to the dataset associated with the experiments. Xiang and Liang contributed to the global planner implementation.

Chapter 4

Centerline Networks

Open-source contributions have been instrumental in accelerating the development of autonomous vehicle technology in recent years. Various contributions come in the form of open-source datasets like Argoverse2 [WQA$^+$21], Waymo Open Dataset [SKD$^+$20], and Nu-Plan [CKT$^+$21]. These datasets contain meticulously labeled data, catering to various tasks such as detection, tracking, prediction, and planning.

A core component in many of these datasets is the map information–which defines lane-level details and road network connectivity as discussed in Chapter 2. This data is used in prediction models to provide context and facilitate trajectory generation and path planning to characterize traversable and obstacle free trajectories. However, the modules that utilize map data make the assumption of a static world, which can lead to challenges when deployed in real-world scenarios with changing road conditions due to construction or other factors. Hence, as the operation design domain grows and scalability requirements become more important, the need for dynamic and real-time scene understanding becomes evident.

To begin tackling these challenges, an automatic semantic mapping framework is characterized in Chapter 2. These semantic maps effectively capture the contextual information in-scene by leveraging a camera-LiDAR fusions approach to identify drivable regions, lane marks, sidewalks, vegetation, and crosswalks. One limitation however involves the computational complexity of semantic segmentation in real-time computation. Hence, the mapping process is performed offline. While the generated maps can be utilized for downstream tasks in real-time to provide context, an open research remains that involves the identification of real-time methods that are lightweight and can jointly provide the necessary information for prediction and motion planning tasks.

To explore strategies that can reduce this gap, this chapter explores methods that focus directly on identifying relevant features in the scene that are needed for the trajectory generation task. One of those features relevant for the driving task, specifically for urban scenarios, include the centerline tracks. In the presence of lane boundaries, a centerline corresponds to the virtual line that falls in between the left and right lane boundaries of a lane. However, in the absence of lane boundaries, such as intersections, the centerlines would be the line segments that connect two lanes on opposite sides of an intersection.

To this end, we review a number of related research work in Section, and propose a strategy for centerline prediction in 2D and 3D in Section. The formulation includes an approach for automatic centerline label generation. This pipeline is leveraged to generate over 900,000 labels with 2D and 3D feature correspondences. We find that providing additional context during the training process can help resolve issues related to severe to complete occlusion. These experimental results and results are described in Section.

4.1 Related Work

Lane detection and semantic segmentation has remained an active area of research for decades. Early work for autonomous driving applications relied on traditional edge detection techniques, as described in [Dic07]. However, with the introduction of learning-based methods for lane detection and segmentation tasks, a number of methods have been developed in recent years. These methods cover semantic segmentation as well as key-point and parametric-based approaches.

The semantic segmentation task consists of a per-pixel prediction task, in which every pixel from an image is classified into different classes. Public datasets including Mapillary [NORBK17] and Cityscapes [COR$^+$16] have enabled multiple open-source contributions in the semantic segmentation task that leverage encoder-decoder architectures and spatial pyramid pooling modules [CPSA17] and attention mechanisms to resolve fine grain details [TSC20a]. These methods are capable of resolving small details. However, semantic segmentation can present a large computational overhead during training and inference time, which can limit the use cases for real-time scene understanding.

Another approach for inferring drivable regions and trajectories involves key-point and spline prediction for lanes directly. Examples of real-time image based key-point detection include PINet [KLA$^+$20]. In a similar way, the lane detection tasks have been extended to include parameterized spline representations such as Bezier curves [FGT$^+$22] that compress a lane representation by using a fixed number of control points. However, these methods can struggle when recovering irregular lanes and sharp curves that may not be fully characterized by a finite number of control points. More recently, the lane detection task has also been extended to 3D [BCF$^+$22] by using synthetic [GCZ$^+$20] and real-world datasets [CSL$^+$22].

In contrast to existing work, the work covered in this section focuses on directly predicting centerline key-points as shown in Fig 4.1. A key benefit of directly predicting centerlines involves downstream applications and post-processing. In practice, the centerlines of each lane provide reference trajectories that can be directly utilized by motion planners. Standard methods for detection and segmentation of lanes still infer an overhead for centerline extraction and reasoning depending on the context of the scene and whether a bounded lane mark boundary is traversable or not.

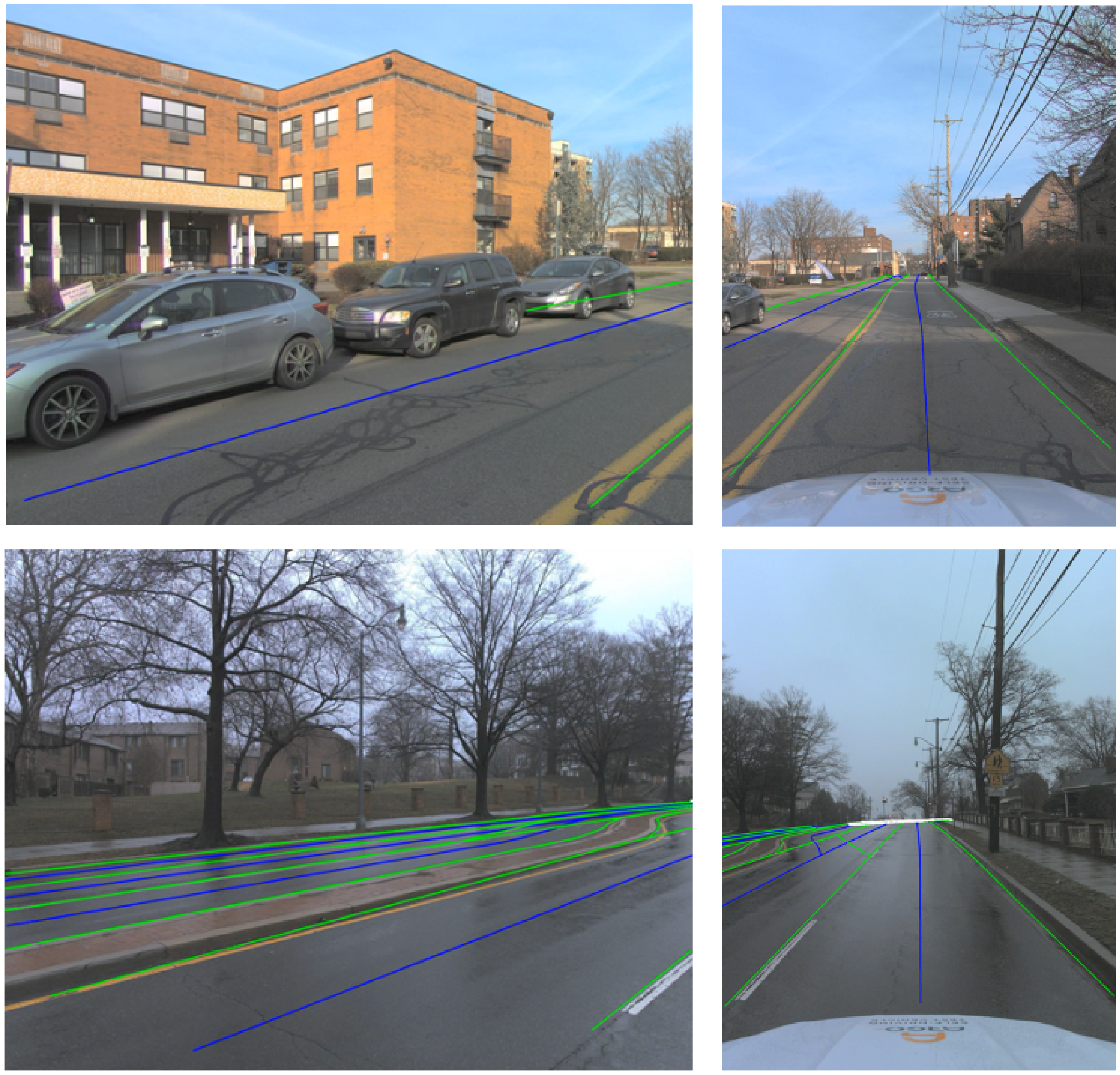


Figure 4.1: In contrast to traditional lane detection tasks (shown in green), our work explores an approach for centerline feature prediction, as denoted by blue lines. The approach leverages automatic label generation to process a large image dataset from an urban driving scenario with 2D and 3D ground truth information.

4.2 Centerline Networks

In this section, a baseline design for a 2D and 3D centerline prediction moduled termed CLiNet is described. A number of design considerations are made to decouple sensor-specific assumptions from the training process by incorporating the intrinsic parameters of each camera during the depth prediction task. Furthermore, in contrast to semantic segmentation frameworks that seek to classify each pixel with various semantic classes, we simplify the complexity of the model by only predicting the centerline key-points that are directly relevant for the driving task.

4.2.1 CLiNet Architecture

Our approach is motivated by the stacked hourglass network described in the Point Instance Network (PINet) work proposed in [KLA+20]. This approach stacks multiple hourglass networks during the training process that can be separated during testing as a tradeoff between inference time and precision. The model architecture and training process can be visualized in Fig. 4.2; where each network jointly learns centerline key-point features and contributes to the overall loss. First, an input image with known intrinsic parameters is normalized and resized to a fixed representation $\mathbf{I_0} \in \mathbb{R}^{H_0 \times W_0 \times 3}$. A sequence of convolutional layers with batch normalization is then applied to increase channel depth to a final representation $\mathbf{I_1} \in \mathbb{R}^{H_1 \times W_1 \times C_1}$ such that $H_1 = H_0/s$, where s is a scaling factor. The prediction heads for each hourglass block further process the intermediate representation $\mathbf{I_1}$ and output a confidence map ($\hat{\mathbf{F}} \in \mathbb{R}^{H_1 \times W_1 \times 1}$), depth estimates ($\hat{\mathbf{D}} \in \mathbb{R}^{N_e \times 1}$), and key-point offset values ($\hat{\mathbf{O}} \in \mathbb{R}^{H_1 \times W_1 \times 2}$) for each key-point on the centerline. For the experiments performed, $C_1 = 128$, $H_1 = 256$, $W_1 = 512$, and $s = 8$.

Losses

The confidence map focuses on the binary classification of key-points, i.e. predictions close to one indicate centerline key-point existence. Given a confidence prediction $\hat{F}_{i,j}$, the

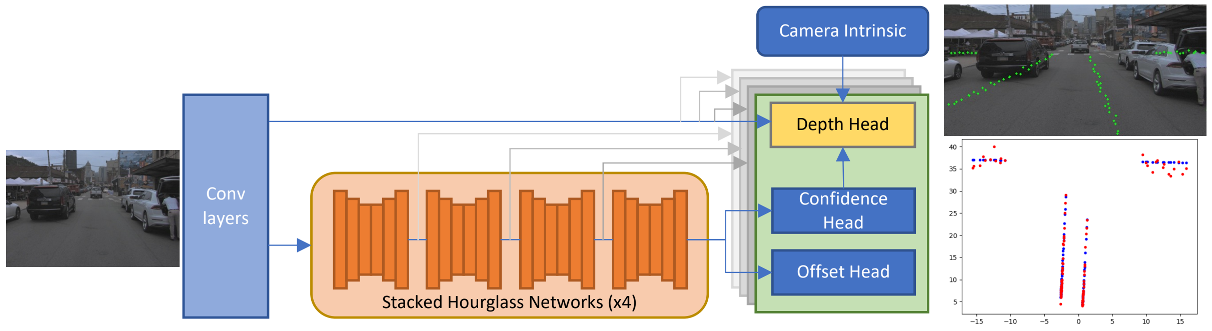


Figure 4.2: The method introduced utilizes stacked hourglass networks to localize centerline key-points in 2D while generating depth estimates to transform key-point estimates to 3D. Additional details on the hourglass networks can be found in [KLA+20].

confidence loss is characterized by Eq. 4.1, where the first term minimizes false negatives and the second minimizes false positives. Each term is normalized based on the number of key-points with a ground truth class that corresponds to a key-point N_e and the number of key-points that do not correspond to a key-point N_d. A ground truth confidence label is denoted by $F_{i,j}$.

$$L_{\text{conf}} = \frac{1}{N_e} \sum_{\{i,j|F_{i,j}=1\}} \left(1-\hat{F}_{i,j}\right)^2 + \frac{1}{N_d} \sum_{\{i,j|F_{i,j}=0\}} \left(\hat{F}_{i,j}\right)^2 \tag{4.1}$$

Together with the confidence map, a two-dimensional offset $\hat{\mathbf{O}}_{i,j} \in \mathbb{R}^2$ is predicted that indicates a scaling factor to map from a $H_1 \times W_1$ representation to the $H_0 \times W_0$ input scale. Therefore, the transformed prediction $\hat{F}_{i,j}$ in original scale, is given by $\left[j + s \cdot \hat{O}^x_{i,j}, i + s \cdot \hat{O}^y_{i,j}\right]^\top$ and $\hat{O}^x_{i,j}, \hat{O}^y_{i,j} \in [0,1]$. The offset loss is then characterized by

$$\begin{aligned} L_{\text{offset}} = & \frac{1}{N_e} \sum_{\{i,j|F_{i,j}=1\}} \left(O^x_{i,j} - \hat{O}^x_{i,j}\right)^2 + \\ & \frac{1}{N_e} \sum_{\{i,j|F_{i,j}=1\}} \left(O^y_{i,j} - \hat{O}^y_{i,j}\right)^2 \end{aligned} \tag{4.2}$$

Given that the features are encoded in 2D, an additional multilayer perceptron (MLP) is utilized to infer depth for each centerline key-point. A depth prediction $\hat{Z}_{i,j}$ is estimated by first converting a key-point prediction in pixel space $[i,j]^\top$ to camera coordinates. This

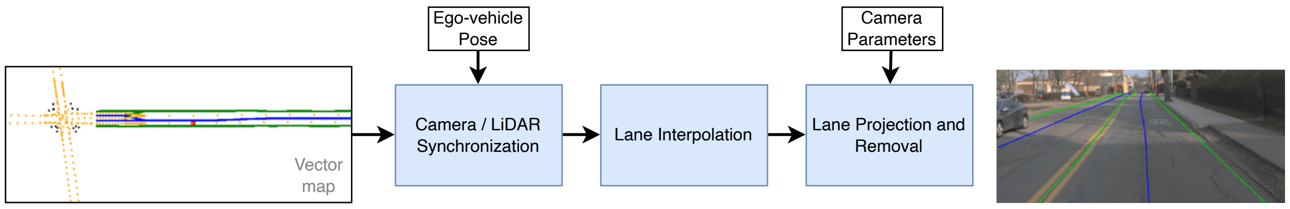


Figure 4.3: A pipeline is designed to automatically generate image key-point labels that correspond to centerlines via projective geometry. This is accomplished by aligning vector map information (left hand side) regarding each camera on board of the ego-vehicle and projecting 3D features onto an image perspective. Line segments that are far and that correspond to intersection road networks are removed.

transformation is given by the intrinsic camera parameters and by accounting for image resizing and crop operations. The camera-centric coordinates $\left[X_{i.j}, Y_{i,j}\right]^\top$ are then concatenated with its corresponding image embedding $\mathbf{I_1}_{i,j}$ to provide additional context. This final representation is passed through the MLP to predict its z-component, namely $\hat{Z}_{i,j}$. The depth loss is then characterized as shown below, where $Z_{i,j}$ denotes ground truth.

$$L_{\text{depth}} = \frac{1}{N_e} \sum_{\left\{i,j \mid F_{i,j}=1\right\}} \left(Z_{i,j} - \hat{Z}_{i,j}\right)^2$$

In summary, the losses are added and optimized jointly during the training process using the Adam optimizer with a constant learning rate. The overall loss is given by $L = L_{\text{conf}} + L_{\text{offset}} + \gamma L_{\text{depth}}$, where γ is a constant.

4.2.2 Experiments

To evaluate our reference solution, we introduce a large dataset for training, validation, and testing based on the Argoverse2 sensor data [WQA+21]. The evaluation additionally consists of a benchmark to quantify the performance of the approach for key-point localization in 2D and depth estimation (3D). Additional details on the data and benchmarks are provided below.

Centerlines Dataset

We augment the dataset to create a new reference termed AV Breadcrumbs that consists of $942,161$ individually labeled image frames across three front-facing cameras with known camera parameters. In contrast to standard labeling techniques that generally involve extensive manual labeling, we design a pipeline that automatically generates image key-point features with ground truth depth information. First, the ego-vehicle pose is synchronized to each camera by utilizing the LiDAR; given the pose of each camera over time and performing pose corrections, we align vector map information defined in a city coordinate frame and project its 3D features into each image frame. Given that projection preserves 2D and 3D correspondences, this facilitates the process of assigning unique lane IDs.

The dataset includes labels for centerlines and lane boundaries; where each label is interpolated in 3D and post-processed in an image frame to remove features that are too far or too tightly packed after projection. An important consideration involves intersections: although the full road network connectivity for each intersection is available, we find that sufficient context to identify entry and exit points of each intersection lane is often lacking when the vehicle is at an intersection. For this reason, we additionally remove labels that are too short or are part of intersections. The total split for the train, validation, and test sets corresponds to 659,720, 141,138, and 141,303 image frames, respectively. The auto-labeling pipeline is shown in Fig. 4.3 and a few data labels can be observed in the first row of Fig. 4.4. On average, we observe adequate alignment between the automatically generated features and the lanes visualized in each image.

Results

Each of the logs in the training, validation and test set comprises of sequential frames of data. We take a subset of these logs for our training and testing as two consecutive frames of data present similar features. As a result of which, we empirically chose every 10th image in the

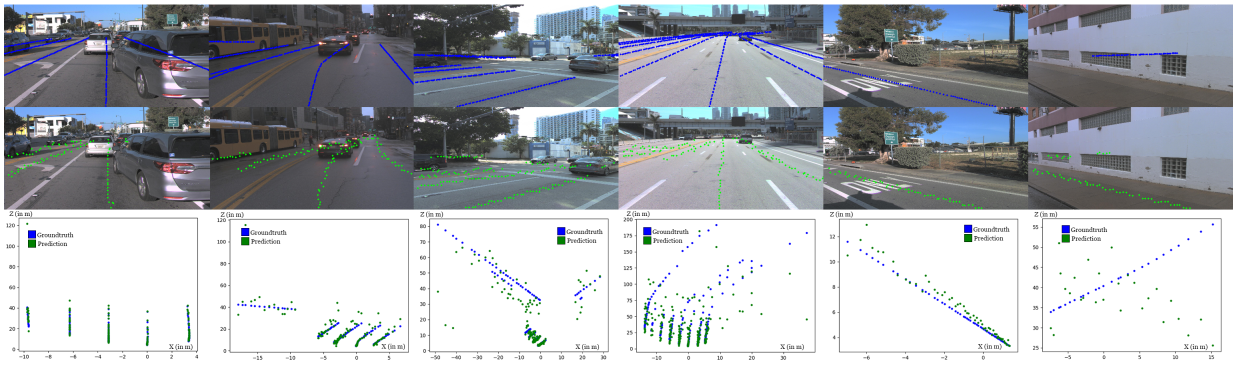


Figure 4.4: Data visualizations correspond to ground truth 2D labels (top row), 2D predictions from our approach (CLiNet - middle row), as well as depth estimates where the vertical axis corresponds to the z-axis and the horizontal axis corresponds to the x-axis regarding each camera frame (bottom row). Predictions and ground truth labels are denoted in green and blue, respectively.

sequence for training and testing. This reduced the number of test images to 13,046 images.

Fig: 4.4 shows the results from our model for the test set of our dataset. It can be seen that our model performs well in detecting the key-points on the straight line segments (row:2, col:4) as well as curved line segments (row:2, col: 2). Additionally, we observe that there are some challenging scenarios due to lack of context (col:1) and complexities due to complete occlusions (col:6).

The third row of the figure illustrates predicted depth versus ground truth depth for each of the input images in the top row. We observe that the points that are at a greater distance from the camera center have relatively large error compared to those that are closer to the camera center (row:3, col:4).

Table 4.1 shows the average precision, recall and F1 score of our model on the test set. We create an occupancy grid map of the predicted key-points where the centerline key-point indices are set to 1. To calculate the number of true positives, false positives and false negatives, we then create a window of size *n* x *n*. If the occupancy grid does not contain a key-point within the windowed region, we count this key-point to be a false negative. We zero out the windowed region otherwise, which we later use for counting false positives.

To calculate the average depth error, we use the L1 norm between the ground truth and the predicted **z** values. We normalize this error based on the distance of the key-point from the camera center. Error for key-points closer to the camera center is penalized more than the error for key-points farther away. Average normalized depth error for the test set is **2.0834**.

Table 4.1: Benchmark Results on the Test Set

Window Size	Precision	Recall	F1 Score
5	0.822	0.585	0.684
3	0.748	0.512	0.608
1	0.378	0.229	0.285

4.3 Occlusion Handling and Semantics

In Section 4.2, the centerline prediction method proposed leveraged ego-vehicle localization, sensor parameters, and lane-level map information to automatically label and process image data; this pipeline is shown in Figure 4.3. Nevertheless, one challenge that is observed in the training process involves occlusion. Since the map information is directly utilized to automatically label images based on projective geometry, some road structures and moving agents can severely occlude the centerline features of interest. An example can be visualized in Figure 4.5: while the centerline features from the nearby road elements have valid projections with respect to the camera's field of view, the keypoints are fully occluded by a bus that is too close to the camera and cannot be realistically recovered.

To model occlusions caused by dynamic and large static objects observed in the training data, semantic segmentation strategies can be utilized to reason about objects undergoing occlusion. For example, in Figure 4.6, a number of centerline labels are superimposed on their corresponding image frame shown in the first row and the outputs of a semantic segmentation model are shown in the second row. Each of the semantic mask outputs generated contains

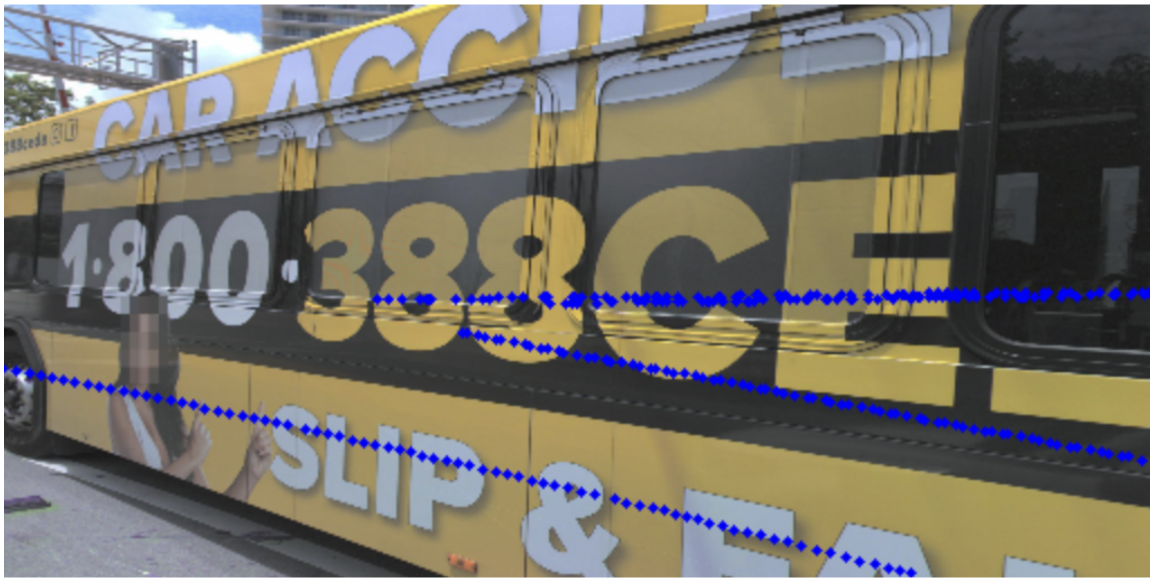


Figure 4.5: Data sample generated by the automatic labeling process without explicitly modeling occlusion. In this scenario, a bus is fully occluding centerline features.

information about drivable regions, vehicles, buildings, etc. and by comparing the semantic classes that each of the centerline keypoints overlap with, various strategies can be derived to remove the keypoints from the training set. For example, in the last row of Figure 4.6, only the centerline keypoints that directly overlap with the drivable region class are maintained.

In this section, the centerline prediction task specifically focuses on utilizing semantic segmentation to reason about occluded regions and introduces this information in the training process. In the first part of the section, occlusion is modeled in terms of percent occlusion thresholds to better reason about light vs severe occlusion. A number of experiments are then performed in the second part of the section, which include 2D and 3D keypoint prediction as a function of various percent occlusion increments. Finally, qualitative experiments are performed using data from a full scale autonomous driving platform.

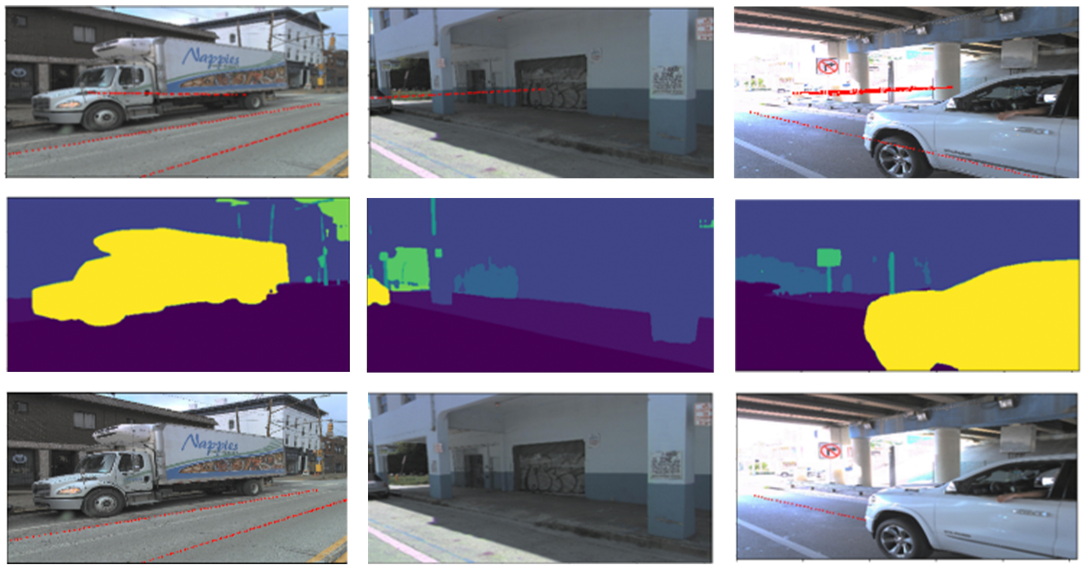


Figure 4.6: Visualization of various urban driving scenarios. On the top row, the centerline labels are projected using the first version of the automatic label generation pipeline. In the middle row, a semantic segmentation process is applied to the original image data to reason about different object classes. Using the object classes to formulate occlusion, centerlines are removed based on the semantic classes that infer occlusion as shown in the bottom row.

4.3.1 Semantic Segmentation for Occlusion Handling

Objects and structures of various sizes and shapes play a significant role when quantifying occlusion from a camera centric perspective. For instance, a vehicle driving on its lane that is far away from the camera will only partially occlude its centerline. On the other hand, if the vehicle is close to the camera, it will occlude more of the lane features. While it may be impossible to recover road information from full occlusion, the latter scenario with partial occlusion can be potentially resolved by inferring the missing keypoints based on the lane features that are observable. Evidently, there are other possibilities in between as the vehicle moves away from the camera: from full occlusion to light occlusion.

In addition to occlusion generated by dynamic objects such as cars and pedestrians, large and stationary objects such as buildings can impact the quality of the predictions. For instance, on the last column of Figure 4.4, we observe that centerline features behind buildings can influence

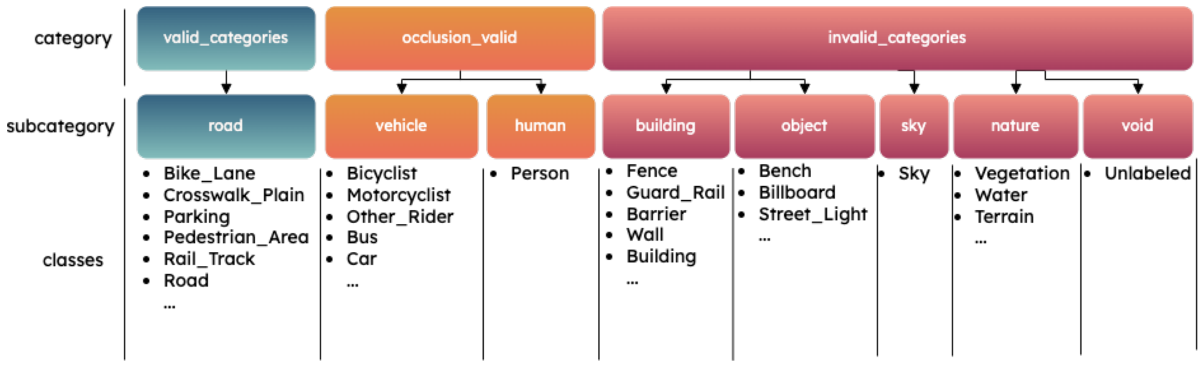


Figure 4.7: Semantic categories utilized to quantify occlusion specific logic. The various categories and subcategories are derived using the Mapillary dataset class definitions.

the training process and ultimately cause the network to hallucinate during prediction. In other words, objects that are not on the road are often more difficult to associate with centerline features and as a result provide less context in the training process. With these observations in mind, we characterize occlusion from objects based on three different categories. The first one corresponds to occlusion from objects with insufficient context (invalid). The second one captures occlusion from on-road objects such as pedestrians and vehicles (occlusion valid) that are directly related to the driving task and contain contextual information. Finally, the third one corresponds to road level semantic classes such as road and crosswalk (valid). Figure 4.7 illustrates the different classes of objects that are included in each category set. For simplicity we refer to these categories as invalid, occlusion_valid, and valid and utilize the well established Mapillary dataset semantic classes in the formulation of these categories.

The Mapillary dataset contains up to 124 semantic object classes. From these classes, we cluster individual classes into nine different subcategories based on their semantic meanings and assign each subcategory into the valid, occlusion_valid, or invalid parent categories. This provides a straightforward formulation for calculating percent occlusion as outlined in Algorithm 1. An important benefit of this formulation is that we can create class exceptions for unique classes as shown in line 13. For example, the Mapillary dataset includes a bike lane class and in the event that bike lane centerline features are included in the map information, we can leverage this class to eliminate keypoints that overlap with the bike lane class.

Algorithm 1 Filtering Keypoints Algorithm

```
1: Input: List keypoints, List keypoints_classes, List class_ids
2: Output: List filtered_kp, List filtered_classes,
3: Initialize empty lists: filtered_kp,
4: for i in range len(keypoints) do
5:     $lane \leftarrow keypoints[i]$
6:     $classes \leftarrow keypoints\_classes[i]$
7:     $init\_lane\_length \leftarrow len(lane)$
8:     $invalid\_count \leftarrow 0$
9:     $filtered\_lane \leftarrow []$
10:    for j in range len(classes) do
11:        $current\_label \leftarrow labels[int(classes[j])]$
12:        if current_label.category ∈ semantics_config["invalid_categories"] then
13:            $invalid\_count \leftarrow invalid\_count + 1$
14:            continue
15:        else if current_label.category ∈ semantics_config["occlusion_valid_categories"] then
16:            $invalid\_count \leftarrow invalid\_count + 1$
17:            filtered_lane.append(lane[j])
18:        else if current_label.category ∈ semantics_config["valid_categories"] then
19:            filtered_lane.append(lane[j])
20:        end if
21:    end for
22:    if invalid_count/init_lane_length < semantics_config["semantic_threshold"] then
23:        filtered_kp.append(np.array(filtered_lane))
24:        filtered_classes.append(class_ids[i])
25:    end if
26: end for
```

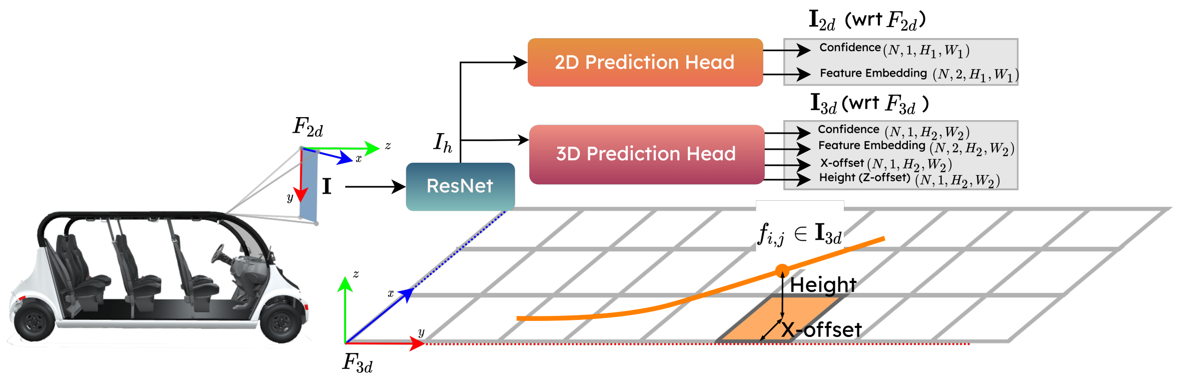
4.3.2 Experiments

To quantify the performance implications of the centerline prediction task subject to different occlusion threshold profiles, the dataset introduced in Section 4.2.2 is augmented with semantic segmentation attributes. To achieve this, a semantic segmentation process is incorporated into the pipeline proposed in Figure 4.3. The semantic segmentation process leverages the model introduced in [TSC20b]–which is trained on the Mapillary dataset. The model processes individual image frames and decorates the 2D and 3D centerline features based on the semantic segmentation masks generated and their overlap with the keypoints. A benefit of the approach lies in the performance and robustness of the hierarchical multi-scale attention method of the network. This attention approach achieves state of the art performance by combining the outputs of a single input processed at different scales (i.e. 0.5x, 1.0x, and 2.0x). The lower scale representations focus on regions that span a larger portion of the image, whereas the higher scales resolve smaller and fine-grain features with respect to the image frame. A few visualizations from this semantic segmentation model can be seen in the second row of Figure 4.6. In general, robust performance is achieved for the semantic classes of interest–such as vehicles, driveable regions, and buildings–which makes it a suitable option for our pipeline and task. Readers are referred to the original work for additional details on the implementation of the network architecture.

The Argoverse2 dataset described in Section 4.2.2 is augmented with the semantic attributes generated from the semantic segmentation pipeline. In total, $942,161$ individual frames are processed across the train, validation, and test sets. In the remainder of the section, the network implementation, the training process, and the results are presented.

Semantic Centerline Prediction

The model utilized for 2D and 3D centerline prediction differs from the approach implemented in Section 4.2.1 in a number of ways. First, the depth estimation process is performed with respect to a ground frame of reference perspective rather than predicting depth given a 2D coordinate in a camera frame. Overall, by switching to a ground plane perspective, the observed reconstructed depth estimates are more consistent and less noisy compared to the results presented in Figure 4.4. The frames utilized are shown in Figure 4.8–in combination with a high-level overview of the model architecture. This architecture is adapted from the 3D lane detection task introduced by Wang et al. [WQLC22]; however, a number of modifications are performed to extend the implementation for the centerline prediction task.



$\mathbf{I}_{2d}$ (wrt F_{2d})
2D Prediction Head
Confidence $(N,1,H_1,W_1)$
Feature Embedding $(N,2,H_1,W_1)$
$\mathbf{I}_{3d}$ (wrt F_{3d})
3D Prediction Head
Confidence $(N,1,H_2,W_2)$
Feature Embedding $(N,2,H_2,W_2)$
X-offset $(N,1,H_2,W_2)$
Height (Z-offset) $(N,1,H_2,W_2)$
F_{2d}
I_h
ResNet
$\mathbf{I}$
$f_{i,j} \in \mathbf{I}_{3d}$
Height
X-offset
F_{3d}

Figure 4.8: Frames of reference used in the 2D and 3D centerline prediction tasks adapted from [WQLC22].

The image frame (**I**) is processed using a ResNet backbone [HZRS16b]. The output generated from the backbone ($\mathbf{I_h}$) is then processed to generate the 2D ($\mathbf{I}_{2d}$) and the 3D ($\mathbf{I}_{3d}$) centerline predictions. The 2D output includes a confidence mask over the keypoints with dimensions $(N \times 1 \times H_1 \times W_1)$ and a 2D feature embedding represented by a $(N \times 2 \times H_1 \times W_1)$ tensor, where N corresponds to the batch size. The confidence output defines the existence of a centerline keypoint based on the score for each given cell. On the other hand, the feature embedding is useful for clustering centerlines and separating different instances. This is another key difference compared to the previous implementation for centerline prediction discussed in Section 4.2.1.

Similarly, the 3D prediction head includes information about confidence $(N \times 1 \times H_2 \times W_2)$, a 2D feature embedding $(N \times 2 \times H_2 \times W_2)$, an x-axis offset $(N \times 1 \times H_2 \times W_2)$, as well as an estimate for the height along the z-axis with dimensions $(N \times 1 \times H_2 \times W_2)$. The $H_2 \times W_2$ grid corresponds to a discretized representation of the ground plane defined below the camera center. The rigid body transformation between the camera and the ground plane is performed for each of the labels to represent the 3D centerlines in a ground frame. The secondary outputs generated by the 3D prediction head include an x-axis offset and a height estimate along the z-axis. Given that the 2D output grid loses information as a result of the discretization process due to the grid representation, the x-axis offset allows for lateral corrections for a particular cell and retrieves estimates at a centimeter-level. In addition, the height estimate provides an estimate for the elevation along the z-axis to predict variations in road elevations and slopes. To provide a more intuitive explanation, pinpointing the keypoint $F_{i,j}$ along the orange centerline shown in Figure 4.8 can be localized with more precision using the grid representation by using the x-offset and height predictions.

Another difference compared to the centerline prediction introduced in Section 4.2.1 lies in the objective function utilized to train the model. The loss function shown in Equation 4.3 incorporates a 2D loss and a 3D loss component. The 2D loss is calculated using the Binary Cross Entropy loss for confidence predictions and a 2D embedding loss. The embedding loss [WQLC22] is estimated using a distance metric between the 2D feature embeddings that aims to maximize the distance between keypoints that belong to different centerline instances while minimizing the distance between keypoints that belong to the same centerline instance. Similarly, the 3D loss incorporates the confidence and feature embedding components but additionally incorporates an offset loss for correcting the centerline predictions along the x-axis and a height component to penalize large errors in elevation. Additional details on each of the loss terms can be found in [WQLC22].

$$L_{\text{total}} = L_{\text{2d}}^{\text{bce}} + L_{\text{2d}}^{\text{embed}} + L_{\text{3d}}^{\text{bce}} + L_{\text{3d}}^{\text{embed}} + L_{\text{3d}}^{\text{offset}} + L_{\text{3d}}^{\text{height}} \tag{4.3}$$

Results and Discussion

Using the augmented dataset with the categories and subcategories defined, an evaluation of the centerline prediction model is performed as a function of various percent occlusion thresholds. The evaluations are performed using standard metrics from lane detection which include F1-score, precision, recall, as well as distance metrics to evaluate the errors along the x and z axes. Since the values along the y-axis are sampled at fixed increments during evaluation, this axis is omitted from the benchmark.

For each percent occlusion increment, a new training process is instantiated as a result of the occlusion handling logic outlined in Algorithm 1. This implies that the capabilities of each of the methods can differ as the occlusion logic becomes more or less strict. To provide further intuition, occlusion thresholds closer to zero are more strict and filter out the highest number of centerlines in the training set. On the other hand, occlusion thresholds closer to one are less strict and the condition is relaxed thus permitting most of the centerlines in the training set even under severe to complete occlusion scenarios. This further implies that a direct comparison is not trivial in the sense of using a single dataset to quantify the performance. Thus, the evaluation and benchmarks instead provide insights on the difficulty of predicting centerline features under light to severe occlusion rather than a direct comparison between each model trained.

For each threshold, a separate training process is initiated using two NVIDIA RTX 3090 GPUs trained for 70 epochs. While the training set is initially composed of 659,720 samples, the data was collected from a moving vehicle. This implies that in many scenarios, the variation between adjacent frames collected may be minimal. Thus, rather than using all of the training samples within a single epoch, a binning approach is performed by separating the training set in windows of size 20. For each group, a single frame is then randomly selected to provide variability throughout the training process. This equates to sampling a random data frame within a time window of 2s for the data collected since each camera generates data at 10Hz. The motivation for this windowing approach is to speed up the training process. For the experiments performed in this section, $H_1 = 144$, $W_1 = 256$, $H_2 = 200$, and $W_2 = 48$. For the BEV features, the discretization factor is selected based on a $0.5m/px$ discretization factor–implying that the longitudinal perception range is up to $100m$.

The results across 10 occlusion thresholds are presented in Figure 4.9. On the right plot, we visualize the error along the x and z axes at close and far ranges with respect to the groundtruth 3D centerline information. For consistency, the evaluation is performed using the BEV perception

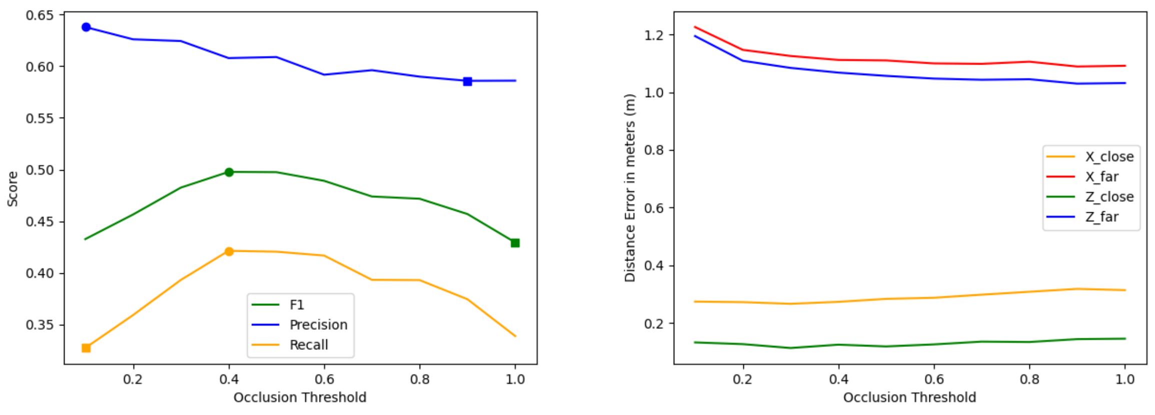


Figure 4.9: The performance of various models trained using different occlusion thresholds are shown. On the left, the F1 score, precision, and recall are plotted as functions of occlusion thresholds. On the right, the distance errors in terms of meters are shown as functions of the different occlusion thresholds; the errors are divided based on the x and z axes and close and far range. Close vs far range errors are quantified using a 40m distance threshold.

region: $x \in [-10m, 10m]$ and $y \in [3m, 103m]$. Aside from the 0.1 to 0.2 range, the distance errors remain consistent throughout the remaining occlusion thresholds and negligible differences are observed. This can imply that distance errors are not directly influenced as a function of occlusion thresholds. On the contrary, the F1 score, the precision, and recall are plotted on the left of the figure. We can observe that precision remains steadily but gradually decreases as the occlusion threshold increases. In contrast, recall exhibits a behavior that is less straightforward: it increases and peaks at 0.4 and begins to decrease again. The F1 score which takes into account precision and recall follows a similar pattern as recall. To better understand these patterns, a number of visualizations are extracted from the test set. A specific sample from the test set evaluated across all 10 occlusion thresholds is shown in Figure 4.10. In the figure, we observe that for low occlusion thresholds (i.e. between 0.1 and 0.2), the model does not generate valid predictions and it does not produce an initial prediction until a threshold of 0.3 is reached. Even in light occlusions scenarios, the model appears to be conservative in the predictions, which explains why the false positive rate is lower and as result the precision is slightly higher compared to the other models with higher thresholds. Hence, since the models with lower thresholds are more

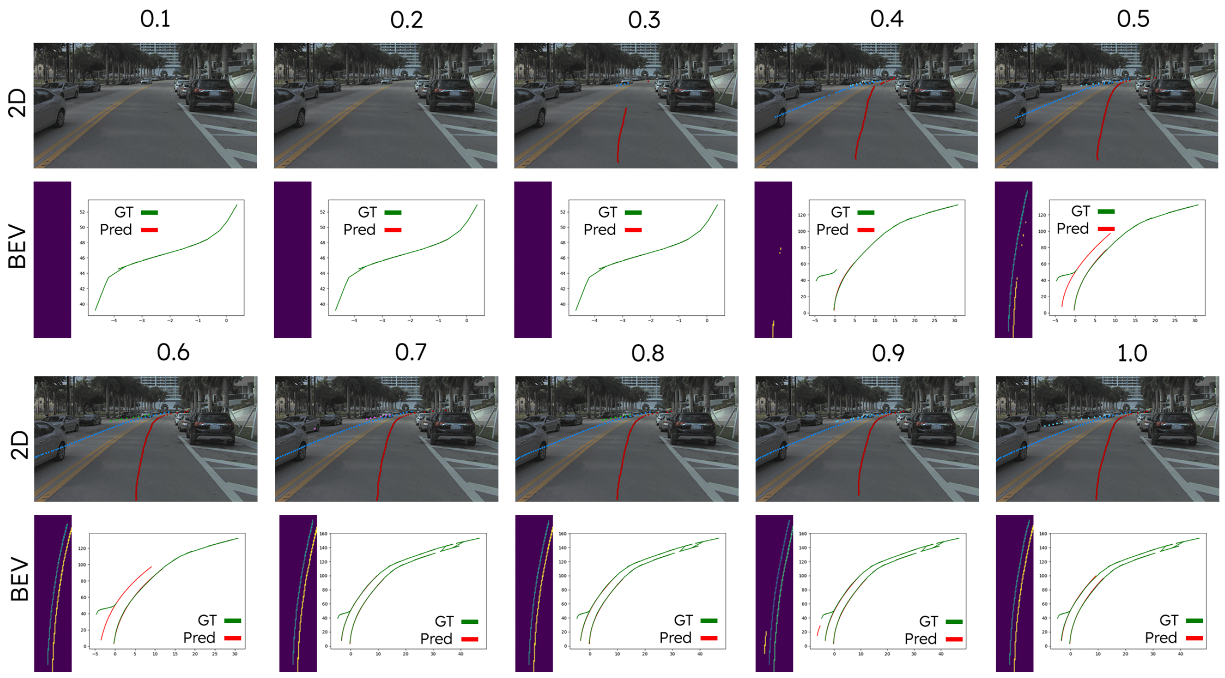


Figure 4.10: A sample that corresponds to a long curved road from the Argoverse2 dataset is presented using different occlusion thresholds in increments of 0.1. The 2D predictions are superimposed on the original image data where the different colors denote individual centerline instances extracting using a clustering algorithm that leverages the feature embedding predicted by the network. Similarly, two BEV visualizations are provided for each sample. The first visualization showcases the ability of the model to cluster individual centerlines in BEV and the second visualization allows for comparisons between the groundtruth (green) and the predictions (red); the units are given in meters.

conservative, they tend to increase the number of false negatives which in turn lowers recall.

In contrast, as the occlusion threshold increases, the models begin to predict centerlines under more severe occlusion and the quality of the predictions significantly improve in 2D and 3D. While referring back to the F1 score, we observe that within the 0.4 and 0.8 range, the best performance is reached before it decreases again. This performance gap is captured in Figure 4.11, where we observe that the models with thresholds 0.9 and 1.0 must predict heavily occluded centerlines within an adjacent intersection road with little direct visibility. Evidently, the models with higher occlusion thresholds infer a higher number of false positives in these more difficult scenarios causing the recall to decrease. Nevertheless, a qualitative analysis indicates that the performance in 3D remains adequate in 2D and 3D up to a 0.9 threshold. We note that when

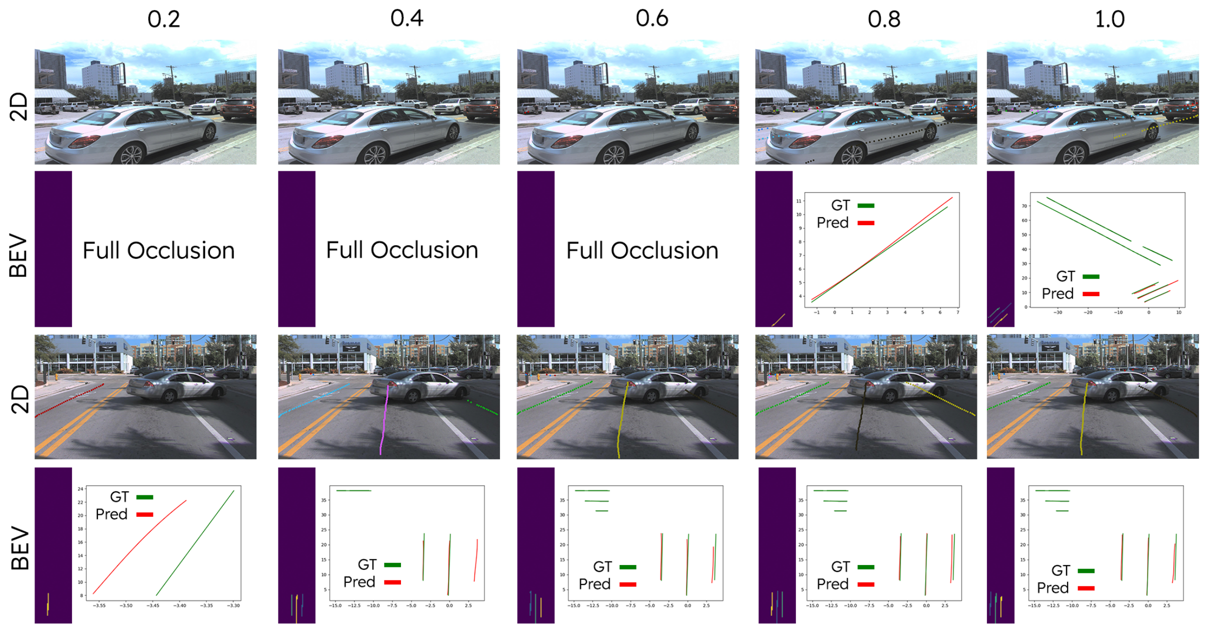


Figure 4.11: A sample that corresponds to a heavily occluded scene from the Argoverse2 dataset is presented. The same visualization tools presented in Figure 4.10 are utilized. The units of the 3D projection plots are given in meters

the threshold reaches 1.0, some of the predictions become more noisier in particular to the 2D prediction task.

To perform a qualitative analysis and evaluate how well the models generalize, a data collection process is performed using a camera setup outside of the training distribution. The data specifically is collected at the UC San Diego university campus and captures various road topologies including turns and steep hills. Figures 4.12 showcases some of the capabilities of the models across different occlusion thresholds. Evidently, the models trained with lower occlusion thresholds remain consistent with the evaluation on the Argoverse2 dataset. For higher thresholds up to 0.9, the models generalize well in the image frame but present visible errors in the 3D perspective given the domain gap and different sensor configurations from the training set. An important note to make is that for the threshold of 1.0, the performance in the image domain does not necessarily increase. For instance, we note that on 2D predictions corresponding to 1.0, the centerline corresponding to the ego vehicle does not fully extend compared to 0.9. In general,

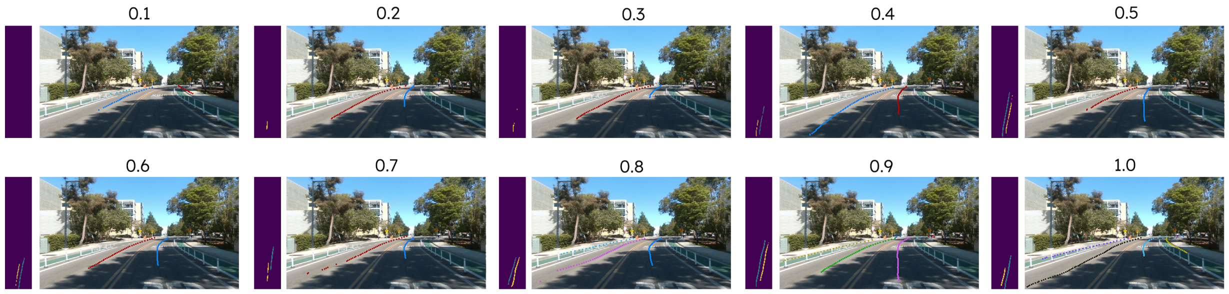


Figure 4.12: Curved road predictions for a university environment. The BEV prediction from the 3D prediction head is shown to the left of each output and the 2D predictions are superimposed to the right. In average, an inference time of 10ms is observed.

we note that the model trained with a threshold of 0.9 performs and generalizes better for the university dataset. As an added benefit of the approach utilized, we note that the architecture can accommodate camera streams with rates up to 95.7fps. In average, we observe that the inference time is within a 10ms and can thus meet real-time requirements across multiple camera streams.

Additional visualizations from the Argoverse2 and UC San Diego test sets can be found in Appendix B.1 and B.2.

4.3.3 Summary

In this chapter, we introduced a large-scale dataset with multiple reference solutions for 2D and 3D centerline detection. By explicitly modeling occlusion through the use of keypoint-level semantics, the methods show real-time and robust performance in scenarios with light and severe occlusion. This novel task shows promising results on the Argoverse2 2D and 3D dataset for real-time scene understanding and urban navigation applications. On different sensor platforms (i.e. the UC San Diego data set), the models generalize well in the 2D detection task; however, additional work is necessary to reduce the gap in the 3D modeling task. This gap be potentially addressed from multiple directions including a fusion based strategy similar to the camera-LiDAR method proposed in Chapter 2 or by improving performance on depth estimation of the networks. Future work includes the integration of similar real-time centerline prediction methods with global planning strategies. This integration can be explored using conditional generative models (i.e. Chapter 3) or using traditional map matching strategies with coarse standard definition maps.

Chapter 4 includes material being prepared for peer review. The dissertation author is the primary co-author and contributed to the design, implementation, dataset generation, and evaluations. Narayanan Elavathur Ranganatha contributed to the implementation and dataset generation process. Srinidhi Srinivas contributed to the implementation, and dataset generation. Yunchao Yao assisted in the development of the datasets and data curation.

Chapter 5

Conclusion

In conclusion, this dissertation showcases a comprehensive exploration of cutting-edge techniques for advancing autonomous driving capabilities. The three chapters collectively contribute to addressing critical issues within this domain. By introducing a framework for semantic map generation in Chapter 2, the work enables accurate localization and semantic mapping through a sensor fusion process. The dynamic trajectory generation approach in Chapter 3 leverages conditional generative models to adapt to changing global plans while reducing reliance on extensive map priors. Chapter 4 further extends the research by introducing real-time centerline prediction models, emphasizing the practicality of real-time scene understanding. These advancements and findings can pave the way for more efficient, cost-effective, and adaptable autonomous driving solutions and applications.

In regards to future research, the integration of the methodologies presented in this study holds paramount importance as they transition towards autonomous driving software stacks. A pivotal step forward involves conducting extensive long term evaluations across different types of deployments, ranging from small scale to expansive scenarios. A fundamental question that demands additional exploration pertains to the optimal balance between fully dynamic approaches and reliance on static maps (i.e. HD maps). This investigation is essential in quantifying the potential of achieving fully dynamic strategies without dependence on high-definition maps, or whether a judicious fusion of both paradigms is requisite. In the event that a hybrid solution proves optimal, a promising avenue for subsequent research emerges: the development of a dynamic management system capable of real-time scene understanding and decision-making, a direction that could involve the map change detection and online HD mapping research domains. Ultimately, this trajectory shows potential to reshape the architecture and design of autonomous driving systems, imbuing them with enhanced adaptability and real-time capabilities. It is evident that the diverse operational design domains demand varied requisites for autonomous systems and architectures. As we strive towards a universally applicable and scalable system, tailored for the intricacies of dynamic, lengthy path planning missions, the aforementioned contemplations emerge.

Appendix A

Semantic Mapping

A.1 2D Semantic Maps - UC San Diego Data

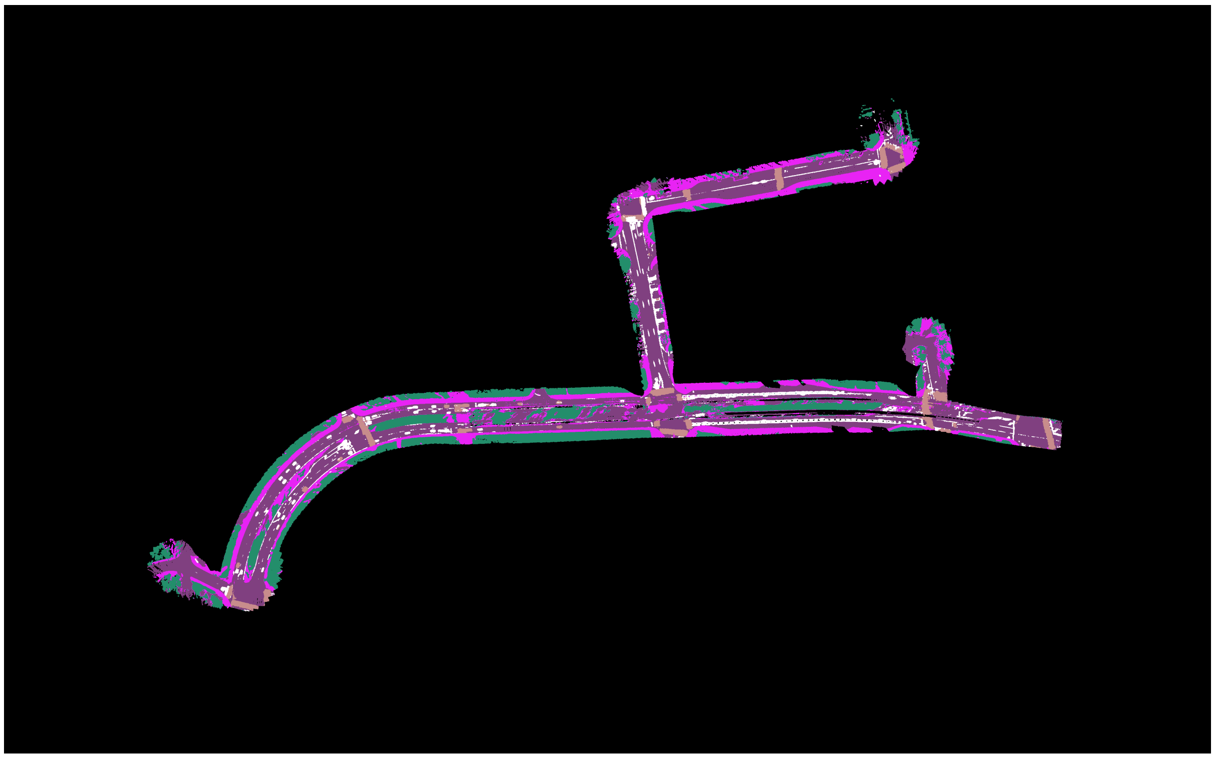


Figure A.1: 2D semantic map for Gilman Dr. shown at 11% of the original scale. Various artifacts are observed as a result of construction elements in the fusion process.

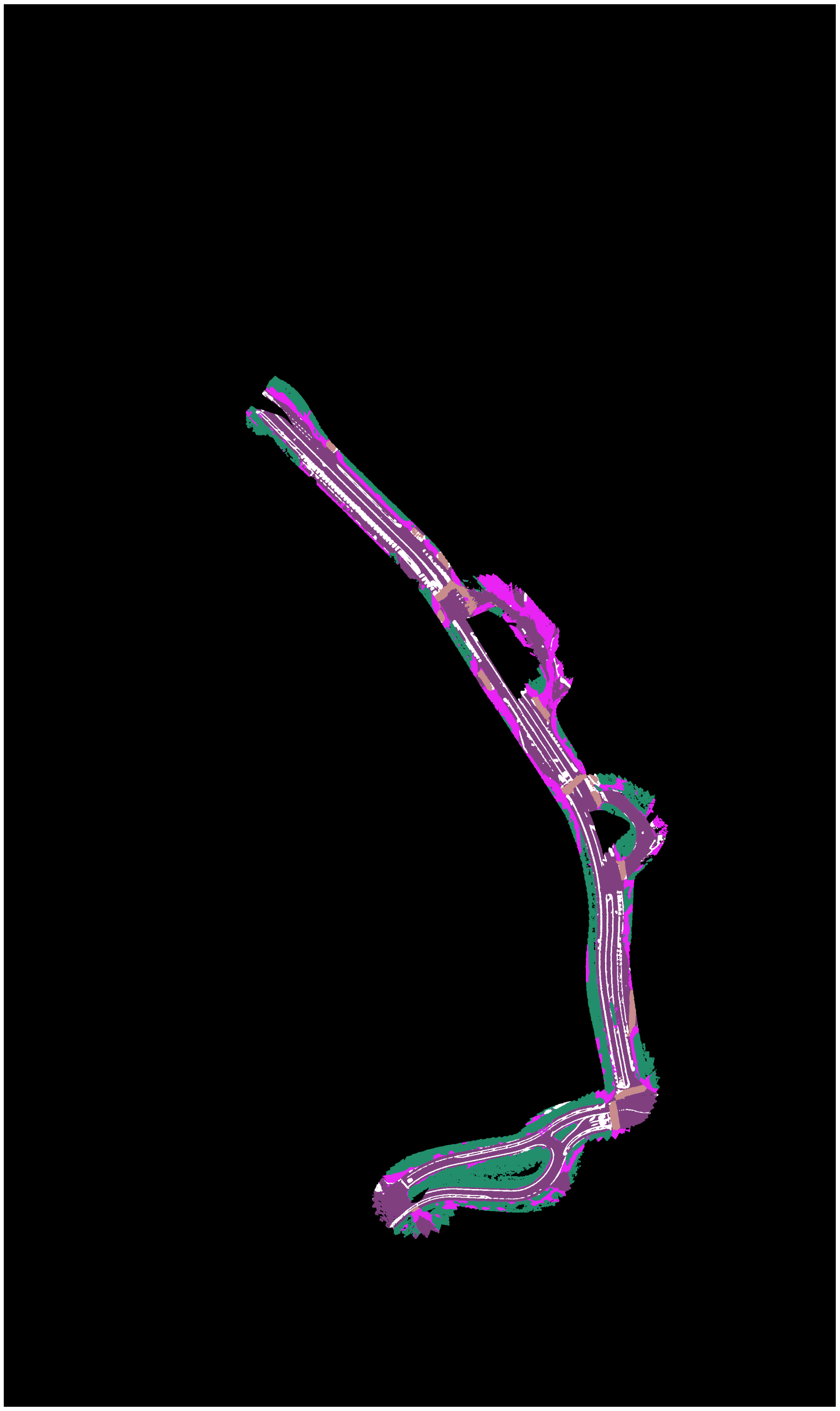


Figure A.2: 2D semantic map for East Campus shown at 14% of the original scale. The map showcases diverse road geometries.

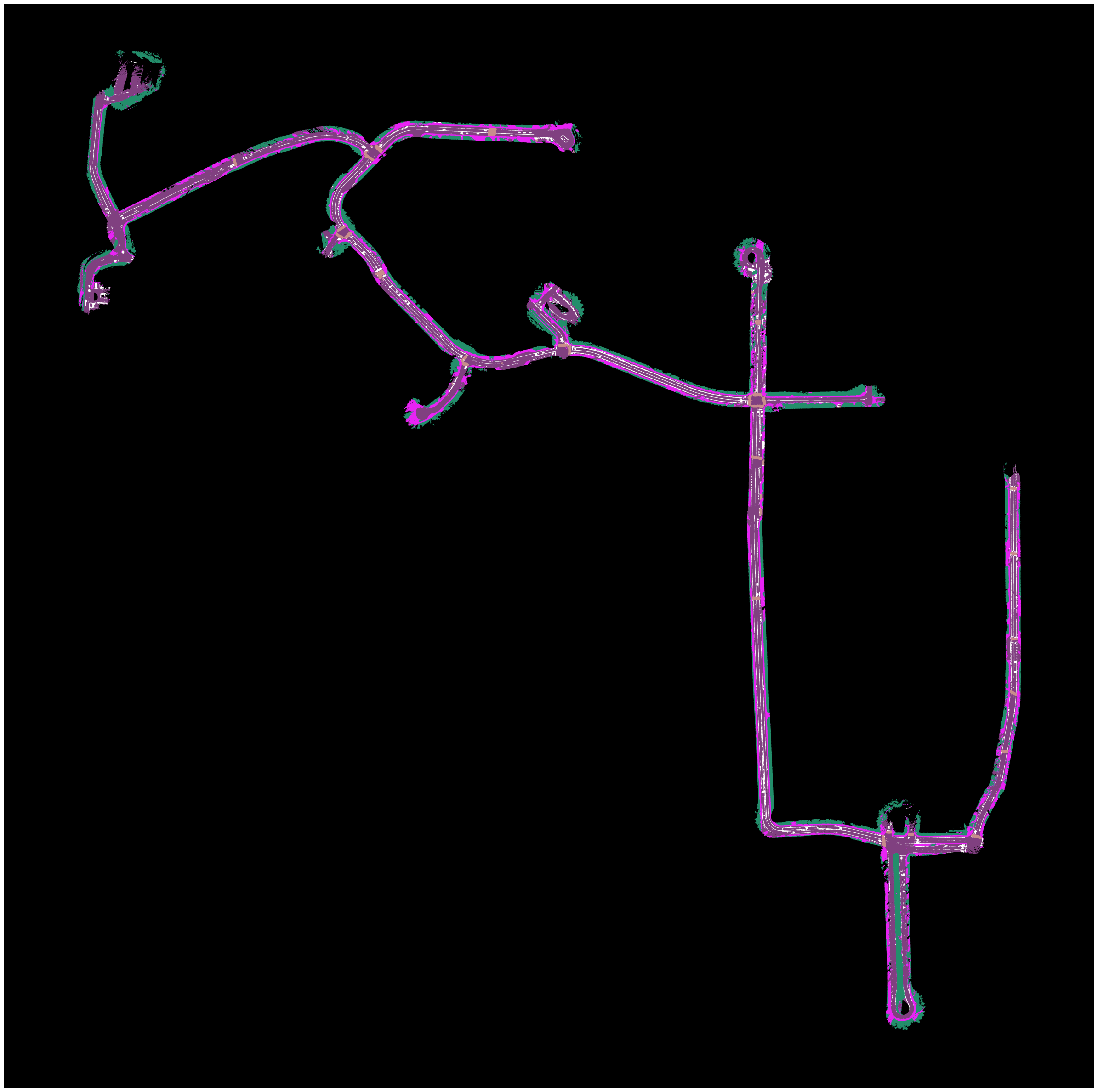


Figure A.3: 2D semantic map for North and East Campus shown at 6% of the original scale. North points in the downward direction.

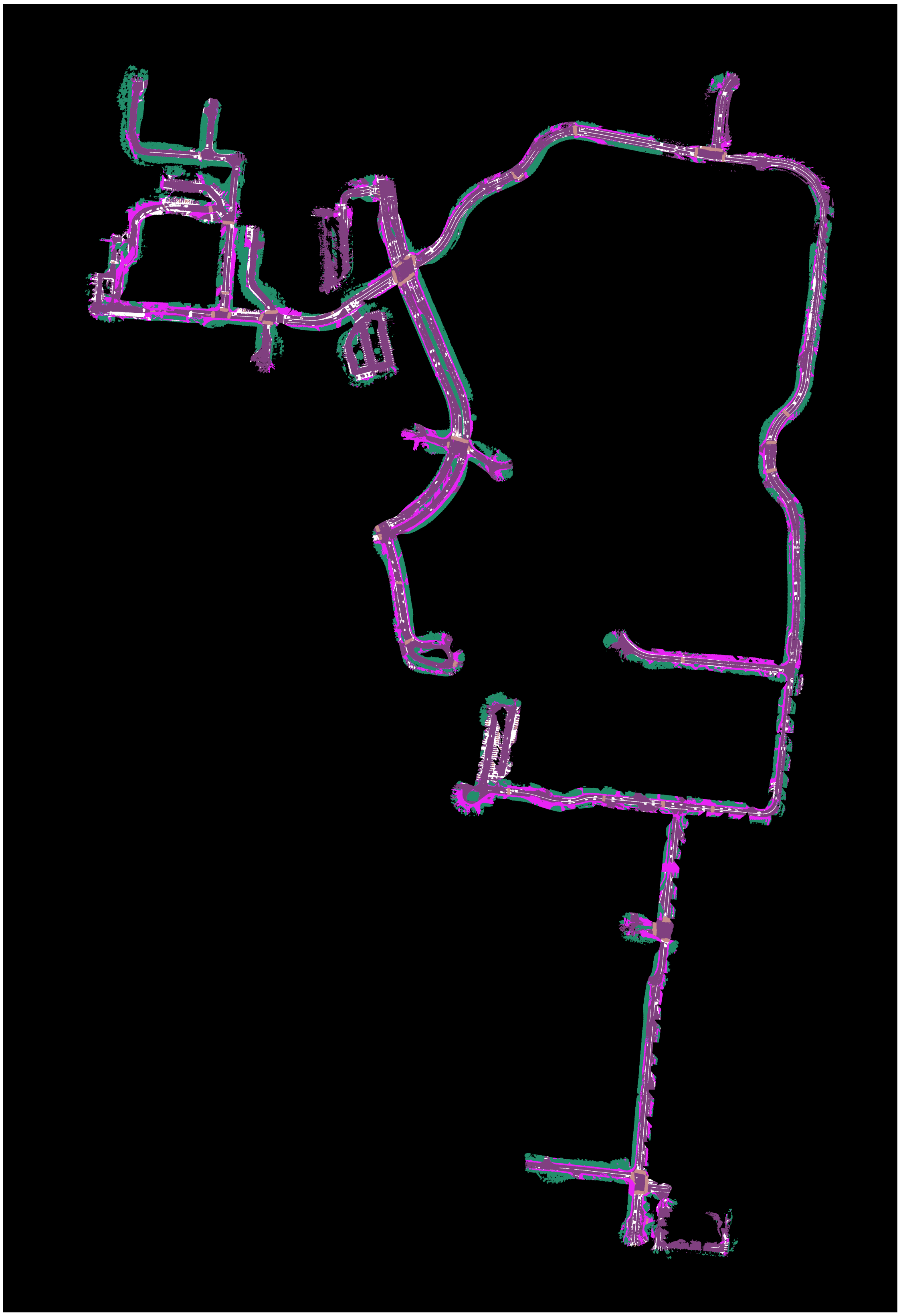


Figure A.4: The largest 2D semantic map generated for the UC San Diego campus that covers the South, West, and North parts of campus. The map is shown at 8% of the original scale.

Appendix B

Centerline Networks

B.1 2D and 3D Centerline Predictions - Argoverse2

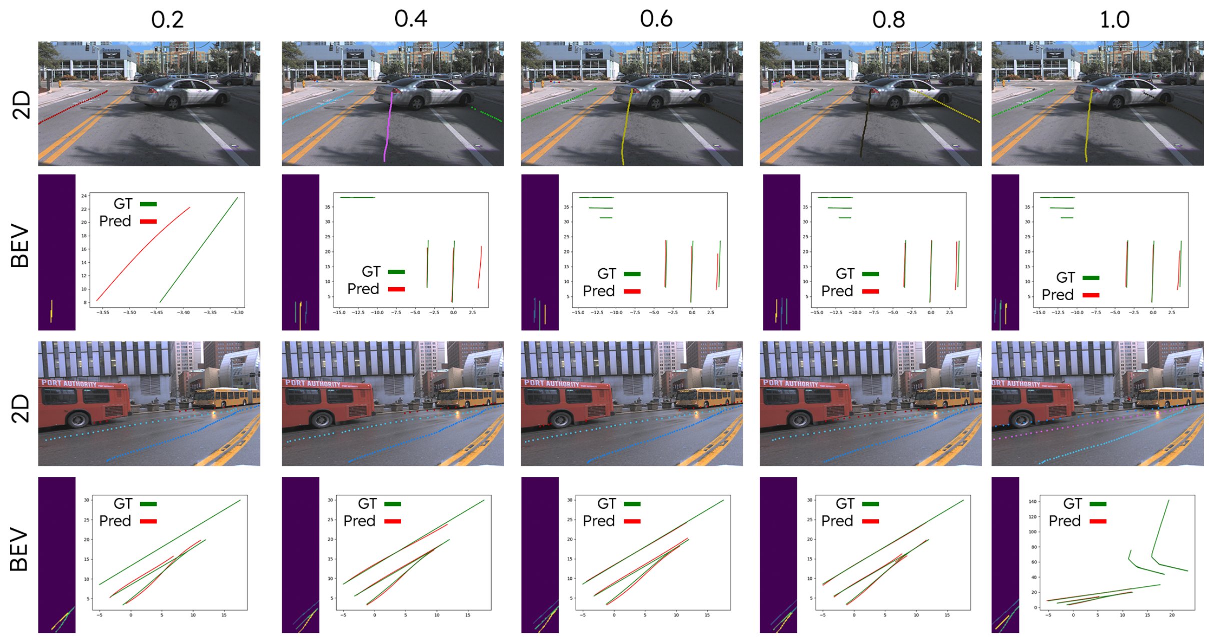


Figure B.1: Argoverse2 test samples using different occlusion thresholds in increments of 0.2. The 2D predictions are superimposed on the original image data where the different colors denote individual centerline instances extracting using a clustering algorithm that leverages the feature embedding predicted by the network. Two BEV visualizations are provided for each sample. The first visualization showcases the ability of the model to cluster individual centerlines in BEV and the second visualization allows for comparisons between the groundtruth (green) and the predictions (red); the units are given in meters.

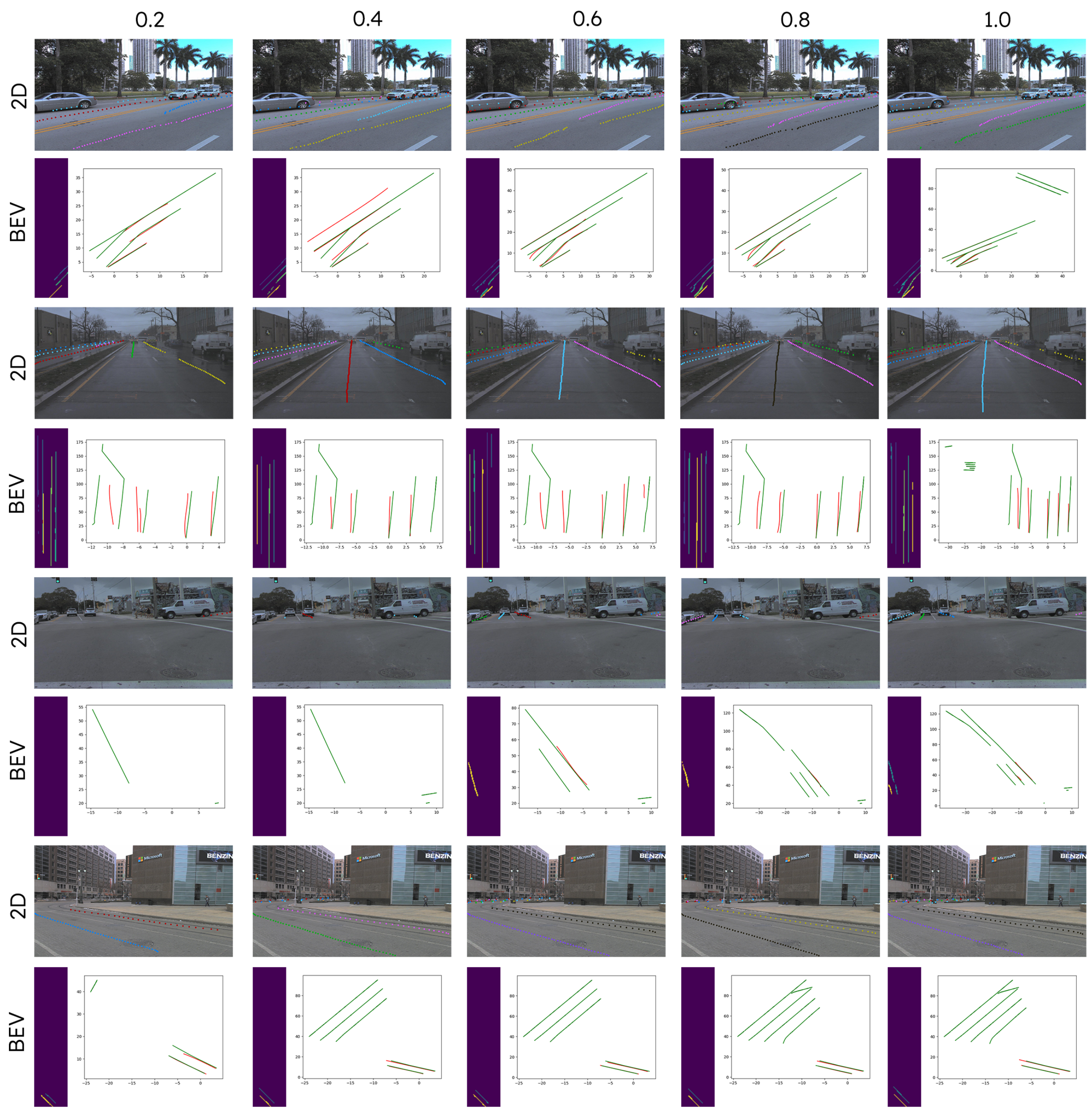


Figure B.2: Argoverse2 test samples using different occlusion thresholds in increments of 0.2. The 2D predictions are superimposed on the original image data where the different colors denote individual centerline instances extracting using a clustering algorithm that leverages the feature embedding predicted by the network. Two BEV visualizations are provided for each sample. The first visualization showcases the ability of the model to cluster individual centerlines in BEV and the second visualization allows for comparisons between the groundtruth (green) and the predictions (red); the units are given in meters.

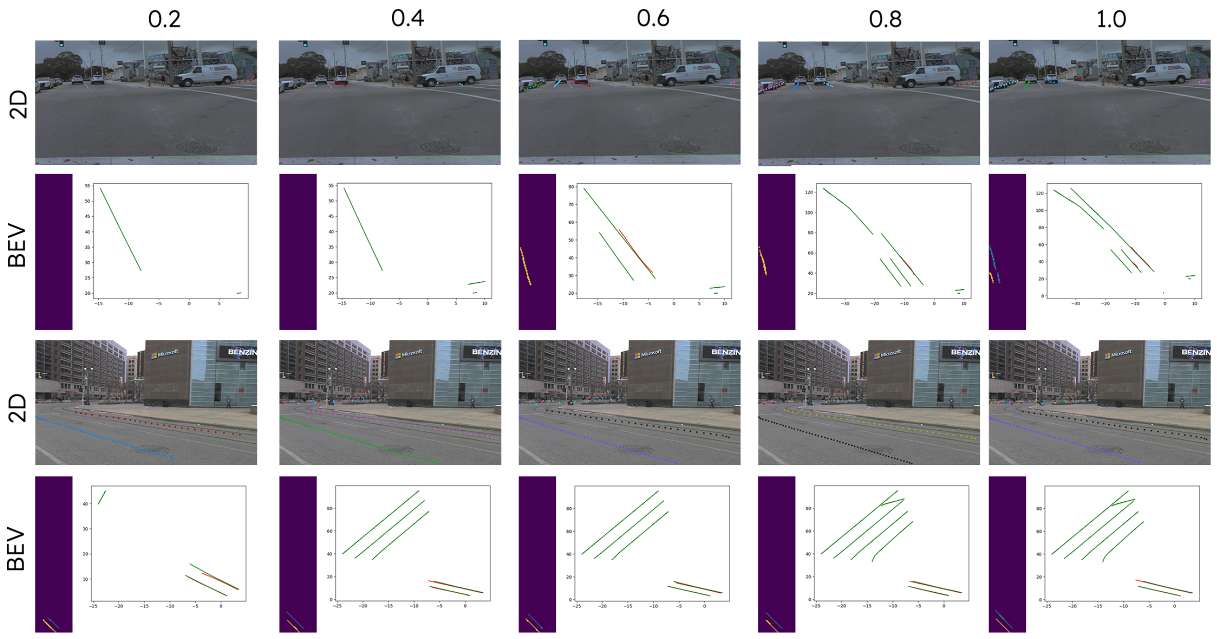


Figure B.3: Argoverse2 test samples using different occlusion thresholds in increments of 0.2. The 2D predictions are superimposed on the original image data where the different colors denote individual centerline instances extracting using a clustering algorithm that leverages the feature embedding predicted by the network. Two BEV visualizations are provided for each sample. The first visualization showcases the ability of the model to cluster individual centerlines in BEV and the second visualization allows for comparisons between the groundtruth (green) and the predictions (red); the units are given in meters.

B.2 2D and 3D Centerline Predictions - UC San Diego Data

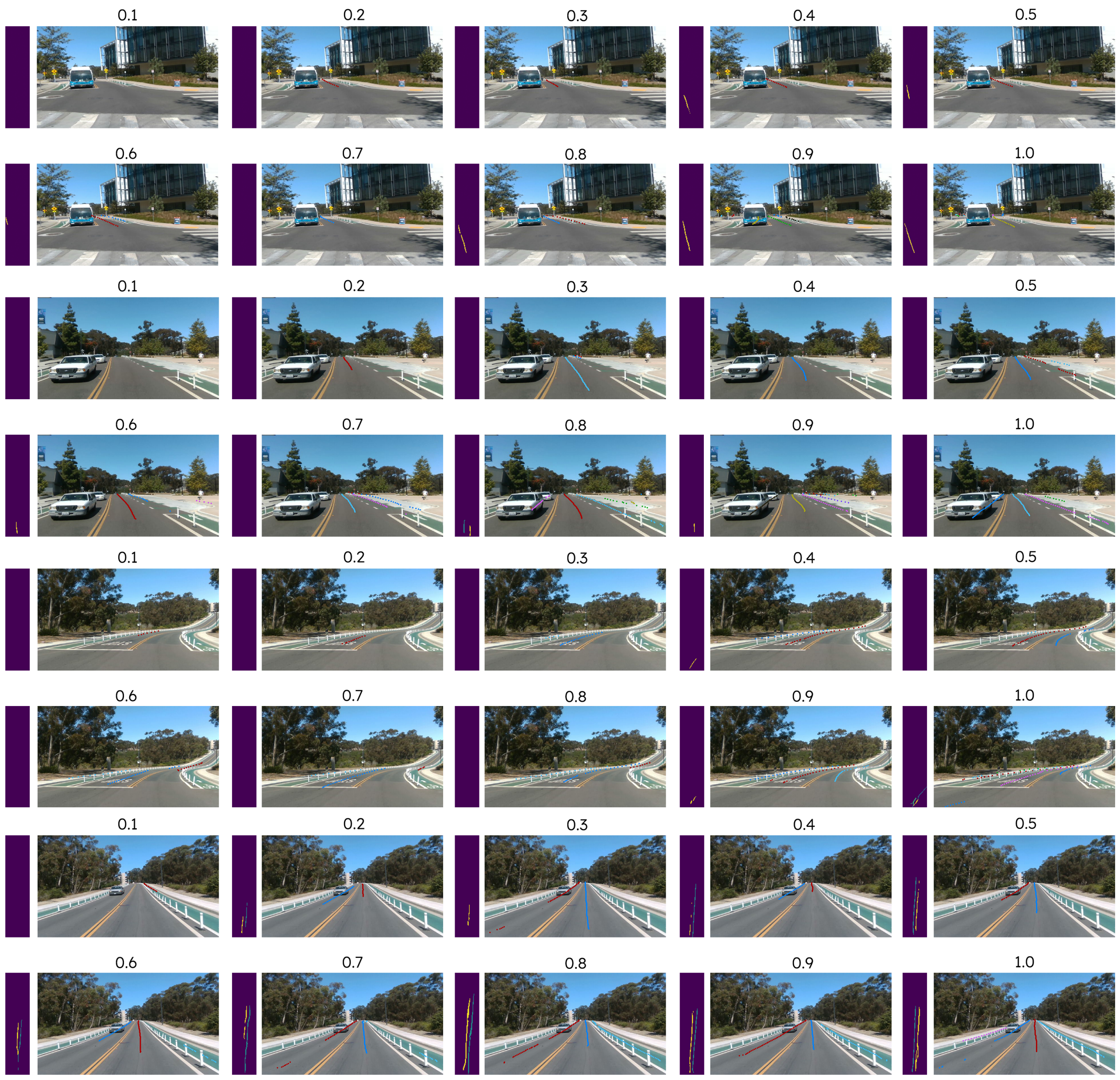


Figure B.4: Data samples from UC San Diego data using different occlusion handling models (using Argoverse2 data). The samples show various road topologies including intersectinos and curved road segments undergoing light to full occlusion.

Bibliography

[AGR$^+$16] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[apo17] Apollo: An open autonomous driving platform, 2017. Available online: `https://github.com/ApolloAuto/apollo` (accessed on 29 June 2023).

[ARKR19] Alexander Amini, Guy Rosman, Sertac Karaman, and Daniela Rus. Variational end-to-end navigation and localization. In *2019 International Conference on Robotics and Automation (ICRA)*, page 8958–8964. IEEE, 2019.

[BCF$^+$22] Yifeng Bai, Zhirong Chen, Zhangjie Fu, Lang Peng, Pengpeng Liang, and Erkang Cheng. Curveformer: 3d lane detection by curve propagation with curve queries and attention, 2022.

[BFC08] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x):xx–xx, 2008.

[BKO18] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst, 2018.

[BZT$^+$23] Martin Büchner, Jannik Zürn, Ion-George Todoran, Abhinav Valada, and Wolfram Burgard. Learning and aggregating lane graphs for urban automated driving. *arXiv preprint arXiv:2302.06175*, 2023.

[Cho17] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[CKT$^+$21] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[CLU$^+$22] Yigit Baran Can, Alexander Liniger, Ozan Unal, Danda Paudel, and Luc Van Gool. Understanding bird's-eye view of road semantics using an onboard camera. *IEEE Robot. Autom. Lett.*, 7(2):3302–3309, 2022.

[COR$^+$16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[CPZ$^+$21] Henrik Christensen, David Paz, Hengyuan Zhang, Dominique Meyer, Hao Xiang, Yunhai Han, Yuhan Liu, Andrew Liang, Zheng Zhong, and Shiqi Tang. Autonomous vehicles for micro-mobility. *Auton. Intell. Syst.*, 1(11):1–35, Nov 2021.

[CSL$^+$22] Li Chen, Chonghao Sima, Yang Li, Zehan Zheng, Jiajie Xu, Xiangwei Geng, Hongyang Li, Conghui He, Jianping Shi, Yu Qiao, and Junchi Yan. Persformer: 3d lane detection via perspective transformer and the openlane benchmark. In *European Conference on Computer Vision (ECCV)*, 2022.

[CSU21] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14403–14412, 20-25 June 2021.

[CvMG$^+$14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[CZP$^+$18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 833–851, Cham, 2018. Springer International Publishing.

[DFRDW11] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte. Classification and semantic mapping of urban environments. *The International Journal of Robotics Research*, 30(1):5–32, 2011.

[Dic07] E. Dickmanns. *Dynamic Vision for Perception and Control of Motion*. Springer Verlag, Heidelberg, October 2007.

[DMCD21] Isht Dwivedi, Srikanth Malla, Yi-Ting Chen, and Behzad Dariush. Bird's eye view segmentation using lifted 2d semantic features. In *British Machine Vision Conference (BMVC)*, pages 6985–6994, 22-25 Nov 2021.

[DRC$^+$17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *1st Annual Conference on Robot Learning*, volume 78, pages 1–16. PMLR, 13–15 Nov 2017.

[FDSF19] T. Fernando, S. Denman, S. Sridharan, and C. Fookes. GD-GAN: Generative Adversarial Networks for Trajectory Prediction and Group Detection in Crowds. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, volume 11361, pages 314–330, Cham, 2019. Springer.

[FGT$^+$22] Zhengyang Feng, Shaohua Guo, Xin Tan, Ke Xu, Min Wang, and Lizhuang Ma. Rethinking efficient lane detection via curve modeling. In *Computer Vision and Pattern Recognition*, 2022.

[GCZ$^+$20] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jinghao Miao, Jingao Wang, and Tae Eun Choe. Gen-lanenet: A generalized and scalable approach for 3d lane detection. In *Computer Vision - ECCV 2020 - 16th European Conference*, 2020.

[GJF$^+$18] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[HDVG18] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11211, pages 435–453, Cham, 2018. Springer.

[HDVG19] Simon Hecker, Dengxin Dai, and Luc Van Gool. Learning accurate, comfortable and human-like driving. *arXiv preprint arXiv:1903.10995*, 2019.

[HML$^+$19] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2911–2920, 2019.

[HMLU18] Namdar Homayounfar, Wei-Chiu Ma, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Hierarchical recurrent attention networks for structured online maps. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3417–3426. IEEE, 2018.

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968.

[HPSS20] Robin Heinzler, Florian Piewak, Philipp Schindler, and Wilhelm Stork. Cnn-based lidar point cloud de-noising in adverse weather. *IEEE Robotics and Automation Letters*, 5(2):2514–2521, 2020.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[HSG$^+$20] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemysław Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, and Alex Kndall. Urban driving with conditional imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257, Paris, France, 31 May 2020.

[HYC$^+$23] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, Vancouver, BC, Canada, June 2023.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IER$^+$20] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. Mats: An interpretable trajectory forecasting representation for planning and control, 2020.

[ILSP20] Boris Ivanovic, Karen Leung, Edward Schmerling, and Marco Pavone. Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach. *IEEE Robotics and Automation Letters*, 6(2):295–302, 2020.

[Jia18] J. Jiao. Machine learning assisted high-definition map creation. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 367–373, July 2018.

[KG15] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86 – 103, 2015.

[KLA$^+$20] Yeongmin Ko, Younkwan Lee, Shoaib Azam, Farzeen Munir, Moongu Jeon, and Witold Pedrycz. Key points estimation and point instance segmentation approach for lane detection, 2020.

[KTI+15] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[LCG+23] Tianyu Li, Li Chen, Xiangwei Geng, Huijie Wang, Yang Li, Zhenbo Liu, Shengyin Jiang, Yuting Wang, Hang Xu, Chunjing Xu, Feng Wen, Ping Luo, Junchi Yan, Wei Zhang, Xiaogang Wang, Yu Qiao, and Hongyang Li. Topology reasoning for driving scenes. *arXiv preprint arXiv:2304.05277*, 2023.

[LCW+23] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. In *International Conference on Learning Representations*, 2023.

[LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[LT10] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE International Conference on Robotics and Automation*, pages 4372–4378, May 2010.

[LYW+23] Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. In *International conference on machine learning*. PMLR, 2023.

[MCUS18] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-time semantic mapping for autonomous off-road navigation. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 335–350, Cham, 2018. Springer International Publishing.

[MLU17] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3438–3446, 2017.

[NORBK17] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[PLH+19] David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit Binnani, and Henrik Christensen. Lessons learned from deploying autonomous vehicles at UC San Diego. In *Field and Service Robotics*, Tokyo, JP, August 2019.

[PXLC22] David Paz, Hao Xiang, Andrew Liang, and Henrik Iskov Christensen. Tridentnetv2: Lightweight graphical global plan representations for dynamic trajectory generation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9265–9271, Philadelphia, PA, USA, 23-27 May 2022.

[PZC22] David Paz, Hengyuan Zhang, and Henrik I. Christensen. Tridentnet: A conditional generative model for dynamic trajectory generation. In Marcelo H. Ang Jr, Hajime Asama, Wei Lin, and Shaohui Foong, editors, *Intelligent Autonomous Systems 16*, pages 403–416, Cham, June 2022. Springer.

[PZL$^+$] David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik I Christensen. Probabilistic semantic mapping for urban autonomous driving applications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2059–2064. IEEE.

[RDS$^+$15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[RST$^+$20] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pages 1–6, Rhodes, Greece, 20-23 September 2020.

[SGST13] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. S. Torr. Urban 3d semantic modelling using stereo vision. In *2013 IEEE International Conference on Robotics and Automation*, pages 580–585, May 2013.

[SICP20] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[SKD$^+$20] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[SSLT12] S. Sengupta, P. Sturgess, L. Ladický, and P. H. S. Torr. Automatic dense visual semantic mapping from street-level imagery. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 857–862, Oct 2012.

[STGBS17] Yunming Shao, Charles Toth, Dorota A. Grejner-Brzezinska, and Lowber B. Strange. High-accuracy vehicle localization using a pre-built probability map. 2017.

[TCA$^+$17] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. *2017 International Conference on 3D Vision (3DV)*, Oct 2017.

[TSC20a] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation, 2020.

[TSC20b] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation, 2020.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 4-9 December 2017. Curran Associates Inc.

[WLY21] Robin Walters, Jinxi Li, and Rose Yu. Trajectory prediction using equivariant continuous convolution. *International Conference on Learning Representations*, 2021.

[WQA$^+$21] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[WQLC22] Ruihao Wang, Jian Qin, Kaiying Li, and Dong Cao. Bev lane det: Fast lane detection on bev ground. *arXiv preprint arXiv:2210.06006*, 2022.

[WRA18] J. Wu, J. Ruenz, and M. Althoff. Probabilistic map-based pedestrian motion prediction taking traffic participants into consideration. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1285–1292, June 2018.

[WS08] D. F. Wolf and G. S. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, April 2008.

[WSY$^+$18] Yuan Wang, Tianyue Shi, Peng Yun, Lei Tai, and Ming Liu. Pointseg: Real-time semantic segmentation based on 3d lidar point cloud, 2018.

[WWYK18] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[XGD$^+$17] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[YYZ17] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[ZLD$^+$21] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, 10-17 Oct 2021.

[ZLS$^+$] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. pages 8660–8669.

[ZSQ$^+$17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[ZSR$^+$19] Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[ZTTZ21] Yiyang Zhou, Yuichi Takeda, Masayoshi Tomizuka, and Wei Zhan. Automatic construction of lane-level hd maps for urban scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6649–6656, 2021.