# Online HD Semantic Map Construction and Evaluation

Vishal Jadhav, Vainktesh Swami

Department of Automotive Engineering, Clemson University

International Center for Automotive Research (CU-ICAR)

## Abstract

*An essential component of autonomous driving is constructing HD semantic maps. However, traditional pipelines require a large amount of human effort and resources to maintain the semantics of the map, which limits its scalability. During this project, we worked on the problem of HD semantic map learning, which dynamically builds the local semantics based on onboard sensor observations. We executed and tweaked the baseline semantic map learning method, dubbed HDMapNet. HDMapNet encodes image features from surrounding cameras or point clouds from LiDAR and predicts vectorized map elements in the bird's-eye view. To evaluate the map learning performance, we develop semantic-level and instance-level metrics. Lastly, we demonstrate that the showcased method can predict locally consistent maps. Here we study this novel map learning problem.*

### 1.1. Introduction

Traditional pipelines to construct such HD semantic maps involve capturing point clouds beforehand, building globally consistent maps using SLAM, and annotating semantics in the maps. This paradigm, though producing accurate HD maps and adopted by many autonomous driving companies, requires a vast number of human efforts. The traditional approach of construction of map follows a few basic principles:

- All data must be aligned with geometric map through which AV can place itself.
- Data from the autonomous vehicle and fleet is most trusted information.
- Leverage pre-existing data (map) and build on top of it.

Traditional approach looks something like figure1.

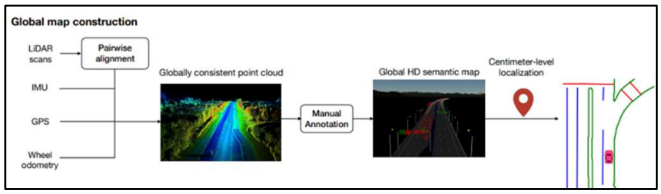


Fig. 1: Global semantic map by annotation

We studied semantic map learning method HDMapNet which produces vectorized map elements from images of the camera and point clouds from Lidar. In HDMapNet family, it generated based on input data. If it only uses camera input then HDMapNet(Surr), using Lidar data constructs HDMapNet(LiDAR) and HDMapNet(Fusion) if it takes both surrounding images and Lidar. In our case, we studied HDMapNet(Fusion) which is fusion of camera images and Lidar data.

### 1.2. Motivation

- Constructing HD semantic maps is a central component of autonomous driving. At SAE Level 5, HD maps are crucial for autonomous vehicles operate safely and behave naturally for riders and other vehicles on the road.
- Layered maps in Autonomous Navigation provide required precision.
- Geometric map allows the vehicle to localize itself and place itself accurately in the map; the semantic map helps it stays intact inside lanes.
- It gives autonomous vehicle understanding of how all the roads, lanes and crosswalk interconnected.

- Enables Autonomous Vehicles to understand interdependency of all the Elements in Environment.

### 1.3. Comparison between Traditional pipeline and Proposed method.

Traditional Method Involves a complex pipeline whereas the proposed model has an end-to-end Pipeline. Traditional approach is costly and labor expensive involving manual annotation and needs timely updates as the environment changes. The proposed model is cheap, automated and perceives the dynamic environment. Traditional Approach has Limited scalability due to usage of so many resources. HD semantic map learning constructs local semantics using on board sensors Using HDmapNet, features are extracted from Images and Lidar point cloud data, and vectorized map elements in bird eye view are predicted.

### 1.4. Resources Used

We used the following libraries:

- PyTorch 1.11.+cu11.3
- CUDA 11.3
- cuDNN 8.1 libraries

The training and evaluation were carried out on palmetto cluster using Nvidia Tesla V100 GPU and allocating 16 GB of memory for GPU and 30 GB for the CPU.

### 1.5. Related Work

Constructing a semantic map, the majority of extant HD semantic maps are manually or semi-automatically annotated on LiDAR point clouds of the environment, integrated from multiple sources. LiDAR scans acquired from high-end survey vehicles The most widely used SLAM algorithms are GPS and IMU. LiDAR images were fused using algorithms to create a highly accurate and consistent point cloud. To begin, pairwise alignment. ICP, NDT, and their variations are examples of algorithms. used to compare LiDAR data from two close timestamps employing semantic or geometric data Second, the formula for calculating accurate ego vehicle poses is a factor graph or a non-linear least-squares problem. This is necessary for creating a globally consistent map. However, it is still laborious and costly to maintain an HD semantic map since it requires high precision and timely update. In baseline paper, proposed local semantic map learning task is a potentially more scalable solution for autonomous driving. We change hyperparameters, parameters of vectorized map and activation function. And compared the performance of our model.

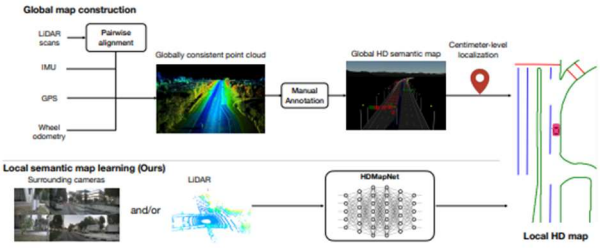


Fig. 2: Novel local map learning framework that makes use of on-board sensors to estimate local semantic maps.[1]

### 1.6 Overview of Model

A. HDMapNet(Fusion)

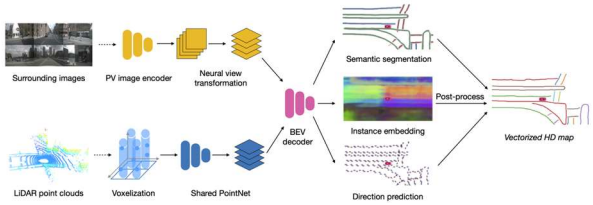


Fig. 3: A model overview. HDMapNet works with images and point clouds, outputs semantic segmentation, instance embedding and directions, and finally produces a vectorized local semantic map. Top left: image branch. Bottom left: point cloud branch. Right: HD semantic map.[1]

1) Image Encoder: Image encoder part have two components, perspective image encoder and neural view transformer.

**Perspective view image encoder:** Each image embedded by a shared neural network to get perspective view feature map. We get feature height, width and feature dimension in output.

**Neural view transformer:** For bird eye view, we need to convert image from pixel coordinates system to ego vehicle coordinate system. The relation between features in pixel coordinate system to camera coordinate system is modeled by a multi-layer perceptron. Then feature map in camera coordinate system to bird's eye view (ego

coordinate system) is obtained using geometric projection with camera extrinsic parameters.

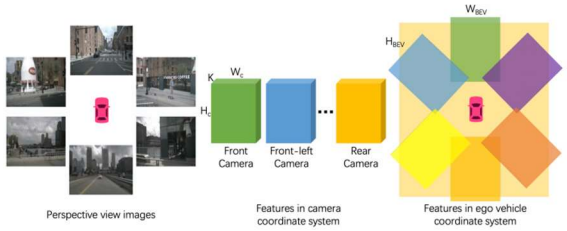


Fig. 4: Feature Transformation. Left: the 6-input image in the perspective view Middle: 6 feature maps in camera coordinate system, which are obtained by extracting features using image encoder and transforming these features with MLP; each feature map (in different colors) covers a certain area. Right: the feature map (in orange) in the ego vehicle coordinate system; this is fused from 6 feature maps and transformed to the ego vehicle coordinate system with camera extrinsic.[1]

2**) Point cloud encoder**: Point cloud encoder is a variant of PointPillar[3] with dynamic voxelization[4]. Voxelization is technique with which divide the 3d space into pillars and learn feature maps from pillar-wise point clouds. When projecting features from points to bird's-eye view, multiple points can potentially fall into same pillar. To aggregate features from points in a pillar, a PointNet[5] is used. The pillar wise features are further encoded through a convolutional neural network.

3**) Bird's eye view decoder**: This graph network includes instance-level and directional level information. Autonomous vehicles need vectorized lane lines so that they can be followed by self-driving vehicle. BEV decoder gives three outputs semantic segmentation, instance embedding and lane direction.

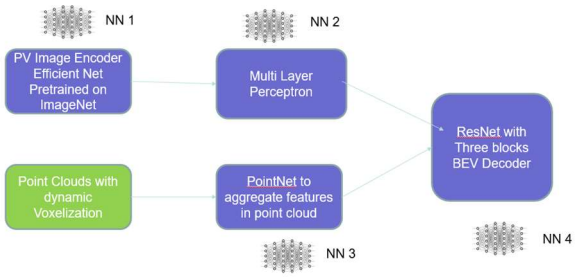


Fig. 5: Neural networks in overall architecture

**Overall architecture**: The BEV decoder is a fully convolutional network with branches, namely the semantic segmentation branch, instance embedding branch, and direction prediction branch. BEV decoder takes image feature map and point cloud feature map. If both exist then it concatenate both. If only one type of data exist then decode only that data.

**Semantic prediction**. The semantic prediction module is a fully convolutional network. Used cross-entropy loss for semantic prediction.

**Instance embedding:** Our module seeks to cluster embedding in each bird's eye view embedding. Embedding is done by labeling one a 1 and other 0. Most pixel in bird eye view map don't lie on the lanes, which means they don't have directions. Pixel with zero vector never do backpropagation during training. In baseline paper they use BCE loss, and one adoption is that they used "softmax" the label, that is if there are two 1s, then they normalize them to 0.5. But we directly used sigmoid as the activation function.

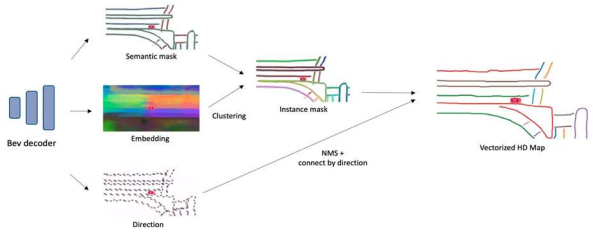


Fig. 6: Vectorization

**Vectorization**. During inference, first cluster instance embeddings using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Then non-maximum suppression (NMS) is used to reduce redundancy. Finally, the vector representations are obtained by greedily connecting the pixels with the help of the predicted direction.

### 1.7 Implementation

1) Dataset: For this implementation we used v1.0-mini nuScenes dataset. From this dataset we used CAM_BACK, CAM_FRONT, CAM_BACK_LEFT, CAM_BACK_RIGHT, CAM_FRONT_LEFT, CAM_FRONT_RIGHT and LIDAR_TOP data. Below figure is the visualization of data at one timestamp 2018-08-01-15-16-36-0400.

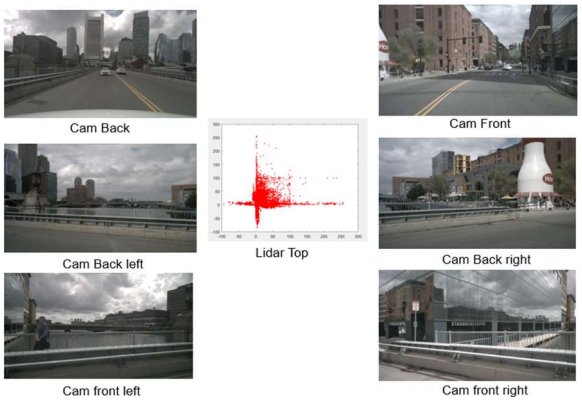


Fig. 7a: Data at single timestamp

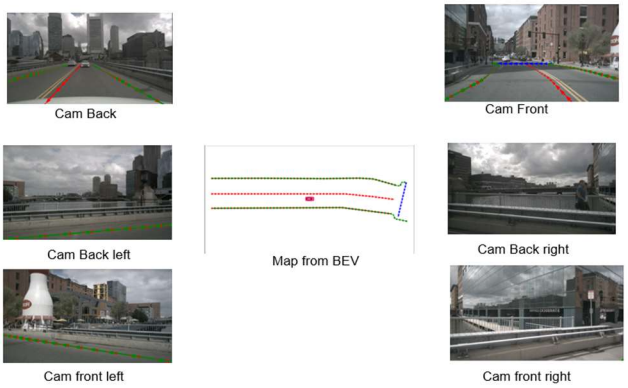


Fig 7b. Annotated Images

In addition to above, we used nuScenes-map-expansion-v1.3 dataset which contain maps. It contains map of boston-seaport, singapore-hollandvillage, singapore-onenorth and singapore-queenstown. Below figure is visualization of boston-seaport map. Addition to this, it contains calibrated sensor, ego pose and log information in .json format.

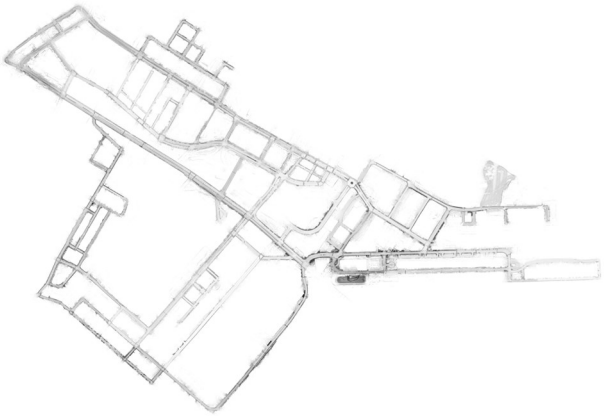

Fig. 8: Map of boston-seaport

2) Ground Truth Data; With using HDMapNetDataset we label the above dataset. For generation of ground truth on v1.0-mini dataset used argument like xbound an ybound. The ground truth generated in defined folder. Below figure is visualization labeled data.

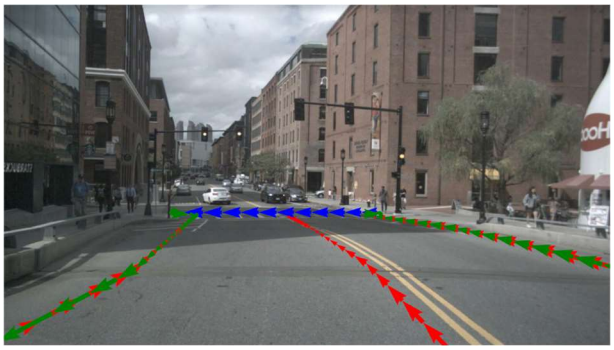

Fig. 9: Ground truth labeling on cam front image

In above figure, red arrows are lane boundaries and green are road boundary condition. Blue arrows are pedestrian crossing. Features in six perspective images projected to the bird's eye view. We can see that feature in map from bird eye view. The red car is ego vehicle. With feature it also positions the location of ego vehicle.

3) Training:

We trained model on palmetto cluster. First, we run the baseline code and then we run it on our model.

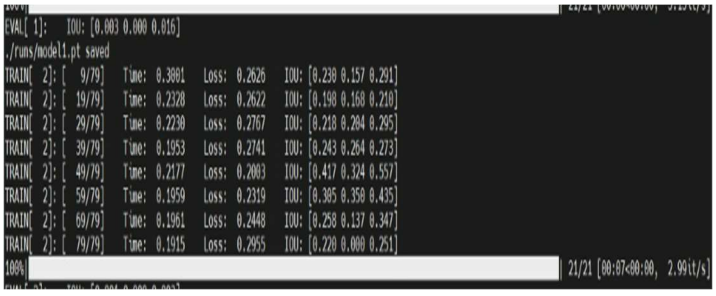


Fig. 10: Training on palmetto cluster

We run both model for 30 epochs. We tried iterations of learning rate, weight decay and epochs in case of our model. In addition to that we used sigmoid activation function instead of softmax. Softmax function was giving output of direction in form of probability, but we defined direction as 0 and 1. Which made it classification problem.

4) Result Visualization: Vehicle need information in form if vectorize map instead of pixel level information. Therefore, BEV decoder outputs sematic segmentation, instance embedding and direction prediction.

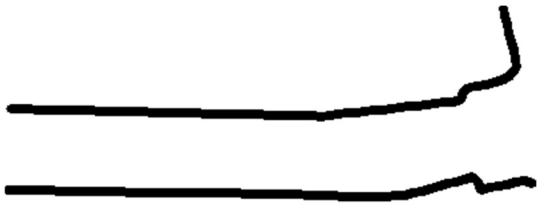

Fig. 11: Sematic Segmentation

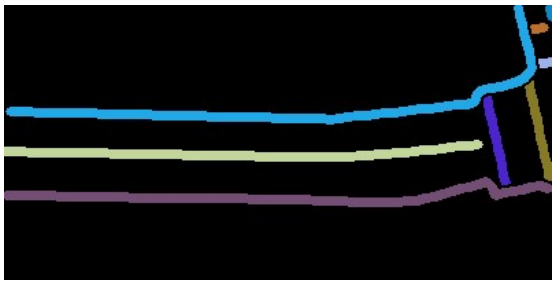

Fig. 12: Instance Embedding

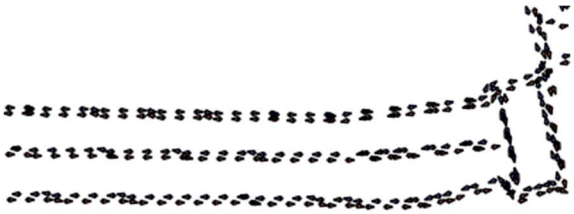

Fig. 13: Direction Prediction

Non-max suppression and direction prediction take place on output of BEV decoder which gives output as the vectorize map output which is in json file format.

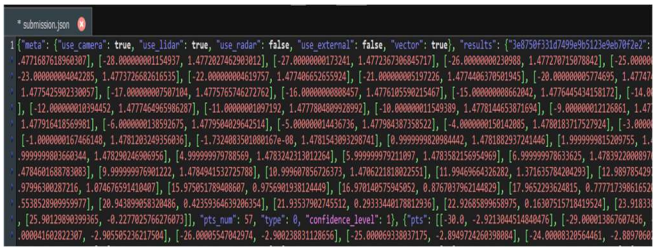


Fig. 14: Submission file which is result in JSON file format

Vectorize HD map can be visualize as follows.

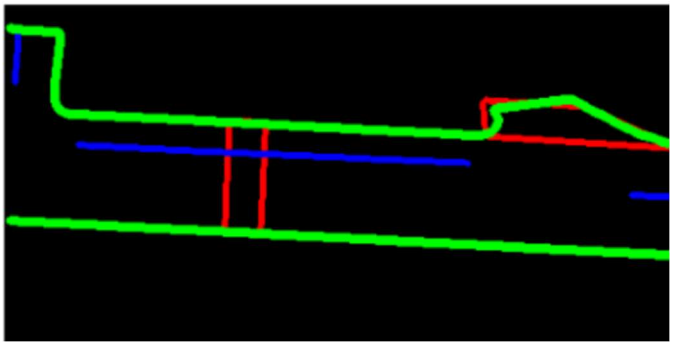

Fig. 15: Vectorize Map

5) Results:

Result of running base model and our model. The first column is baseline model result while second column is result of our model.

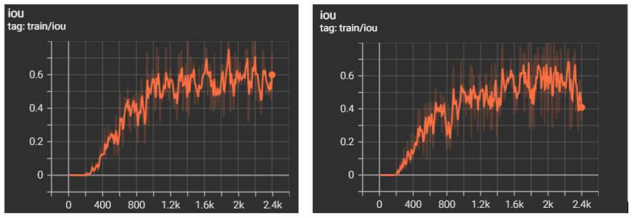


Fig. 16: Intersection over Union

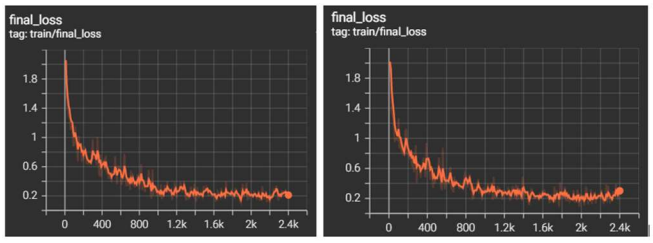


Fig. 17: Final loss

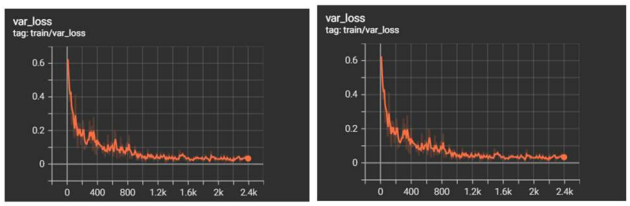


Fig. 18: Var loss on training dataset

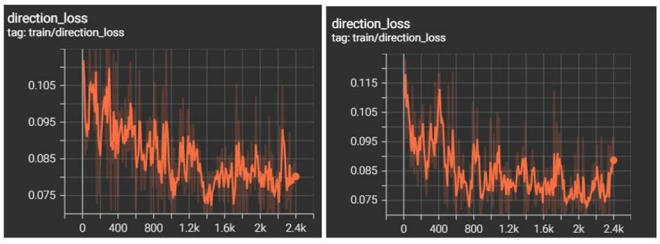


Fig. 19: Direction loss

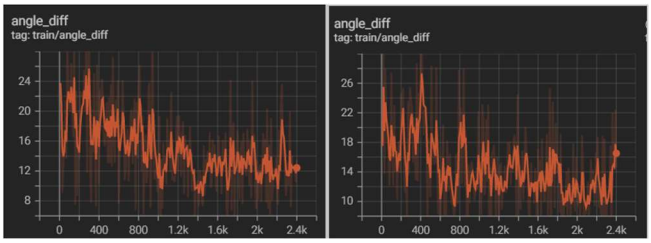


Fig. 20: Angle differnce

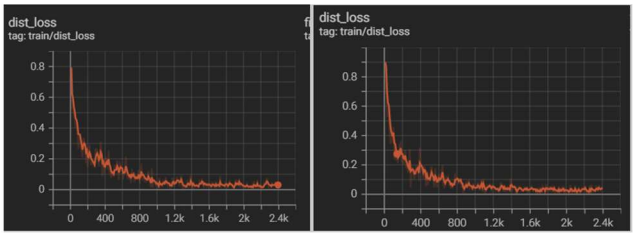


Fig. 21: Distance loss

6) Challenges**:**

We faced memory usage on palmetto. Job gets frequently terminated due to exceeded limit of allocated memory usage. Later we successfully figure out correct memory and computation resources to run 30 epochs on nuScenes dataset.

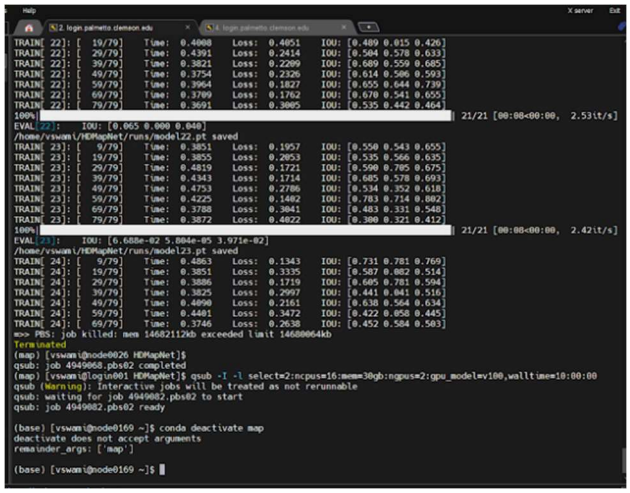

Fig. 22: Training stop on palmetto

In addition to that faced segmentation fault (core dumped) issue on palmetto cluster.


Fig. 23: Segmentation fault (core dumped)

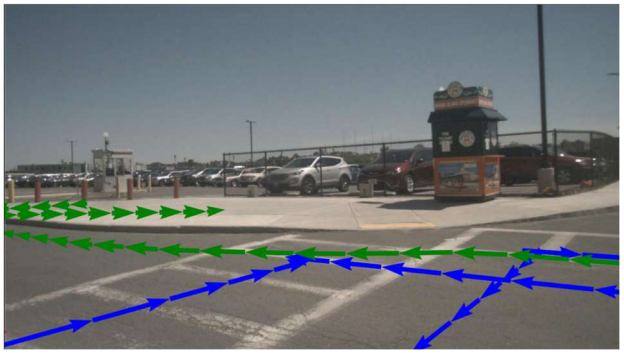

Fig. 24: Ground truth misalignment in some cases

We found that the ground truth lines projected on surrounding views had some misalignment with the real line elements. The results may be attributed to supposing the Z=0 for all map elements in labeling dataset. Optimization of model can solve this problem.

7) Contribution:

We changed argument and error in baseline code. We resolved this issue. And notify maintainer of repository about changes. They recognize our contribution.

8) Areas of Improvements:

- Solving misalignment in ground truth by optimization of model.
- 2.We can integrate Radar in addition to camera and Lidar data from nuScenes dataset.
- We can improve intersection over union in our results.

Conclusion:

We successfully run baseline model and our model. HDMapNet predicts HD semantic maps directly from camera images and/or LiDAR point clouds. Even though our baseline method of semantic map learning does not produce map elements as accurate, it gives system developers another possible choice of the trade-off between scalability and accuracy.

References**:**

1. Qi Li , Yue Wang, Yilun Wang and Hang Zhao, “HDMapNet: An Online HD Map Construction and Evaluation Framework.”

2. https://github.com/Tsinghua-MARS-Lab/HDMapNet

3. F. Yu, J. Xiao, and T. Funkhouser, “Semantic alignment of lidar data at city scale,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1722–1731.

4. A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

5. Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, “End-to-end multi-view fusion for 3d object detection in LiDAR point clouds,” in The Conference on Robot Learning (CoRL), 2019.

6. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.