Lane Point Cloud Prediction for Autonomous Driving

Shawn Manuel
Stanford University
CA, 94305, USA
sman64@stanford.edu

Abstract

3D lane detection is an essential functionality for Self-driving cars to stay within upcoming lane boundaries. We present a deep learning model for predicting the point clouds representing arbitrary lane curves from an image of the front facing view of a simulated autonomous vehicle. We also introduce an approach for gathering a dataset to train this model when specialized 3D sensing tools such as LiDAR are unavailable. Results show that our model is able to accurately predict the lane point clouds from a single image however highest accuracy is usually only achieved when the image is taken from the center of the track.

1. Introduction

Autonomous driving based on image perception has been a popular application of computer vision allowing for lane detection and obstacle avoidance without the need of LIDAR sensors [6, 1, 3]. Lane detection is essential for autonomous vehicles to stay within the boundaries of the road it is driving on when sharing the road with other drivers. As a result, predicting the 3D location of the lanes of a road is useful for planning the steering of the car in future time steps to keep the car within its lane as it is driving.

3D localization of objects in relation to an autonomous vehicle is an active area of research with recent approaches to detect the lane delimiters of urban roads. Two common approaches for scene modeling is with 3D point clouds [1, 3] and parameterizing object boundaries with polynomials [2]. Using polynomials to define the curvature of lanes has been effective for predicting the lanes of straight or smooth roads, however it is also reliant on this assumption during inferance and may not generalize to sharp turns. On the other hand, 3D point clouds are a less explored method which is more computationally intensive to calculate but can adapt to any track curvature. With the recent advancement in deep learning for image processing, it is becoming more and more feasible to compute large 3D point clouds at a rate close to the runtime required for real-time control of autonomous vehicles [1].

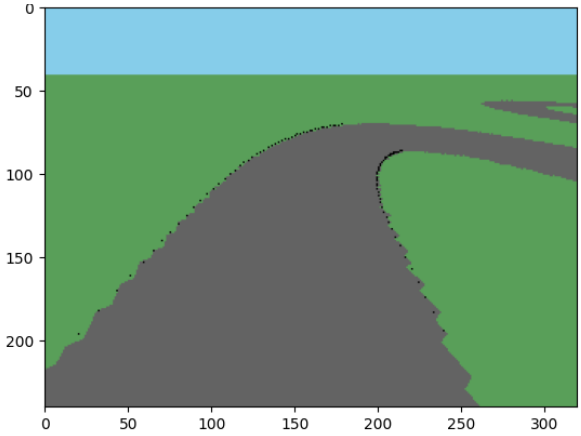

Figure 1. 3D view of simulated track segment

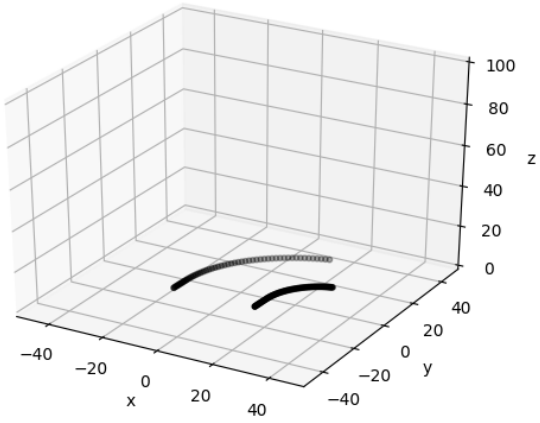

Figure 2. Point sequences representing track boundaries

The problem to be solved by this project involves 3D point cloud estimation from a monocular image to predict the boundaries of the track segment being observed from a camera's perspective view. Figure 1 shows the 3D first person perspective image of the track rendered by a camera mounted at the front of a simulated car. The goal is to predict the point cloud of 3D points that represent the track

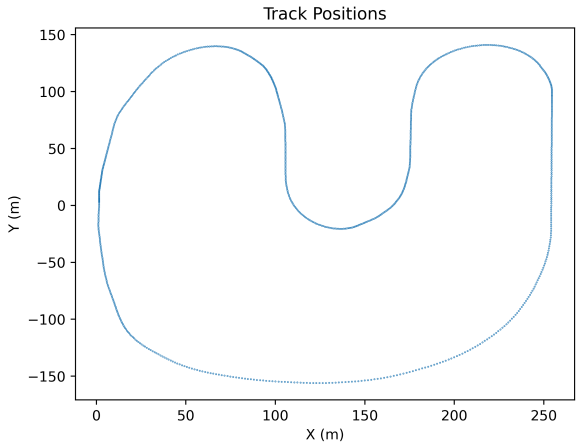

Figure 3. Track specification as point sequence

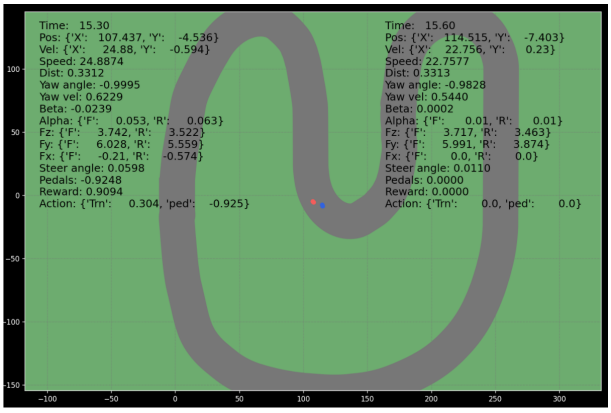

Figure 4. Birds eye view of track boundaries

boundaries up to a fixed length ahead of the car's position as shown by the 3D black points in Figure 2, which are also seen as the black dots in Figure 1 when projected onto the image plane. Since sampling 3D point clouds of lanes requires specialized tools such as LiDAR sensors [1], we will design a 2D car simulation environment to render input images of the track view and generate the boundaries of a fixed length segment of the track to be predicted. We will then define a convolutional neural network model to predict positions of the target output points.

This paper has the following structure: Section 2 provides an overview of the previous work in 3D lane detection for autonomous driving; Section 3 describes the approach for modeling the autonomous driving environment for data collection and training of the lane prediction model; Section 4 presents our experimental setup and results; and Section 5 concludes and suggests future research directions.

2. Background

In this section we provide an overview of previous approaches to localize nearby lanes for autonomous driving.

2.1. Related Work

Previous approaches for localizing lanes in 3D have involved using deep learning to process images of the road view from the vehicle's front facing camera and predict the location of lanes in 2D through pixel feature maps, or in 3D in the form of curves parameterized by polynomials.

Kim and Lee developed an image based lane detection method using convolutional neural networks to output an image mask where high pixel values correspond to regions in the input image that are lanes. This approach applied edge detection with a CNN to output a simplified edge map which was then passed through the RANSAC algorithm to filter out edges that are not lanes. While this approach can effectively identify lanes, it doesn't provide any 3D information about the location of the lanes based on the single image input [4].

Garnett et al. developed a deep learning approach to predict the 3D curves representing lanes as observed from the driver's position looking in front of the car [2]. This was taken from a single image of the driver's view and input to a neural network which predicts the parameters for curves defining the lanes. This approach first learns a mapping from the projected input image space to a birds eye view of the lane, from which it then fits polynomials to each lane detected from the top view image. This approach was able to accurately match lanes observed in the image to 3D curves, however, it was mostly trained on smooth and slow-curving roads whereas sharp turns may have high variance when fit with polynomials learned for straight roads.

Finally, Chen et al. use LiDAR sensors to generate 3D point clouds to reconstruct the scene that a car is driving within. It doesn't use deep learning for lane detection, but uses a Bayes filter to localize the car within the scene. Although the point clouds generated could be useful for lane detection, generating a dataset and extracting lanes manually is unfeasible in a short time frame which emphasises our motivation for using a car simulator to generate lane point clouds.

3. Technical Approach

In this section, we present the approach for training a deep learning model to predict the point cloud representing the lane boundaries of a visible road segment from a projected image of the road. Since measuring the 3D point cloud sequence of lanes in the real world is expensive, we will use a simple 3D car simulator to render views of the road as input and calculate the corresponding point cloud that should be output by the network.

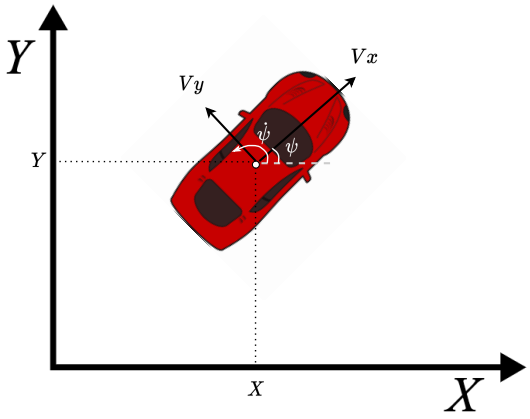

Figure 5. Kinematic (X, Y, ψ) and dynamic (Vx, Vy, $\dot{\psi}$) properties describing the state of the car at a given time.

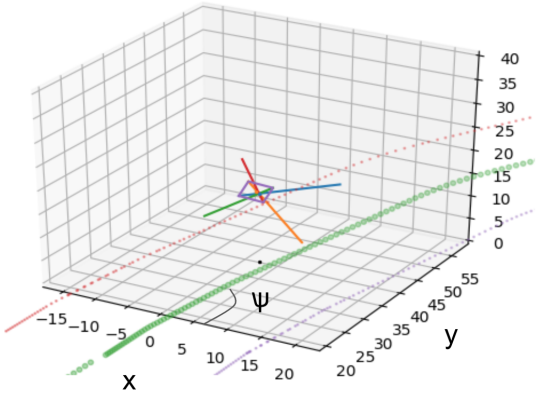

Figure 6. Camera model field of view from current orientation and location in world coordinates.

3.1. 3D Lane Point Cloud Modeling

The first step in designing a 3D car simulator is to define a sequence of 2D points on the x-y plane that form the track as shown in Figure 3. We also specify a desired track width used for generating the sequence of boundary points on either side of the track resulting in a path shown in Figure 4.

The second component is the dynamics of the car which will be defined by simple Bicycle Model vehicle dynamics [5] which determines the state transition of the car at each time step. The dynamics operates primarily on the car's position (X, Y), heading ψ, forward velocity V_x, lateral velocity V_y and yaw velocity $\dot{\psi}$ as shown in Figure 5.

Given the (X, Y) location and the heading ψ of the car, a camera model can be defined to view the scene at the current time step. This camera model is located at the car's (X, Y) location in a 3D world space at a fixed height h and facing the car's ψ direction taken from the positive x-axis as shown in Figure 6. In order to render the image viewed by the camera and define a target lane point cloud, a set of transformation matrices is needed to map true positions of the track boundaries to be relative to the camera. The translation matrix T is defined to be the 4×4 transformation matrix that transforms the car's location in homogenious coordinates $(X, Y, 0, 1)$ to the camera's origin $(0, 0, 0, 1)$ on the x-y plane. Then, a rotation matrix R is defined to be the 4×4 transformation matrix that rotates the camera from pointing along the positive x-axis to rotate a certain pitch in the vertical direction, and angle ψ about the z-axis. Finally, the camera intrinsic matrix K is defined to map 3D points into pixel coordinates on a specified image plane.

The image of the perspective view of the road from the car's location can then be rendered with ray tracing from the focal point for each pixel coordinate (u, v) with the ray direction $(dx, dy, dz) = R^T K^{-1} \times (u, v, 1)^T$. An example of a rendered image is shown in Figure 1.

For each image state rendered, the prediction target is a point cloud representing the lane boundaries of a fixed length segment of the track starting at the camera position. The starting track index is found as the index of the point nearest to the car's position. First, these 3D points need to be transformed from world coordinate system to the camera coordinate system as shown in Figure 2 by multiplying the coordinates by the T and R matrices.

3.2. Model Architecture

The model for predicting the fixed point cloud sequences of the two lanes from the images of the the road will be a neural network consisting of convolutional layers and linear layers to map each input RGB image into a set of 3D coordinates. The network has 4 convolutional layers each with a kernel size of 3, stride of 2 and padding of 1 mapping the channels from 3 to 4, 8, 16 and 32 respectively in sequence. Then, the output of the last convolutional layer is flattened and passed to two linear layers with 1000 hidden units before being transformed to a shape representing a set of 3D points. The standard mean squared error loss is used to train the network to output the 3D positions corresponding to target point cloud.

4. Experiments

To evaluate the effectiveness of the model for predicting lane point clouds, we generated a dataset of input image to output point cloud by first training a controller policy with Reinforcement Learning and running the optimal policies for 5 rollouts of the circuit shown in Figure 4 with each rollout being approximately 475 time steps long. Between each rollout, the percentage of random actions was increased from 0% to 20% in %5 intervals to sample images from different scenarios where the car may drive off

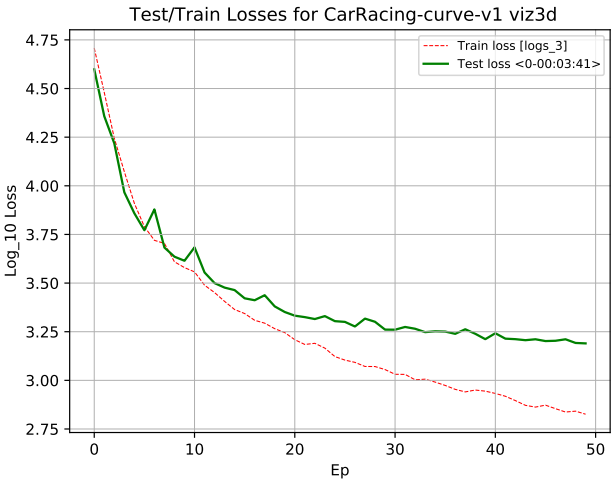

Figure 7. Training and test loss curve against epoch iteration. Losses are displayed in log_{10} scale

the road and the two lane boundaries may not be visible in the image. During each rollout, a random process noise was applied to the pitch (vertical viewing tilt) to sample different projected views of the lanes.

For each input-output pair, the image rendered was of size $240 \times 320 \times 3$ and the output consisted of a sequence of 50 3D points for the left and right lanes taken at the index of the track closest to the car. The two lane sequences of x-y positions are then transformed into camera space and concatenated to form an output size of 300. When certain 3D points were not visible on the screen, the corresponding 3D coordinates were set to $(0, 0, 0)$ for those points. Figure 7 shows the loss vs iteration plot for training the network on 2400 images with an 80-20 train-test split.

To evaluate the effectiveness of the model, we sampled new images with the policy at different regions on and off the track and compared the location of the predicted points with the true locations of the lane points gathered based on the car's current location. We categorized the test samples into three scenarios where the first has images when the car is in the center of the track with view of both lanes as shown in Figure 8 and Figure 9. The second has images where the car is on the edge of the track and part of one of the track boundaries is not visible as shown in Figure 10. The third has images from when the car is off the track and viewing distant parts of the track as seen in Figure 11. The 3D plot on the right side of each image is the point cloud from camera space rotated to be flat on the x-y plane to normalize the variation in pitch angle. The predicted point cloud is shown in red with the ground truth shown as black points.

For the first scenario of driving on the center of the track, it was found that the model is able to predict the point clouds closely for straight parts of the track as seen in Figure 8 as well as on turns as seen in Figure 9. The results were also stable for different pitch angles. This is expected since changing the pitch would also result in the target point cloud being rotated about the camera origin which can be learned by a neural network since the pitch angle limits are approximately 20 degrees which is relatively linear. Thus, when the output points of the network are rotated back to the x-y plane, there is little deviation in the z values of each point.

For the second scenario of driving on the edge of the track, it was found that when part of one side of the track was not visible, the points for that side of track would linearly interpolate from the camera center $(0, 0, 0)$ to the points where the lane became visible. This is seen in Figure 10 where the right lane curves toward the origin. This effect is due to the regression loss used to train the network which is prone to learning interpolated predictions between multimodal target distributions.

For the third scenario where the car is off the track, the predicted point cloud is less interpretable as two distinct lanes and instead is more clustered randomly around the camera origin as shown in Figure 11. This can be due to the reduced proportion of training samples from this scenario as the car would only deviate from the track for a few time steps in the two rollouts where the maximum random actions were selected. As a result, the network may perceive the large green region in the center of the screen to be similar to the scenario where the track is not visible and hence predict points around $(0, 0, 0)$ as this would lead to a lower overall loss than predicting lane clouds for track regions observed at a further distance from the camera.

5. Conclusions and Future Work

In this paper we presented a deep learning model for predicting the point clouds of the nearest lane boundaries with a fixed number of points. We also presented an approach for efficiently training this model on 3D projection views of a track from a simulated car's point of view with target point clouds for each image. Results showed that the model is effective for predicting the lane point clouds from an image viewing the track from the center of the lane, and generalizes to different pitch angles. For predictions where part or all of the closest lanes are not visible in the image, the predictions have much higher variance and are less interpretable.

For future work, the reliability of the model in regions away from the center of the track can be improved by using a recurrent neural network maintaining a history of observed images to predict the new location of the lanes based on its movement throughout the history. With further improvement in the generalizability of the model, the lane predictions could also be used for image state preprocessing in reinforcement learning from visual observations where an agent can use the 3D lane predictions to learn a policy that stays within the predicted lanes.

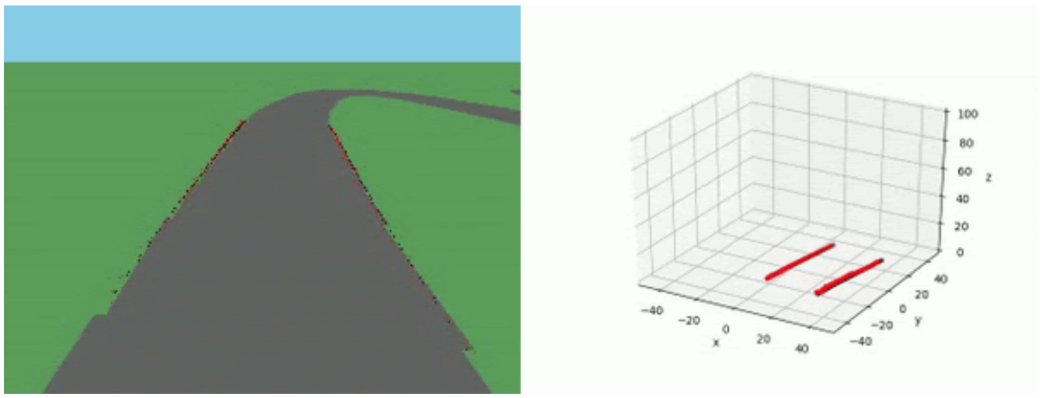

Figure 8. Accurate point cloud prediction from image in the center of straight track.

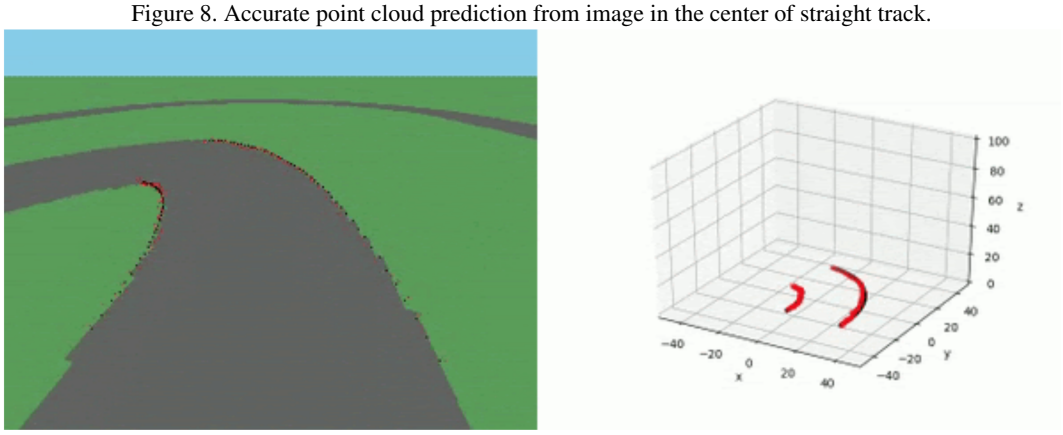

Figure 9. Accurate point cloud prediction from image in the center of a turn.

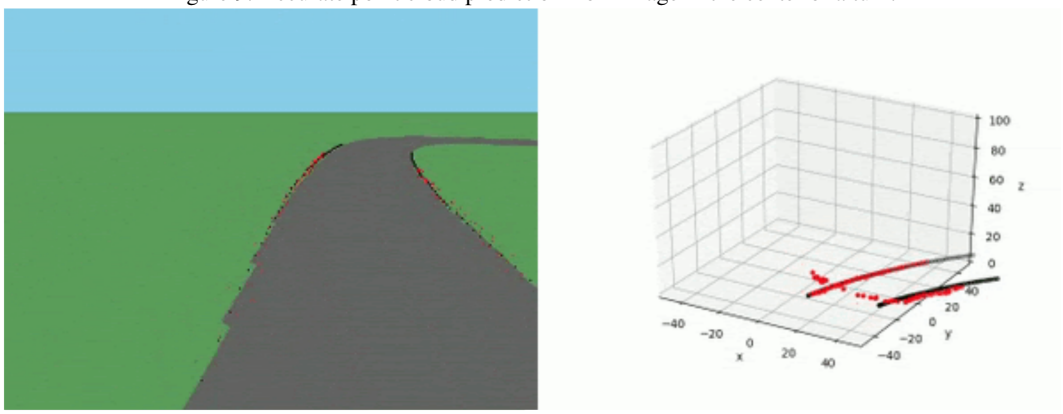

Figure 10. Predicted point cloud when image on edge of track interpolates hidden points from the origin to the visible lane.

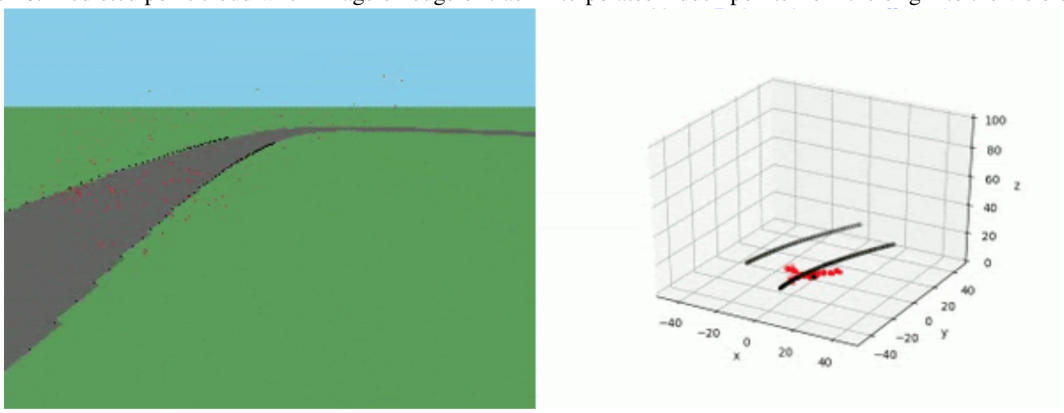

Figure 11. Predicted point cloud is clustered around origin when camera is off the track.

Supplemental Material

The source code for this project is available on github[1] as well as a supplementary video[2] showing the three scenarios that the model was evaluated on for more figures comparing the predictions with the ground truth.

References

[1] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving. *arXiv preprint arXiv:2003.00601*, 2020.

[2] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. 3d-lanenet: End-to-end 3d multiple lane detection, 2019.

[3] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[4] Jihun Kim and Minho Lee. Robust lane detection based on convolutional neural network and random sample consensus. In Chu Kiong Loo, Keem Siah Yap, Kok Wai Wong, Andrew Teoh, and Kaizhu Huang, editors, *Neural Information Processing*, pages 454–461, Cham, 2014. Springer International Publishing. ISBN 978-3-319-12637-1.

[5] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017. doi: 10.1109/IVS.2017.7995816.

[6] Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107623. URL https://www.sciencedirect.com/science/article/pii/S003132032030426X.

[1] https://github.com/shawnmanuel000/3DLanePointCloudPrediction

[2] https://github.com/shawnmanuel000/3DLanePointCloudPrediction/blob/main/supplementary%20results%20video.mov