3D SEMANTIC SEGMENTATION WITH QUASI SOLID-STATE LIDAR

An Undergraduate Research Scholars Thesis

by

YIFAN SUN

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Dezhen Song

May 2023

Majors: Computer Science
Statistics

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Yifan Sun, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

Page

ABSTRACT

3D Semantic Segmentation with Quasi Solid-State LiDAR

Yifan Sun
Department of Computer Science & Engineering
Texas A&M University

Faculty Research Advisor: Dr. Dezhen Song
Department of Computer Science & Engineering
Texas A&M University

Current, most 3D semantic segmentation models for autonomous driving are mainly trained on spinning Light Detection And Ranging (LiDAR) data because spinning LiDAR sensors have been one the most popular sensors for autonomous driving vehicles and there is an abundance of spinning LiDAR dataset available to the public. However, spinning LiDAR sensors are costly and requires large amounts of energy to operate. The newly emerged quasi solid-state LiDAR sensors are more cost efficient and require lower amount of energy to operate on autonomous driving vehicles. If we reuse the current models pretrained with spinning LiDAR data on quasi solid-state LiDAR data, its performance is below expectation. Currently there are not enough quasi solid-state LiDAR data to train 3D semantic segmentation deep learning models effectively, and the data pattern for quasi solid-state LiDAR is mostly different from the spinning LiDAR data.

This research will first develop a visualization tool and evaluate the existing 3D semantic segmentation models that are pretrained with spinning LiDAR data on some small scaled quasi solid-state LiDAR data. The performance of the model is under expectation, which calls for the

retraining of the 3D semantic segmentation model on quasi solid-state LiDAR. The model chosen is SPVNAS. Since there are few publicly available large scaled quasi solid-state LiDAR datasets, several approaches are taken in parallel to synthesize the suitable dataset for training. The final approach decided was rich point subsampling, which takes a reconstructed scene of rich point cloud, filter out the point that are not in the vehicle's field of view, and sample the points according to the data pattern for quasi solid-state LiDAR data. The processed data is fed into the SPVNAS model for training, validation, and testing. This final model for 3D Semantic Segmentation is the main product of the research.

DEDICATION

This research is dedicated to the 12th Unmanned Team, which is the official autonomous driving team of Texas A&M, for the GM/SAE Autodrive Challenge II. Currently it is year 2 of the four-year Autodrive Challenge, and each of our team members are working on different components of autonomous driving so that our school will achieve great performance in the final competition.

ACKNOWLEDGEMENTS

Contributors

I would like to thank my faculty advisor, Dr. Song, and my mentor Di Wang, for their guidance and support throughout the course of this research. They have helped me lay out the path for my research and develop solutions for each of the components. It would be impossible for me to finish the research successfully without your support.

Thanks also go to the Undergraduate Research Scholars (URS) thesis program at Texas A&M, which offered me the opportunity to deposit my thesis in the university's archive and helped me professionalize the thesis and the research process.

This work is also made possible by the 12th Unmanned Team for various types of datasets provided that were crucial to the study of data patterns.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This project did not receive any sources of funding.

NOMENCLATURE

LiDAR	Light Detection And Ranging
URS	Undergraduate Research Scholars
ROS	Robot Operating System
mIoU	mean Intersection over Union
IoU	Intersection over Union

1. INTRODUCTION

1.1 3D Semantic Segmentation

3D Semantic Segmentation is the detection and identification of various objects in 3D scenes. In particular, it is the task of classifying and assigning every point in the scene with an associated object label, such as pedestrian or road sign [1]. This task is one of the most crucial components in autonomous driving, as it could provide the vehicle with a large-scale understanding of the 3D space around it, so that it has the information to make decision accordingly. For example, if a pedestrian is detected in front of a moving autonomous vehicle, it will most likely brake or turn around based on current speed and direction. The performance of an autonomous vehicle is largely associated with the accuracy and performance of the 3D semantic segmentation model built in. The vehicle can only make fast and correct decisions if the model can identify surrounding objects quickly and accurately.

Currently, the most modern approach for 3D Semantic Segmentation is deep learning. It is a supervised machine learning technique which utilizes neural networks, which are networks of artificial neurons in layers connected together by weights, to teach computers to perform certain tasks by giving them examples. The deep learning models for solving 3D Semantic Segmentation will be trained with a large quantity of 3D point cloud and corresponding labels. Then the point cloud data will propagate through the layers and the model will predict which label has the highest probability to represent each group of 3D data. The model is initialized with random weights, and each time it predicts a label wrong, it will back propagate and update the weights in the best directions accordingly to increase the accuracy. Deep learning models for 3D semantic segmentation typically employ convolutional neural networks (CNNs) that can learn

rich features from raw point cloud data. These models usually consist of several convolutional layers, followed by pooling and upsampling layers, and a final classification layer that outputs the semantic label for each point. They use volumetric convolutions to extract features from 3D data and integrate spatial and semantic information for accurate segmentation. Additionally, recent advancements in deep learning have led to the development of novel architectures, such as graph neural networks (GNNs), that can better capture the complex relationships between different parts of the 3D scene. Nevertheless, the training process requires very large amounts of training data, which is one of the major challenges of this research. It is impossible for me to manually input correct labels for large 3D datasets without labels that were provided to me by the 12th Unmanned Team, because it requires huge amount of time, and is also prone to errors.

1.2 LiDAR

1.2.1 Overview

In autonomous driving, we have various ways to acquire the information about the surface of the areas around the vehicle. With the unprecedentedly fast development of 3D acquisition technologies, many types of 3D sensors become increasingly available and affordable, and LiDAR is one of them. LiDAR is a remote sensing method which uses light in pulsed laser form to measure ranges to the surfaces. In combination of 2D images captured by camera, 3D data contributes largely for a better understanding of the surrounding environment for machines.

3D data collected by LiDAR is represented in point clouds representation, which preserves the original geometric information in 3D space [2]. They are basically discrete sets of data points in space, and the combination of these 3D data points can visualize the shape of the surfaces. The high-density point clouds representation has been widely used to train deep

learning models for autonomous driving, especially 3D Semantic Segmentation. However, it is difficult to evaluate the performance of 3D Semantic Segmentation models on unlabeled data without visualization tools. The model results are 3D points assigned with predicted labels, but if we lack the ground truth of the labels, we have to visualize the points and their labels in 3D space and compare the label with what objects we see them as.

1.2.2 Spinning and Quasi Solid-State LiDAR

Spinning LiDAR has a spinning physical structure as the laser constantly spins around to get the 360 degrees view of the environment. On the other hand, the quasi solid-state LiDAR has a fixed laser but moving mirrors to modulate the laser [3]. Due to the different architecture of the two types of sensors, the data patterns produced by them are largely different. The data produced by the spinning LiDAR is sparser than the data produced by the quasi solid-state LiDAR since the quasi solid-state LiDAR, which only spins the mirror instead of the entire structure, can spin more frequently than the spinning LiDAR. And for this reason, we can only transform quasi solid-state LiDAR data into spinning LiDAR data by studying the different data patterns [4] produced by them and take subsamples from the quasi solid state LiDAR data and discard the intensity value. On the other hand, we cannot transform spinning LiDAR data to quasi solid-state LiDAR data because of lack of information about the surface. This is one of the major difficulties when building the 3D semantic segmentation model for quasi solid-state LiDAR data.

Spinning LiDAR has been one of the most widely used 3D sensors for autonomous driving vehicles in the last decade, so we have been mostly using spinning LiDAR to develop models for various purposes because of the prevalence of dataset online. Now as we transition to using quasi solid-state LiDAR because of its smaller size, reduced cost, and the 12th unmanned team's sponsor General Motor provided them with a vehicle equipped with the quasi solid-state

LiDAR, we lack the quantity of data to train deep learning models since the technology is relatively new and there are few publicly available datasets.

However, the existing 3D semantic segmentation models with spinning LiDAR data may work well for quasi solid-state LiDAR because the underlying neural network may label the object by learning the shape of each surrounding surface represented by point cloud and make the decision. This means that there is some chance that the existing models can learn the information about the surfaces even better with quasi solid-state LiDAR data than the spinning LiDAR data.

2. METHODS

2.1 Original Data

2.1.1 Mcity Full Drive & Push Cart

Mcity Full Drive and Mcity Push Cart are the two datasets collected by the 12th Unmanned Team and were authorized to be used for this research. They are in the format of folders of .bag files. Each folder contains the data collected for one specific route, and they were generated by LiDAR sensors and cameras on the vehicle that was driven along the route each time after a fixed time interval. The .bag file is a file format in ROS for storing ROS message data, including the point cloud data. The main differences between the two datasets were that Mcity Full Drive was generated by spinning LiDAR sensors, and Mcity Push Cart was generated by quasi solid-state LiDAR sensors. Moreover, the routes taken by the two datasets were mostly different, but they are all in Texas A&M College Station and Rellis campuses.

These two datasets were mainly used for initial evaluation of 3D Semantic Segmentation deep learning models to see whether the models were suitable for various patterns of data, and the final model should have good performance on the Mcity Push Cart dataset. However, the LiDAR data needs to be extracted and transformed into compatible formats (SemanticKITTI) to be fed into the deep learning models. In addition, these two datasets do not contain object labels, so they cannot be used for training deep learning models.

2.1.2 SemanticKITTI

SemanticKiTTI [5] is a large-scale dataset which contains rich sensory information recorded with spinning LiDAR sensors across several suburbs of Karlsruhe, Germany. Because of its abundance of data, most advanced 3D Semantic Segmentation deep learning models are

trained with it. Only dataset of this scale can be used for training deep learning models with considerable performance. The dataset is divided into many individual sequences, including the Velodyne, which is sequences of point cloud files in binary format, and the labels, which is sequences of true object labels that correspond to each training Velodyne file.

The binary point cloud files contain arrays of four-point tuples: (x, y, z, intensity). These values are represented as floating point numbers. The x, y, and z are the three-dimensional coordinate of the point, and the intensity ranged from 0 to 1 represents the relative return strength of the laser beam, affected by the surface reflexibility, angle of arrival, range, roughness, and moisture content.

The labels files contain series of pairs of point coordinates and object labels. The object labels are unsigned integers that correspond to objects in the object map. For example, 31 represents unrecognized object and 8 represents pedestrian. If a point cloud file is labeled, it will have the same file name prefix as the labels file within the same sequence, and all the points should have a label, even if the object is unrecognized.

2.2 Model Selection

The 3D semantic segmentation deep learning model chosen for the task should have at least above 50% accuracy on identifying the objects, and it also need to be pretrained with the scene information. It is unnecessary to build a deep learning model from scratch because there are too many components that need to be wired together and parameters that need to be tuned in order to achieve a good performance on the task. 3D semantic segmentation is a common task that scholars around the world have been researching on. There is a competition on paperwithcode.com called 3D Semantic Segmentation on SemanticKITTI in which scholars around the world use the same point cloud and labels dataset from the SemanticKITTI dataset

and they will build and train their deep learning models on this dataset. Then, they can post their model on the website with the paper, and the system will evaluate their model with the same benchmark score mIoU.

mIoU is one of the most commonly used benchmark score for image classification and object detection which stands for mean Intersection over Union. The IoU, Intersection over Union, is a metric that is the ratio between the area of overlap and the area of union between the ground truth and the predicted areas. Therefore, higher IoU indicates that the predicted and the ground truth is more similar, and an IoU of 1 means that the predicted and the ground truth is exactly the same because the union equals the intersection. The mIoU score is used in the 3D semantic segmentation task because the output is object labels, and there are multiple testing datasets, so the mean of their IoU evaluated on the model uploaded is more representative of the prediction accuracy of the model.

I have examined several models that are the most recent and with highest mIoU scores. The model chosen for 3D semantic segmentation on quasi solid-state LiDAR data should only require point cloud and labels data to train, and it should use the setup that is publicly available. One of the model I've tried is 2DPASS [6], which takes a combination of images and point cloud data to establish a comprehensive view of the environment. It is the most advanced model currently because some of the objects can be better identified with 2D data, and others can be better identified with 3D data. For example, the lanes on the road can only be seen with 2D camera images, whereas the distribution of bushes with the same color can be better detected with 3D LiDAR sensors. Unfortunately, this model require image information to train, which is almost impossible to generate based on a simulated environment or other sources, except for taking the physical vehicle equipped with cameras and LiDAR sensors to go around many places

to collect the data, which exceeds our budget. Another model I attempted to use was the Cylinder3D [7]. It only requires point cloud and labels data to train and it has a very high mIoU score. However, this model utilizes a depreciated version of a package which is the core of the model, so the setup for this model is not applicable.

Finally, I decided on the model SPVNAS [8], which is implemented with Sparse Point-Voxel Convolution (SPVConv). It is a light weighted 3D module with high resolution point-based branch, which preserve the fine details of the scenes. Then it uses the 3D Neural Architecture Search (3D-NAS) to search for the optimal neural network architecture efficiently. This model is verified to be fast and accurate on the 3D semantic segmentation task and achieves a mIoU score of 0.664, ranked 4th on the leaderboard.

2.3 Model Evaluation

2.3.1 Data Conversion

Since the existing 3D semantic segmentation models pretrained with spinning LiDAR data may also work well for the quasi solid-state LiDAR data, the first step would be the evaluation of the best performing models on both spinning LiDAR and the quasi solid-state LiDAR data by examining the result after a testing session.

The model evaluation will use the Mcity Full Drive and Mcity Push Cart datasets because they are small scaled datasets that take a small amount of time to run on. From previous knowledge we infer that the intensity value may not be accurate if we synthesize quasi solid-state LiDAR data from other sources, so I attempted to generate spinning and quasi solid-state LiDAR datasets with and without intensity that is compatible with the model to compare their performance on the deep learning model that is pretrained with spinning LiDAR. Notice that the

data all need to be transformed into KITTI format because most of the best models are trained with KITTI datasets and they can only take point cloud data in the KITTI binary format.

To make the data compatible with the deep learning model, I need to extract them from the .bag files with ROS system and convert them into binary files. Then I will select a pretrained version of the deep learning model and run a test session with the converted dataset, after which I extracted all the labels from the result and visualize them in an interactive three-dimensional space to compare the object label results with their shape.

2.3.2 Evaluation Results

The following figures are plots of the point cloud with labels that are produced by providing one sample from four different datasets as the testing set for the SPVNAS model with pretrained version SemanticKITTI_val_SPVNAS@65GMACs. The datasets used are: Mcity Full Drive with intensity (spinning LiDAR with intensity), Mcity Full Drive without intensity (spinning LiDAR without intensity), Mcity Push Cart with intensity (quasi solid-state LiDAR with intensity), Mcity Push Cart without intensity (quasi solid-state LiDAR without intensity). Notice that the same scene was sampled for Mcity Full Drive and the same scene was sample for Mcity Push Cart. Different samples from other scenes were also plotted, but the qualitative results were similar. Different pretrained version of the model were also tested, and the resulting labels were also largely the same. The figures shown below are the representatives from the four different datasets, and only one perspective is displayed for the 3D environment. Each type of object is encoded with a unique color shown in the plot. A real-world image is also provided for each scene for better reference of the surrounding environment.

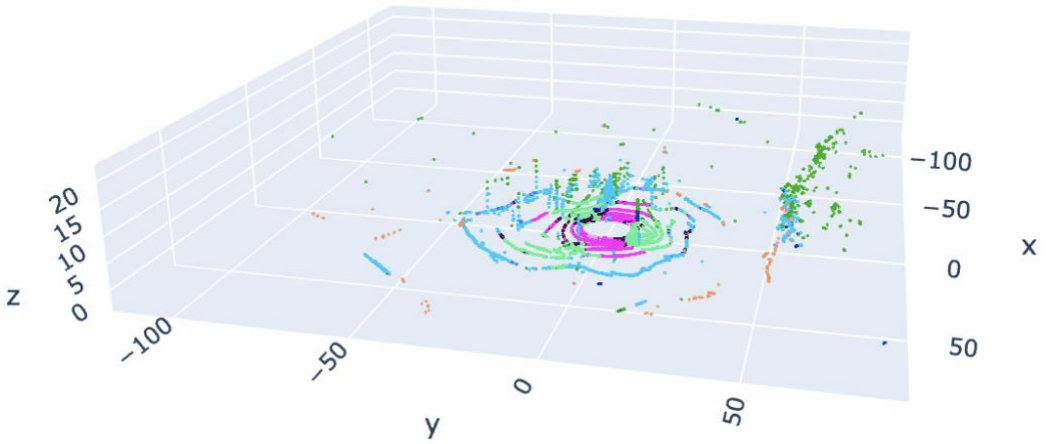

Figure 1: Spinning LiDAR without intensity

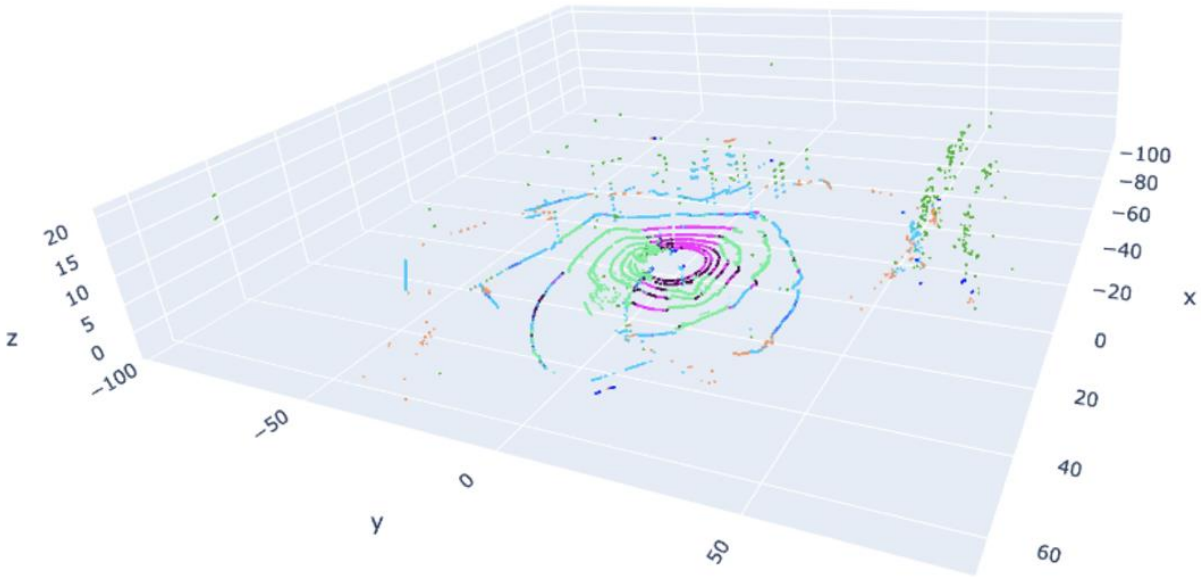

Figure 2: Spinning LiDAR with intensity

Figure 3: Spinning LiDAR real-world image

From the results produced by the spinning LiDAR datasets, it is observed that different objects are detected and distinguished very well because there are large clusters of data with uniform color representing types of recognized object, but the spinning LiDAR dataset without intensity has a clearer boundary between different objects, while the spinning LiDAR dataset with intensity has mix of different labels among several surfaces, so a more detailed look of the surface is achieved with the intensity information. From the real-world image, it is observed that the vehicle was at an intersection and the surrounding environment may have curbs, lawn, and poles, which are mostly identified in the model.

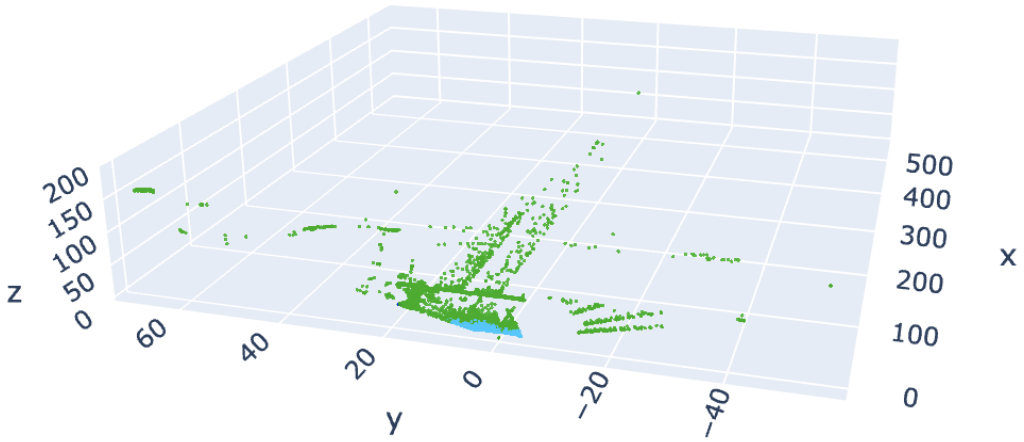

Figure 4: Quasi solid-state LiDAR without intensity

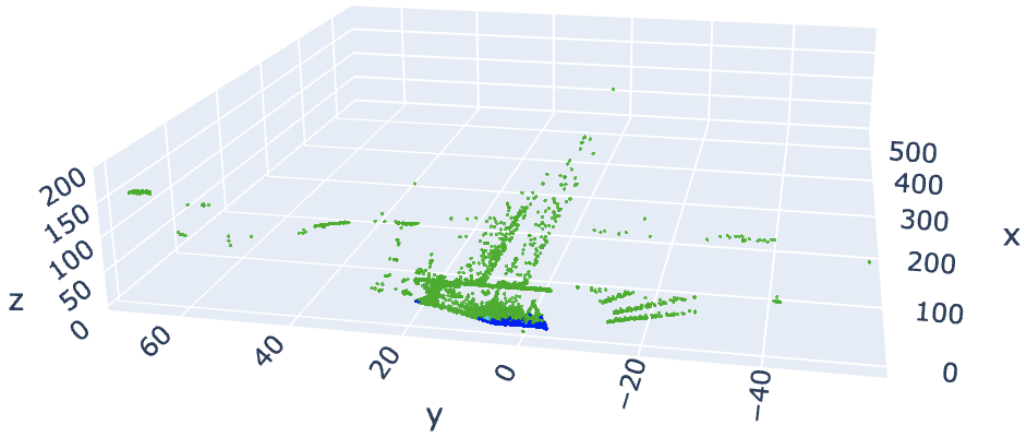

Figure 5: Quasi solid-state LiDAR with intensity

Figure 6: Quasi solid-state LiDAR real-world image

From the results produced by the quasi solid-state LiDAR, it is observed that there is almost no difference between the results with and without intensity except for the different object labels assigned for the cluster of points close to the vehicle. However, the overall result for the quasi solid-state LiDAR data is disappointing. Almost all the points are classified with the same label, which is “unrecognized”. This may infer that the SPVNAS model pretrained with spinning LiDAR data is not directly applicable to the quasi solid-state LiDAR despite the fact that they have similar data patterns. This urges for the next stage of the project, which is the synthesis of large quantities of quasi solid-state LiDAR data that can be applied to the retraining of the SPVNAS model.

2.4 Data Synthesis

2.4.1 Overview

As discussed earlier, we are not able to directly transform the KITTI spinning LiDAR dataset to quasi solid-state LiDAR data with the same labels because spinning LiDAR data is sparser than the quasi solid-state LiDAR data. We cannot go from a lower information state to a higher information state as we lack the information about the surface. There are also few publicly available quasi solid-state LiDAR datasets because the technology is new. In addition, it is not feasible to physically drive the vehicle around to collect large quantities of LiDAR data that is enough for training. Thus, the remaining options are collecting data in a simulated environment or transforming from datasets with an even higher information state. The approaches attempted are scene construction and stereo image segmentation. The goal is to synthesize large quantities of quasi solid-state LiDAR data with labels associated with it.

2.4.2 Scene Construction

The first approach for the synthesis of quasi solid-state LiDAR data is scene construction. It is expensive to drive the physical vehicle around in real world, but we can simulate vehicle movement in a human constructed scene at no cost. The tool I will be using are Unreal Engine and Simulink. Unreal Engine and Simulink are two powerful software tools that can be used together to create realistic simulations for a variety of applications.

Unreal Engine is a game engine that provides a comprehensive suite of tools for designing and developing interactive 3D environments, while Simulink is a graphical programming environment for modeling, simulating, and analyzing dynamic systems. By combining these two tools, it allows us to create complex, multi-domain simulations with high-fidelity graphics and physics. For example, Simulink can be used to model a physical system,

such as a vehicle, and Unreal Engine can be used to provide a realistic 3D environment in which to simulate the behavior of the system. This combination of tools enables us to quickly prototype, and test run the vehicle in a safe and cost-effective manner, without the need for physical prototypes. Unreal Engine allows me to place various objects in the scene, and Simulink allows me to run a vehicle simulation in the scene. For collecting quasi solid-state LiDAR data, I can control the vehicle to drive around the scene and do a 360-degree LiDAR scan of the surrounding every one second and record the point cloud data and the object labels at the same time.

This is an automated data collection process that could produce an abundance of data for training deep learning models. However, this also comes with some limitations. The resulting model pretrained with the quasi solid-state LiDAR data synthesized will be largely homogeneous on the types of objects provided in the scene. Although we can place all types of objects in the scene, if we consistently place only few versions of the same type of object, the model may learn some biased features from the homogeneous surfaces. For example, if Unreal Engine only has three types of trees that can be placed in the scene, and the model is trained with the constructed scene. When it comes to the real world, the model may not be able to identify other types of trees when the vehicle encounters them. The manual construction of the scene also takes a huge amount of labor in order to produce a large enough scene for simulation.

2.4.3 Stereo Image Segmentation

Stereo images are a pair of 2D images that are captured from slightly different perspectives, simulating the way our two eyes see the world. These images can be used to create a 3D representation of a scene, as the slight difference in perspective allows the brain to perceive depth and distance. Stereo images are used for their ability to create realistic 3D environments.

They can be captured using specialized cameras with multiple lenses, or by taking two separate images and aligning them in post-processing. Once the stereo images have been captured or created, they can be used to generate depth maps, which provide information about the distance between objects in the scene. Stereo images in combination of depth maps can represent a scene with dense information. Therefore, it is possible to transform stereo images and depth maps into quasi solid-state LiDAR format.

There are also publicly available stereo image datasets available in KITTI, as well as their corresponding depth maps. However, the problem is the lack of labeling information, which is the ground truth for 3D semantic segmentation. Since stereo images are two dimensional, and the depth map can be placed into the image by coloring the pixels according to relative depth. Therefore, 2D semantic segmentation needs to be conducted before building the 3D semantic segmentation model.

There are two types of 2D semantic segmentation models: pixel-wise models and bounding box models. Pixel-wise models classify each pixel as a specific object, while bounding box models draw boxes around each object that needs to be identified. For this task, we need pixel-wise models to provide us with the accurate coordinates of each object when it is transformed into 3D space in combination with the depth information. Then the transformed quasi solid-state LiDAR data and its labels can be fed into the 3D semantic segmentation model for training.

However, this approach also comes with some limitations. The 2D semantic segmentation models are not one hundred percent accurate, so the ground truth labels provided for the 3D semantic segmentation model may not be the real ground truth. Moreover, most 2D

semantic segmentation models are trained with normal images. Their performance may be reduced when the stereo images are used as the input.

2.4.4 Rich Point Subsampling

While most spinning LiDAR sensors can only generate sparse point clouds from the environment, there are advanced models such as CTICP [9] that can take a scene with sparse point and reconstruct the whole scene with high confidence. The Paris-Lille-3D [10] is three large and high-quality ground truth urban point cloud datasets for semantic segmentation, including Lille1, Lille2, and Paris. The raw dataset was acquired by a Velodyne HDL-32E LiDAR mounted at the rear of the truck, which was driven around Paris and Lille urban areas, and then the raw dataset went through scene reconstruction to produce a large scene with rich point clouds with derived labels. This dataset is dense enough so that we can subsample points from the scene to make the data format to match the features for the quasi solid-state LiDAR. It contains a total of 143.1 million points and 50 classes, which meets the requirements for the training dataset for the SPVNAS model.

There are also several challenges to work with this dataset. Different from the KITTI format where the point clouds and labels are organized into sequences of different scans, the Paris-Lille-3D dataset contains a huge collection of all the points with their labels in each file of the scenes. In order to partition the point into multiple scans, we need information about which scan each scan belongs to. Fortunately, the dataset provides us the x, y, and z origin, which are the coordinates of the position of the LiDAR. We can assume that points from the same scan will have the exact same LiDAR position. Therefore, the point clouds are classified into the same scan if the hash value of the x, y, and z origin evaluates the same.

Another major challenge is that the dataset we acquired is after scene reconstruction, including objects and surfaces that are out of sight for a typical quasi solid-state LiDAR sensor. As the vehicle moves along the road at its moving direction, a typical quasi solid-state LiDAR sensor can only collect points that are within ±45 degrees of the moving direction, and within a certain radius of about 40 meters. Therefore, during the data processing step, all the points that are invisible to the quasi solid-state LiDAR sensor should be removed from the dataset.

First, it is necessary to compute the moving direction as a set of rotation angles from the null frame (x-forward, y-right, z-upward). There are some trials initially with fixed angles with rotation angle of -50 degrees along y-axis, and for most of the points in the Lille1 dataset, the sensor points toward the moving direction, as shown in the visualization below.

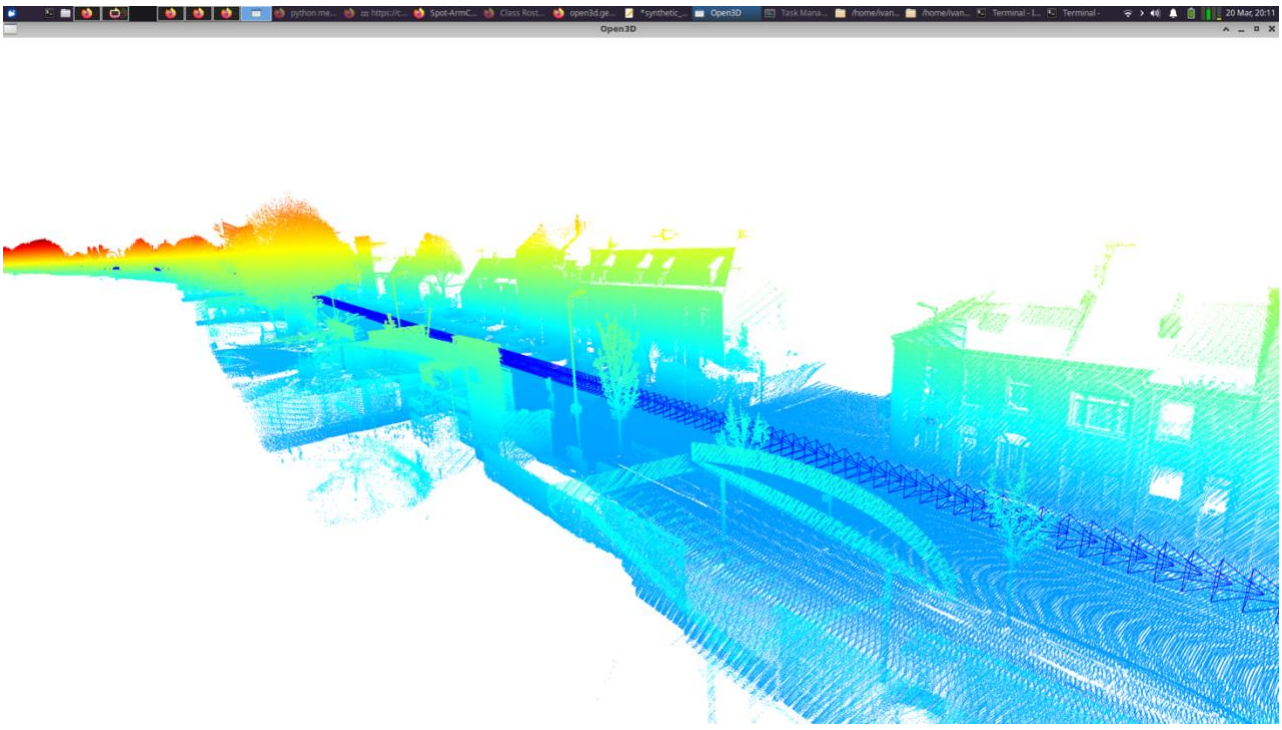

Figure 7: Lille1 Visualization with Correct Moving Direction Frames

The scene is composed of millions of points with color codes that associated with their labels. The blue frames represent the LiDAR position as the vehicle moves along the scene, and it points to the direction from the top vertex of the pentahedron to the bottom square. It can be observed from the above image that the moving direction for most frames are correct along the scene, but in other cases when the vehicle makes a turn, the frames do not match the moving directions for some segments of the road.

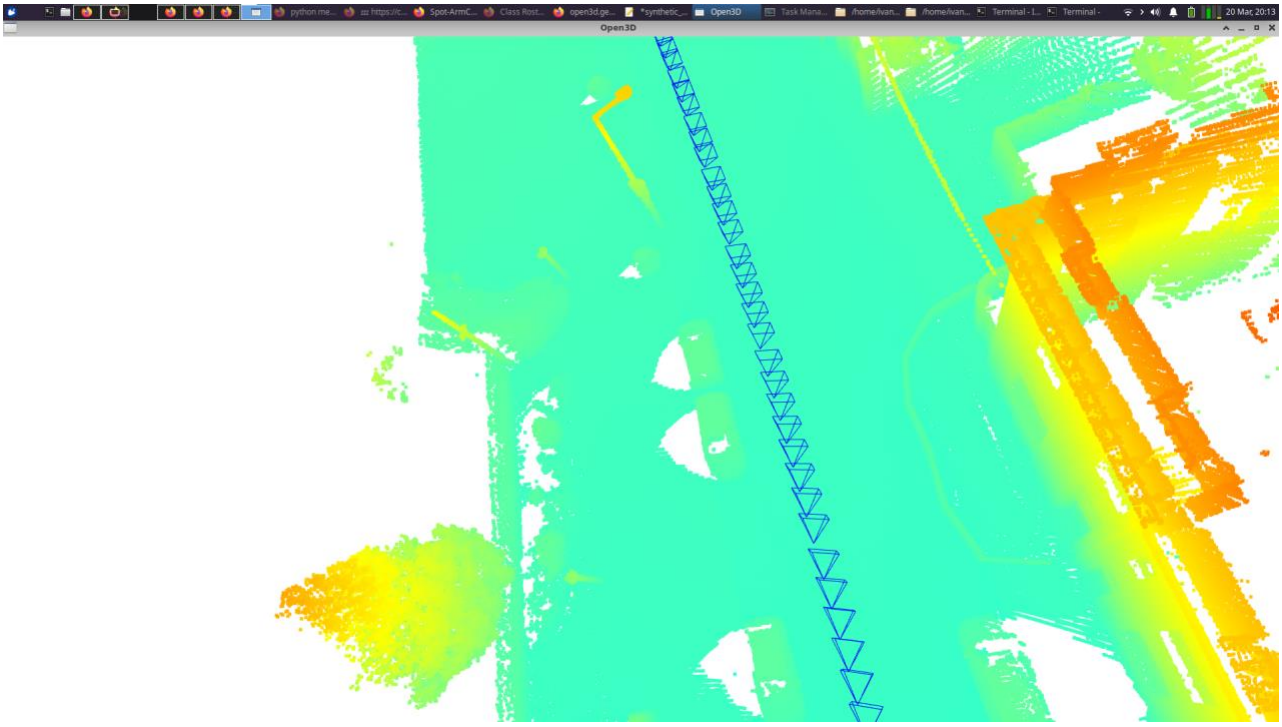

Figure 8: Lille1 Visualization with Correct Moving Direction Frames

This is an example showing that the frames are tilted to the right of the actual moving direction. In order to get an accurate representation of the quasi solid-state LiDAR dataset, the moving angle should be computed based on the position of the consecutive LiDAR sensors. The relative angle for moving directions is calculated by $\arctan\left(\frac{y_2-y_1}{x_2-x_1}\right)$, where x_2 and y_2 are the

position coordinates of the current LiDAR sensor, while x_1 and y_1 are the position coordinates of the previous LiDAR sensor.

After getting the accurate moving direction angles, the program will run through all the LiDAR positions and include the points that are within ±45 degrees of the computed moving direction and within 40 meters from the LiDAR position. Any points that are not covered after the whole iteration are removed from the dataset.

Another crucial idea that should be applied in data processing is hidden point removal. The basic idea is that we cannot see the surface that is behind another surface that is not transparent. Therefore, I applied an algorithm that approximates the visibility of a point cloud from a given view, which is the LiDAR position in this case, without surface reconstruction or normal estimation.

After data processing, the sampling algorithm can be applied according to base on a quasi solid-state LiDAR data pattern file which was created by studying previous quasi solid-state LiDAR datasets. It will take all the remaining points and take a sample of points around each frame as the vehicle moves along that matches the requirements. This is the data synthesis step.

In addition, the characteristics of the scenes that the dataset represent need to be studied before the training is run in practice. Since our goal is to make a well-performed pretrained model for the autonomous vehicle in the US, the elements that make up the scene of the dataset in France should match what we have in the US in majority. The points that make up the road barriers and the traffic lights in the dataset are filtered out, and the shape formed with these points look different from the road barriers and traffic lights we have seen in the US, so we have to remove these points from the dataset unfortunately.

I have been working on the three approaches to synthesize new dataset in parallel, and the rich point subsampling method is the most feasible in practice due to its limitations can be easily solved with data processing, and it requires the least manual labor compared to the other two methods. Thus, rich point subsampling will be the final method decided for synthesizing the training dataset for 3D semantic segmentation on quasi solid-state LiDAR data.

2.5 Training the model

With the rich point subsampling approach, the data processing step removes about 43.42% of points that are invisible in the reconstructed scene, and we are left with about 120 million points in the resulting three scenes for data synthesis. The synthesis step samples all the points into 2874 separate scans each with around 40000 points. Then, the point cloud data and their labels are written into .bin and .label files that resembles the format for the KITTI dataset, and they are split into training, validation, and testing sets with a ratio of 70% to 10% to 20%.

The model that is built from scratch is the SPVCNN model, which is a rudimentary version of the SPVNAS model. It has four convolutional layers, each followed by two residual layers, and then four deconvolutional layers, each followed by two residual layers as well. Initially all the weights are initialized to 1, and the bias is initialized to 0. In each forward propagation step, the point clouds are transformed into a three-dimensional image with a voxel size of 0.05, so that the convolution neural network principles can be applied to it. The optimizer for backward propagation is Stochastic Gradient Descent with a learning rate of 0.24, weight decay of 0.0001, and momentum of 0.9. This ensures that it can process dataset of huge sizes quickly without losing much about the accuracy.

The training of the model is GPU accelerated with eight concurrent worker threads. The original dataset has 101 class labels, but most of them are not useful for making driving

decisions, such as bollard, waste, or trash can. 36 crucial elements including pedestrians, cars, road, sidewalks, etc. are filtered out for the classification task, and the rest of the labels are put under unclassified, which is ignored by the model at the end.

During each epoch of training, all the point cloud and labels files in the training set are fed into the network for updating the weights and improving the model, then it will evaluate itself on the validation set to calculate the IoU metric and the loss. From epoch to epoch, if the model is improving, the IoU should increase, and the loss should decrease. As the model finish training, it will be saved as a checkpoint file which is passed into the evaluation step in which the model will be evaluated against the testing set and new IoU metrics will be calculated, and visualizations will be created.

3. RESULTS

3.1 Metric values

The resulting SPVCNN model has an IoU of 28.912% and a loss of 1.8346 in epoch 1 against the validation set, and an IoU of 30.803% and a loss of 1.3103 in the final epoch. As we can see that the model does improve as it is training, but not very much. This is expected as most classification neural networks start at very low metric values initially and they gradually grow as the models are trained over and over. This indicates that the SPVCNN models need more epochs and possibly more data for it to boost its performance. However, training the model for 15 epochs takes about 6 hours on a remote server, and it reaches the maximum usage for the server's GPU.

The testing set has an IoU of 29.775% and a loss of 1.5284, which is also expected since the testing metrics are most of the time slightly worse than the training set, but it should not diverge largely since the dataset should be homogeneous between training and testing sets.

3.2 Visualization

As we can observed from the visualization that most of the points in this scan is classified as one type showing dark blue in the image, which represents the label 'road'. By inspecting the synthesized dataset, around 20% to 40% of the points have a label of 'road', so the model is heavily biased toward the most prevalent label and classify everything as road to achieve an IoU of around 30% because it is stuck in the state of realizing 30% cannot be improved. While behind the scenes, the weight of the neural network is trapped into a local minimum loss during

gradient descent, which is not the global minimum loss.

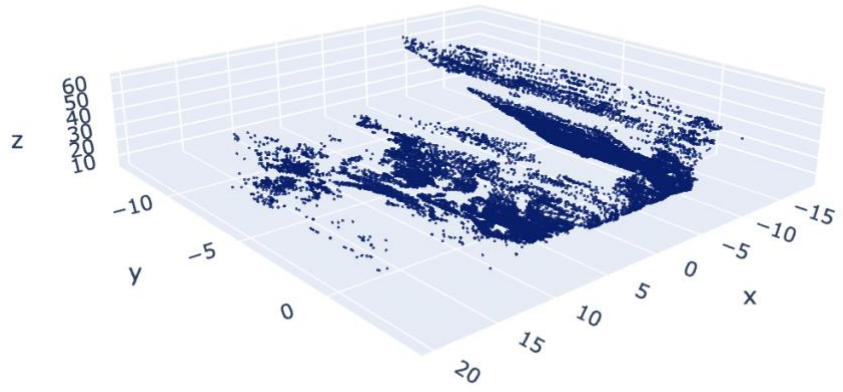

Figure 9: Model Prediction for scan 1962

This data pattern is understandable since during the data processing step, only points that are within the vehicle's field of view can be included in the synthesized dataset. Most of the time the points that can be easily seen are the road because they are dense along the center of the moving direction, while only small portion of the objects on both sides of the road are included as various labels that do not predominantly make up the dataset.

This issue is one of the drawbacks for the model as it will lean heavily toward the most prominent label initially. However, this situation can improve if the model is trained for hundreds of epochs, which requires substantial hardware power and time.

4. CONCLUSION

4.1 Approach Feasibility

In order to train a 3D semantic segmentation model with quasi solid-state data with high performance, the two key components are the data and the model. The synthesized dataset matches the format of quasi solid-state LiDAR data, and it is also abundant enough for the deep learning model to have meaningful prediction results. For the SPVNAS model I utilized, it has a testing IoU of 60.7% for the SemanticKITTI dataset, so it is well suited for the 3D semantic segmentation task. Since quasi solid-state LiDAR data is denser than the spinning LiDAR data, it contains more information for the surface. The model should be also well suited for the synthesized quasi solid-state LiDAR data. Therefore, the approach is generally feasible, but there are still several challenges that need to be overcome until optimal results can be achieved. The first challenge is the hardware that meets the requirement, and the other is the tuning of the parameters of the model.

4.2 Future steps

Currently the model is built and run on a remote server that has a daily limit on the usage of GPU, and it is also hard to keep it activated all the time since the program will halt when the computer hibernates, and all the training progress is lost. It takes about six hours to complete one training session, which is a huge dedication and requires large attention. This limits the scale of training, so it is hard to get meaningful results from the resulting model.

The best solution is to run the model locally on a computer with high performance GPUs or on a privately owned remote server that is also suited with high performance GPUs which can be run overnight. I demonstrated my results to my faculty advisors, and they allowed me to train

my model on a school computer that has the hardware requirement. I will continue to train my model in a better environment to get more meaningful results for 3D semantic segmentation.

Currently the parameters of the model that are wired including the voxel size and the optimizer are best suited for the spinning LiDAR datasets. There is still some room for testing different values of each parameter and examine the performance of the resulting model. This trial-and-error step is time consuming, but it leaves some room for potential improvement of the model performance.

REFERENCES

[1] K. Akadas and S. Gangisetty, “3D semantic segmentation for large-scale scene understanding,” *Computer Vision – ACCV 2020 Workshops*, pp. 87–102, 2021.

[2] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 12, pp. 4338-4364, 1 Dec. 2021. https://doi.org/10.1109/TPAMI.2020.3005434.

[3] Wang, Dingkang, Connor Watkins, and Huikai Xie. 2020. "MEMS Mirrors for LiDAR: A Review" Micromachines 11, no. 5: 456. https://doi.org/10.3390/mi11050456

[4] Montalban, Karl, Christophe Reymann, Dinesh Atchuthan, Paul-Edouard Dupouy, Nicolas Riviere, and Simon Lacroix. 2021. "A Quantitative Analysis of Point Clouds from Automotive Lidars Exposed to Artificial Rain and Fog" Atmosphere 12, no. 6: 738. https://doi.org/10.3390/atmos12060738

[5] “Semantickitti,” SemanticKITTI - A Dataset for LiDAR-based Semantic Scene Understanding. [Online]. Available: http://www.semantic-kitti.org/. [Accessed: 07-Apr-2023].

[6] X. Yan, J. Gao, C. Zheng, C. Zheng, R. Zhang, S. Cui, and Z. Li, “2DPASS: 2d priors assisted semantic segmentation on lidar point clouds,” *arXiv.org*, 14-Oct-2022. [Online]. Available: https://arxiv.org/abs/2207.04397v3. [Accessed: 07-Apr-2023].

[7] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, “Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation,” arXiv.org, 2020. [Online]. Available: https://arxiv.org/pdf/2011.10033v1.pdf. [Accessed: 08-Apr-2023].

[8] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, “Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution,” *arXiv.org*, 2020. [Online]. Available: https://arxiv.org/pdf/2007.16100v2.pdf. [Accessed: 08-Apr-2023].

[9] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, “CT-ICP: Real-time Elastic Lidar odometry with loop closure,” arXiv.org, 24-Feb-2022. [Online]. Available: https://arxiv.org/abs/2109.12979. [Accessed: 07-Apr-2023].

[10] X. Roynard, J.-E. Deschaud, and F. Goulette, “Paris-lille-3d: A large and high-quality ground truth urban point cloud dataset for automatic segmentation and Classification,” *arXiv.org*, 10-Apr-2018. [Online]. Available: https://arxiv.org/abs/1712.00032. [Accessed: 07-Apr-2023].