# CONVENIENT USER INTERFACE FOR LABELLING 3D POINT CLOUD

**TAN SHI HAO**

**A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering**

**Lee Kong Chian Faculty of Engineering and Science**
**Universiti Tunku Abdul Rahman**

**APRIL 2021**

# DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : TAN SHI HAO

ID No. : 16UEB04655

Date : 6 MAY 2021

# APPROVAL FOR SUBMISSION

I certify that this project report entitled **"CONVENIENT USER INTERFACE FOR LABELLING 3D POINT CLOUD"** was prepared by **TAN SHI HAO** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr Ng Oon-Ee

Date : 6 May 2021

Signature :

Co-Supervisor :

Date :

# ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Ng Oon-Ee for his invaluable advice, guidance and his enormous patience throughout the development of the research.

# ABSTRACT

Nowadays, most labelling tools are able to label 2D images well. However, the same cannot be said for 3D point clouds. Thus, the 3D point cloud labelling tools are pretty expensive in the marketplace. This report reviews different types of 3D point cloud labelling tools, including automatic and manual 3D point cloud labelling tools. This project mainly focuses on manual open-source user interfaces for 3D point cloud labelling. A comparison between the open-source user interfaces was conducted. Objective evaluation of usability and labelling speed between the user interfaces was carried out. The user interfaces used for comparison are POINTS, 3D BAT, LATTE. Three of the user interfaces are screen-based and web-based tools, and the comparison is made using the same point cloud scene, taken from KITTI datasets. The usability of the user interfaces was evaluated in terms of instruction and shortcuts, projective views, editable in projective views, fast annotation feature, timer, and main view focus mode. The labelling speed of the user interfaces is evaluated in terms of time and errors. After the comparison, POINTS is the user interface with many useful features, which aids the users to label easily and faster. The labelling time of POINTS is the shortest among three user interfaces. POINTS can perform well as or better than the 3D point cloud labelling tool in the marketplace.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS / ABBREVIATIONS

| | |
|---|---|
| % | percentage |
| 3D | three dimensional |
| 2D | two dimensional |
| 3D BAT | 3D Bounding Box Annotation Toolbox |
| AI | artificial intelligent |
| API | application programming interface |
| AR | augmented reality |
| BIN | binary format |
| CNN | convolutional neural network |
| CSS | Cascading Style Sheets |
| CPU | central processing unit |
| DBSCAN | Density-based spatial clustering of applications with noise |
| DoF | degree of freedom |
| DNN | deep neural network |
| FP | false positive |
| FN | false negative |
| GPU | graphics processing unit |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IDE | integrated development environment |
| IoU | intersection over union |
| min | minutes |
| LATTE | open-sourced annotation tool for LiDAR point clouds via sensor fusion, one-click annotation and tracking |
| LiDAR | light detection and ranging |
| PCD | Point Cloud Data |
| POINTS | Portable Point-cloud Interactive Annotation Platform System |
| PointNet | deep learning on point sets for 3D classification and segmentation |
| RGB | red, green, blue |
| RGB-D | red, green, blue and depth information |

| | |
|---|---|
| S3DIS | Stanford Large-Scale 3D Indoor Spaces |
| s | seconds |
| TP | true positive |
| UI | user interface |
| VGG16 | Visual Geometry Group from Oxford |
| VR | virtual reality |
| WebGL | Web Graphics Library |

# LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 General Introduction

In this new era of technology, the development of the automation sector is growing rapidly due to the revolution of Industry 4.0. Some automation and robotic systems require 3D data to complete the jobs effectively because 2D images lack detailed information such as depth and positioning information (Bello et al., 2020). 3D data provides more information for the machines to adapt themselves to the surrounding environment. Various applications use 3D data, such as virtual reality, robotics, remote sensing, autonomous driving, and so on (Guo et al., 2019). By comparing the regular camera and 3D scanner, the regular camera can capture the 3D world in 2D images, whereas 3D scanners, LiDAR, and RGB-D camera are handy in extracting the 3D pieces of information or 3D model (Rehman et al., 2019). The 3D point cloud is one type of format to represent 3D data. A 3D point cloud comprises a set of points in the cartesian coordinates of the X, Y, and Z-axis. The 3D point cloud representation can store the original geometric information in 3D space. There are two types of sensors, which are a LiDAR sensor and a 3D laser sensor (National Ocean Service, 2020). LiDAR sensor or 3D laser sensor is a remote sensing device used to collect data and information about the shape and feature of the earth with aircraft or satellites' aid. The acquisition of the 3D point clouds would be more convenient with the aid of sensing devices. LiDAR sensors or 3D laser sensors can also generate 3D point clouds.

Deep learning is an AI function that the machine can learn without human's supervision. The deep learning technique is widely used in various applications, including autonomous driving, computer vision, and robotics (Guo et al., 2019). Nowadays, the development of deep learning on 3D point clouds is rising as well. However, point clouds are generally unstructured, which induces problems and makes it challenging to apply deep learning on 3D point clouds. 3D point cloud classification, detection, and segmentation had addressed the problems faced by deep learning of 3D point clouds. 3D point cloud segmentation techniques are pretty helpful for 3D labelling tools.

Some 3D labelling tools need to undergo a segmentation process before performing 3D point cloud labelling.

3D point cloud labelling is a method to redefine the points with labels, which helps with performing computer vision and object recognition. With the labelled object in a 3D point cloud, it can be used in applications, including robotics, self-driving car, virtual reality, augmented reality, urban planning, and emergency disaster control plan. This is because the objects in the 3D point cloud can be recognized. In this study, the main concern is comparing the open-source user interfaces that can perform 3D point cloud labelling. By trying out the open-source 3D point cloud user interface, the user interface enables users to view the 3D point cloud and label the points when detecting the object. Most of the labelling user interfaces are using a cuboid label to label the object within a specific region. Minority of labelling tools are using point-wise labelling. The user interfaces are designed in various forms, such as point-and-click, AR-based, VR-based, etc. The most important thing about the user interface is that the operative difficulty must be moderate, which is easy for the user to operate. With the simple operation, the accuracy of the label should be higher. The accuracy of labels is normally evaluated in terms of IoU, false-positive ratio and false-negative ratio of points in the bounding box.

## 1.2 Importance of the Study

This study may contribute to a better understanding of labelling point clouds in 3D space. 3D point cloud labelling is assigning labels to points in 3D space to ease object recognition. Labelling in 3D space is better than labelling in 2D images because 3D point clouds consist of more detailed information than 2D representations. The scenes or objects represented by 2D images lack depth and positioning information, whereas 3D point clouds can provide the depth information and preserve the specific objects' original position with minimized discretization. 3D point cloud labelling can be effectively used in applications involving computer vision or object recognition, such as autonomous driving, robotics, urban planning, virtual reality, and so on. With the aid of a 3D point cloud labelling system, the specific applications' operation will be efficient and smooth.

Implementation of 3D point cloud labelling in the automation and robotics sectors can improve efficiency and effectiveness. For example, applying a 3D labelling system on a robot enables the robot to recognize the obstacles and avoid the obstacles, which blocks a robot's movement. Besides, 3D point cloud labelling can contribute to urban planning and emergency disaster control plan. The 3D point cloud of a city can be generated using a 3D scanner such as a LiDAR sensor or 3D laser sensor (Babahajiani et al., 2017). All objects, regardless of small or large objects, including buildings, vehicles, fire hydrants, and other objects in the city, are recognized and labelled. For urban planning, the authority may observe and analyze the labelled 3D point cloud to decide the suitable areas for the city's development. For the emergency disaster control plan, all the emergency items in the city are labelled. The labels will ease checking whether the amount of emergency items in the particular area is sufficient or not and analyzing the level of safety. By implementing a 3D point cloud labelling system, the work will be done more efficiently because the planning can be done without going to the particular site to observe the conditions. This system will save a lot of time and workforce.

## 1.3 Problem Statement

Machine learning nowadays is able to handle 2D data well. However, the same cannot be said for 3D data. This is because the point clouds in 3D space contain complex and detailed information as well as require a large number of points, up to millions or even billions of points. By comparing with 2D images, the number of pixels needed for machine learning is significantly less than the points needed for 3D data. Most labelling tools are able to label 2D images instead of 3D point clouds. Besides, most of the labelling tools use the 2D representation technique to label the 3D point clouds. However, 2D data lacks depth information, which will greatly affect the quality of labelling and cause the problem of mislabelling.

Some researchers had worked on completing the 3D point cloud labelling directly on 3D representation without transferring the 3D data into 2D representation. As an example, Huang and You (2016) had proposed a method to overcome the problem faced by using 2D representation.

Voxelization enables the 3D labelling process to be done in 3D representation. The 3D labelling tools can be classified into automatic and manual labelling. The 3D point cloud labelling tools must be convenient and easy to use for the users. There are some famous 3D point cloud labelling tools, including Supervise.ly, Scale.ai, Playment.io, Pointly.ai, Amazon Sagemaker Ground Truth, which are convenient to use and consisted of many useful features. However, the user interfaces are not open source and quite expensive to purchase since the payment must be paid monthly or yearly. Thus, the literature reviews on open-source 3D labelling tools will be conducted to evaluate the performance of the open-source 3D labelling tools, which can be easy to use and free of charge.

## 1.4 Aim and Objectives

The user interface should be able to label directly on 3D point clouds instead of 2D images to overcome the stated problem in section 1.3. Besides, user interfaces for labelling 3D point clouds with good features at the marketplace are pretty expensive. Thus, the literature reviews on open-source user interfaces for labelling 3D point clouds should be conducted. More specifically, the aim of this study is to research convenient and free user interfaces for labelling 3D point clouds, which can handle 3D data well and allow users to traverse in a 3D environment and label the object in point clouds. The specific objectives of this project, including:

1. Comparison of user interfaces for 3D point cloud labelling.
2. Evaluation of usability between the user interfaces.
3. Evaluation of labelling speed between the user interfaces.

The comparison between 3D point cloud labelling user interfaces will be done by setting up several user interfaces. The results will be done using the same point cloud scene. The performance of the user interface will be evaluated in terms of usability, where the user interface should be convenient and easy to use. The evaluation of labelling speed in terms of time will be conducted between the user interfaces. The user interfaces that are convenient and easy to use, and the labelling speed is fast will be considered as the most efficient user interface.

## 1.5 Scope and Limitation of the Study

The scope of this study is mainly focused on 3D point cloud labelling. All the research papers that had been studied are about the labelling on the 3D point cloud. The general purpose of this study is to research convenient and free user interfaces for labelling 3D point clouds and make comparison between the user interfaces. The user interface allows users to label the objects in a 3D point cloud in order to ease object recognition and computer vision. The 3D object detection can be done on 3D point clouds effectively with the labelled point clouds. However, 3D object detection will not be covered in this study. 3D object detection is the process after the labelling is finished. The duration of this study is about one year period, and the final report for this project was finished in January semester 2021.

## 1.6 Contribution of the Study

This project reviews the existing open-source user interfaces for labelling 3D point cloud and provides insights into the user interfaces. Some open-source user interfaces for 3D point clouds are set up. A convenient user interface is adopted and modified by adding some useful features. The comparison is conducted between open-source user interfaces. Evaluation of the user interfaces' performance is carried out in terms of usability and labelling speed.

## 1.7 Outline of the Report

In this project, the study is focused on 3D point cloud labelling. The main aim and objective are to compare the user interfaces for labelling 3D point clouds and evaluate user interfaces' performance. Before starting the software development, paper research should be done to master this field's knowledge as much as possible. Besides, the advantages and limitations of the various labelling tools will be known, and comparison is able to carry out among the labelling tools.

In this project, five chapters have been discussed in this report, namely introduction, literature review, methodology and work plans, results and discussion, and conclusions and recommendations. Firstly, in chapter 1, there is a general introduction about the 3D point cloud and labelling system in 3D space. Besides, the importance of the 3D point cloud labelling and the problem

encountered had been stated. The subchapters are aim and objectives, scope and limitation, and an outline of the report. Chapter 2 contains the literature review, which mainly focuses on 3D point cloud labelling and discusses the methods in designing 3D point cloud labelling tools approached by other researchers. The literature review will form the basis for decision-making for determining the convenient user interfaces for labelling 3D point clouds with high-quality label. Chapter 3 describes the method for comparison of open-source user interfaces in this project and the schedules of part I and part II projects. Besides, the problems encountered during methodology will be discussed. Chapter 4 describes the results and discussion about the comparison between user interfaces for 3D point cloud labelling. Lastly, Chapter 5 summarizes the overall project and makes recommendations to further improve the user interfaces.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

3D point clouds are datasets where the combination of points in the cartesian coordinates of the X, Y, and Z-axis is used to represent space or object. Basically, the 3D point cloud consists of many points, even up to billions of points. Nowadays, point clouds play an important role in representing 3D data (Bello et al., 2020).

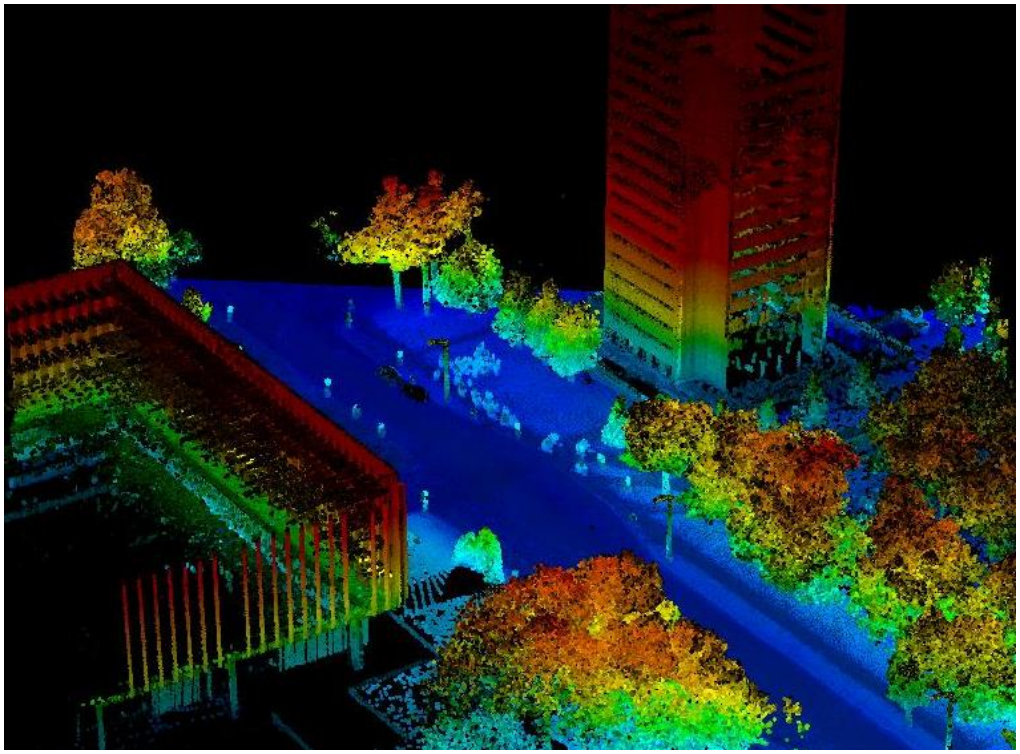

Figure 2.1: 3D LiDAR point cloud (Daniel, n.d.).

Photogrammetry and remote sensing methods work on similar principles for creating a 3D point cloud. The way of photogrammetry used to build the point clouds is by merging multiple photographs with multiple angles, where the photographs are used to investigate and measure the particular area and object. The data of the Earth can be obtained by using an airboat or satellites. By installing LiDAR sensors or 3D laser sensors on aerial vehicles, the data about the shapes and features of the Earth can be detected and

recorded. This kind of method to collect data on the Earth is known as remote sensing (FME Community, 2020). 3D laser scanners and LiDAR sensors can be used to generate the 3D point clouds. Laser light is used to measure the distances of an object to the surface of the Earth. The information about the shape and feature of the Earth obtained is more accurate and exact by the combination of light pulses and data collected from the airborne system (National Ocean Service, 2020). There are some processes that can be used to transfer raw point clouds to become structured point clouds. In this report, the main concern is data annotation, where the labels are given to the points of objects in a point cloud.

## 2.2 Usefulness of Point Cloud Labelling

Point cloud labelling is a method to label the points, which the redefined points can be used to represent the scene or object. Object recognition and computer vision can be done by point cloud labelling. However, the unavailability of 3D point cloud labels is the main problem encountered, which will affect the effectiveness of the classifiers (Wang, Ji, and Chang, 2013). Besides, the ability of the 3D point clouds labelling algorithm in recognizing smaller objects is still limited. The result of labelling smaller objects is not accurate and precise.

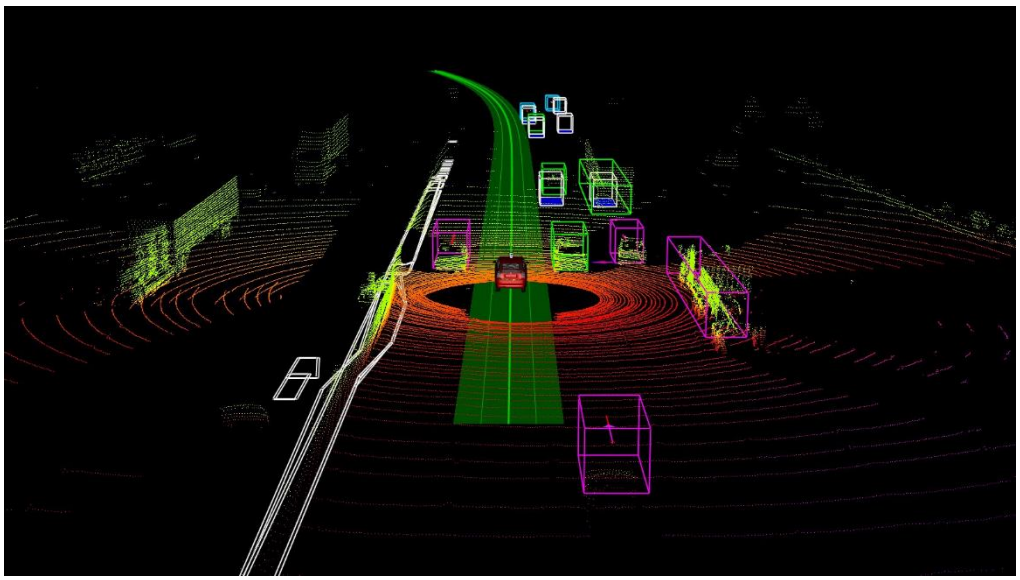

Figure 2.2: The LiDAR point cloud with cuboid labels (Koh, 2018).

Point cloud labelling can be used in many applications. If the 3D point cloud model of a city is labelled, this will benefit some applications such as reinforced reality maps, urban planning, disaster management as well as virtual tourism (Babahajiani et al., 2017). With the labelled city map, the safety of using a self-driving car will improve because the labels can help in providing a guide for the navigation of self-driving cars. In order to enhance the safety system in the city, small objects, such as fire escapes, fire hydrants, and other emergency items are labelled can effectively help in disaster management and emergency plans. Small objects like a fire hydrant, cars, pedestrians, and so on are important in improving urban planning. Apart from that, in order to move in a smooth motion, most of the robots are depending on the sensors mounted on the robot (Kim and Sukhatme, 2014). However, sometimes there are too many obstacles blocking the way of the robot and cause the robot stuck in a particular area. Therefore, by applying a semantic labelling system to the robot, the robot can recognize the objects. Then, the robot can decide to move the objects within the capability of the robot to clear the path. Therefore, the 3D point cloud labelling system is quite useful for robotics.

There are two types of tools, including automatic 3D point cloud labelling algorithm and manual 3D point cloud labelling tool, to complete the 3D point cloud labelling process. For automatic 3D point cloud labelling algorithms, voxel-based 3D CNN, Patch Context Analysis and Multiscale Processing, as well as SnapNet, will be discussed, whereas for manual labelling tools, PointAtMe, 3D BAT, LATTE, and POINTS will be reviewed.

## 2.3 Automatic 3D Point Cloud Labelling

Automatic 3D point cloud labelling is able to finish the labelling process without a user manual label. By applying the labelling algorithm to the original point cloud without labels, the algorithm will run and produce the final point cloud with labels as a result.

### 2.3.1 Voxel-based 3D CNN

According to the research, Huang and You (2016) had used the 3D CNN combine with voxelization to overcome the problems encountered by point

cloud labelling. In general, most of the tasks involving the detection and classification process should undergo the segmentation process. However, segmentation is not required for the voxel-based 3D CNN method because it depends on data that undergo voxelization. Voxelization is a method used to represent 3D point clouds directly. There are some challenges faced by voxelization. Voxelization requires large memory in a computer to operate, and it takes a long time to finish. This is because the algorithm lacks proper optimization.

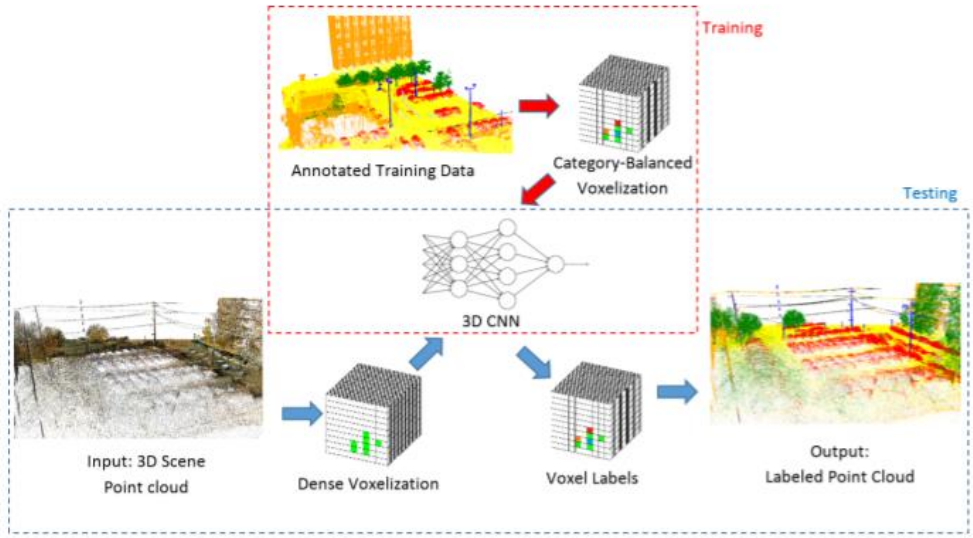


Figure 2.3: The pipeline of the labelling system with two different types of modules (Huang and You, 2016).

The labelling system consisted of two training modules. Firstly, for the online testing module, raw point clouds had been taken as the input. Then, the raw point clouds will go through the dense voxelization process. This process had generated an occupancy voxel grid with the centered voxels. The resulting voxels of the dense voxelization process are served as the input for trained 3D CNN. Each of the voxels will be given a label. Then, the labels were mapped to the original point cloud. Eventually, the labelled point cloud will be obtained, which was shown in Figure 2.3. Besides, for the offline training module, the annotated training data was being taken as the input. Then, the annotated training data went through the voxelization, and an occupancy voxel grids was produced. The resulting voxels are parsed through 3D CNN, and the final output is produced.

Huang and You (2016) had selected the LiDAR point cloud in Ottawa, which is the point clouds of urban area as the dataset for this labelling system. The data were collected by using four car-mounted scanners as well as an airborne scanner. The library used to launch the 3D CNN is Theano library, and the method used to train the network is Stochastic Gradient Descent. Many features and labels are generated, even up to 500000, when the trunks from the data are manually labelled. A total of 50000 and 20000 features and labels are distributed randomly as training data and validation data, respectively. The obtained results were used to compare with the ground truth labels. The accuracy for labelling cars and planes was about 95 %, whereas labelling buildings, wires, and poles are in a range between 80 % to 90 %. Besides, the accuracy for trees is 78 %, which is slightly lower compared to other objects. This is because the presence of scattered clusters like humans and bushes will affect the labelling's performance. In the end, the overall accuracy for point labelling of all categories is 93 %.

This labelling system depends on the 3D CNN and voxelization processed to finish the labelling process without going through segmentation. This method is more suitable for small-scale 3D model analysis. This is because extra dimensions will cause the restriction of voxel resolution and shape precision. Besides, the performance speed is slightly slower because the voxelization requires some time to process. Therefore, more effort can be put into improving the system to get a more accurate and precise result when handling large-scale 3D model analysis and improving the labelling system's speed.

### 2.3.2 Patch Context Analysis and Multiscale Processing

In this research, Hu, Cai, and Lai (2020) had designed an algorithm for semantic segmentation and labelling of 3D point clouds. The combination of Patch Context Analysis and Multiscale Processing can achieve semantic segmentation and labelling. In this system, 3D point clouds are used as the input. The labelling and segmentation system's objectives are assigning each of the points a label for semantic labelling and segmenting the point clouds into a meaningful segmented object for semantic segmentation.

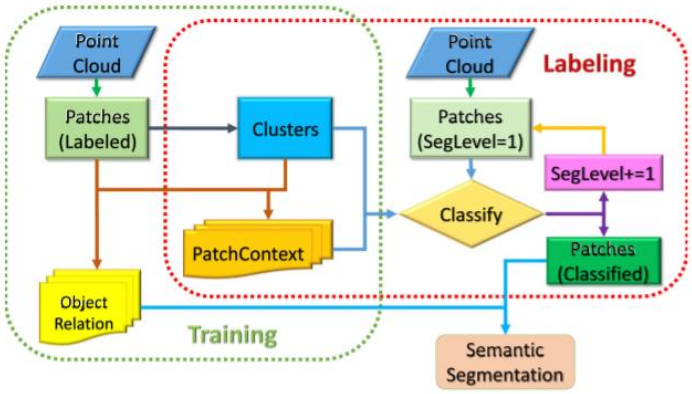


Figure 2.4: The pipeline for the labelling system (Hu, Cai and Lai, 2020).

According to Figure 2.4, the point clouds are segmented into patches with the object labels for the first step. Then, the patches that have similar characteristics are gathered into a particular cluster. This is because the contextual rules can be described more vigorously with the aid of cluster. Patch context information can be learned from the training datasets. The next step is classification, which can be performed as the contextual rules and patch context information had been learned at different segmentation levels. According to the semantic labels and the context, multiscale processing can be carried out to select and control suitable segmentation levels for local regions. The descriptions above are the overall flow of this labelling and segmentation system.

| F-measure | floor | wall | door | board | chair | desk | key-board | monitor | mouse | cup | CPU | window | desk cabinet |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Level_A$ | 98.76 | 88.05 | 00.00 | 00.00 | 77.23 | 86.13 | 00.00 | 82.55 | 00.00 | 83.13 | 00.00 | 72.59 | 60.04 |
| $Level_A + PC$ | 98.75 | 89.19 | 99.68 | 00.69 | 92.62 | 86.70 | 00.38 | 89.70 | 00.00 | 78.08 | 78.66 | 82.11 | 72.27 |
| $Level_B$ | 97.16 | 85.16 | 41.31 | 68.87 | 72.08 | 73.08 | 00.00 | 23.80 | 00.00 | 00.00 | 00.00 | 38.81 | 71.76 |
| $Level_B + PC$ | 98.05 | 92.76 | 96.91 | 94.20 | 86.33 | 76.61 | 35.09 | 84.72 | 00.00 | 51.29 | 37.02 | 62.18 | 71.98 |
| Koppula [38] | 78.90 | 80.61 | 00.00 | 00.28 | 68.27 | 79.64 | 61.98 | 86.89 | 50.03 | 89.38 | 36.48 | 00.00 | 00.00 |
| *MultiScale* | 99.48 | 96.08 | 00.00 | 98.48 | 71.27 | 93.76 | 00.00 | 82.97 | 00.00 | 77.41 | 00.00 | 77.40 | 71.31 |
| $MultiScale + PC$ | 99.59 | 97.56 | 99.87 | 98.55 | 92.51 | 96.52 | 91.63 | 87.83 | 63.93 | 70.04 | 53.43 | 82.51 | 75.18 |
| F-measure | basket | trash | phone | vase | eraser | radiator | book cabinet | pipe | plant | paper | clothes rack | clothes | Overall |
| $Level_A$ | 70.37 | 79.76 | 00.00 | 00.00 | 00.00 | 60.02 | 44.47 | 48.50 | 53.52 | 00.00 | 22.88 | 00.00 | 79.69 |
| $Level_A + PC$ | 84.88 | 83.48 | 36.43 | 39.02 | 00.00 | 80.63 | 71.09 | 51.83 | 56.06 | 26.13 | 40.90 | 41.34 | 85.12 |
| $Level_B$ | 53.20 | 45.97 | 00.00 | 00.00 | 00.00 | 09.81 | 28.53 | 00.00 | 00.00 | 00.00 | 00.00 | 00.00 | 78.19 |
| $Level_B + PC$ | 72.76 | 71.44 | 01.37 | 32.43 | 11.09 | 57.65 | 54.63 | 04.67 | 03.04 | 10.68 | 09.84 | 47.29 | 86.73 |
| Koppula [38] | 00.04 | 00.00 | 00.00 | 00.00 | 00.00 | 00.00 | 00.00 | 00.00 | 03.40 | 00.00 | 00.00 | 00.00 | 66.25 |
| *Multiscale* | 72.30 | 80.88 | 00.00 | 00.00 | 48.77 | 59.95 | 52.29 | 45.50 | 53.58 | 07.32 | 16.89 | 00.00 | 86.98 |
| $Multiscale + PC$ | 87.09 | 84.55 | 82.83 | 15.14 | 70.34 | 78.16 | 79.55 | 49.11 | 66.12 | 51.20 | 62.81 | 40.09 | 93.81 |

Figure 2.5: The labelling performances on the ETH dataset with 25 classes (Hu, Cai and Lai, 2020).

This approach is compared to the labelling system developed by Koppula and other researchers based on the datasets above. The patch-based segmentation and labelling are treated as two different problems. The patch-

based segmentation is completed by using a standard region growing technique. Hu, Cai, and Lai (2020) had chosen the ETH database as the datasets to evaluate the labelling system's performance. ETH database is a dataset consist of 18 office point cloud scenes. Twelve of the scenes are used as training data, and the remaining scenes are used for testing. The combination of Patch Context Analysis and Multiscale Processing showed the highest accuracy rate with 93.81 %, according to Figure 2.5. The Cornell RGB-D dataset is the second dataset, which consists of 59 scenes, including 25 office scenes and 34 home scenes. There are nine classes and ten classes for the office scenes and home scenes, respectively. Macro precision and recall are referred to the average precision of every class, whereas micro-precision and recall are referred to the percentage of correct labelled patches. By comparing the result to Koppula, this approach's performance is better than Koppula with a 3.87 % of overall improvement and an increase of 5.54 % in terms of micro-precision or recall. Next, SceneNN is a labelled point cloud dataset consists of 95 scenes by fusing the RGB-D images. There are 62 scenes used as training data, and 33 scenes are used for testing. There are a total of 15 classes to be labelled.

The combination of Patch Context Analysis and Multiscale Processing has the better performance compare to Multiscale Processing and Koppula. Koppula faces failure in the training stage. The overall accuracy of the combination of Patch Context Analysis and Multiscale Processing had been improved by 5 %. Last but not least, S3DIS dataset consists of 6 area scenes, and Matterport cameras had taken 271 room scenes with 13 classes of objects. This approach is used to compare with deep learning-based methods, PointNet and Engelmann. The mean IoU is 64.6 %, and the overall accuracy and average class accuracy are 88 % and 75.6 %, respectively. All of the three results of this approach are the highest compare to PointNet and Englemann.

### 2.3.3 SnapNet

In this research, Boulch et al. (2017) used 2D deep segmentation network to label the unstructured point cloud. SnapNet is using deep CNN to analyse and interpret multiple 2D image views. This method is slightly similar to the multi-view strategy, but the strategy in choosing views is different. The purpose of

this approach is to labelling the images instead of classifying the images. The way to select the views is taking a lot of partial views instead of taking a whole view of a scene or an object.

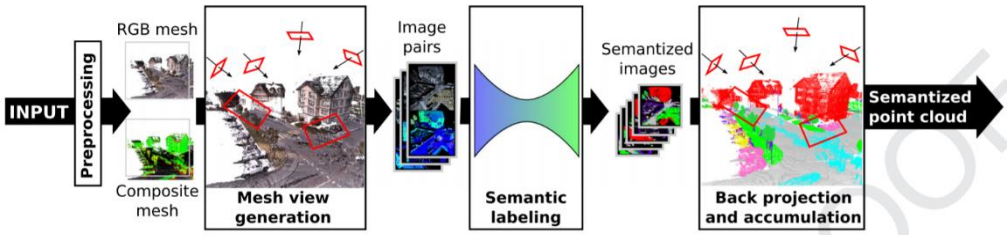


Figure 2.6: The pipeline of SnapNet by using 2D deep segmentation network (Boulch, et al., 2017).

There are four steps to complete the 3D point cloud labelling. Figure 2.6 shows the flow of this approach. The final result can be obtained by going through the four steps:

- Preparing the point cloud
- Generating the images
- Semantic labelling of the image
- Projecting the labelled images back to an initial 3D point cloud

Point-cloud preparation is a pre-processing step to remove the noise in the point clouds and produce mesh. Then, for the snapshot generation, two meshes, such as RGB mesh and composite mesh, are two different views generated from the various camera positions. The two input images were matched into pairs and underwent a semantic labelling process. Semantic labelling is a process to give a label to each pixel. Deep segmentation networks according to SegNet and fusion with residual correction were used. Finally, the back projection was carried out to project the labelled images back to the original 3D point cloud. Therefore, a labelled 3D point cloud was generated.

The SnapNet approach proposed by Boulch et al. (2017) was tested with various point clouds, including the point clouds obtained using Lidar sensors, multiple 2D views, and photogrammetry and RGB-D cameras. Figure 2.7 below shows semantic labelling of photogrammetric data for RGB, depth composite, and prediction map. Firstly, the Semantic 3D dataset has been used for the experiment of LiDAR point clouds. The dataset consists of 30 laser

acquisitions on ten different scenes from numerous places. The training data consists of 15 laser acquisitions, and the remaining 15 acquisitions are used to test in full semantic eight networks. The validation is set by using six acquisitions from the training data. The remaining nine training acquisitions will generate 3600 image pairs for the deep networks. The Stochastic Gradient Descent with momentum is used to train the network. The learning rate is beginning at 0.01 and varying based on a step-down policy. The VGG16 weights are used to help the initialization of the encoder of SegNet. The time to semantic the whole point cloud is about 41 minutes. The different natures of input images can define fusion strategy. Two mono-input SegNets were trained using RGB or depth composite images as the input to evaluate the performance of different fusion methods. Firstly, the addition of RGB and depth composite is not improved compared to depth composite only. This is because the high distribution difference in the prediction maps caused RGB's prediction to be overcome by composite.

On the other hand, if the fusion occurs before SegNet, the problem encountered in the addition of RGB and depth composite will be overcome. This will show an improved result. However, another problem like information loss has occurred. Last but not least, the residual correction network is introduced. The fusion was done after the SegNet, and the convolution is successful in refining the fusion. Therefore, the residual correction network is the best fusion strategy among these fusion strategies. The reduced-8 or full semantic-8 dataset was used as the dataset for the comparison of the result. This approach is used to compare with other approaches, including Random forest and Haris Net. Random forest is a method in which the 3D features in multiscale are trained using random forest classifiers, whereas Harris Net uses 3D Harris point extraction and a deep framework for classification. The approaches used in this research are SegNet and U-Net. The overall performance of U-Net is the best among these methods. The average IoU is the highest by using U-Net because the zoom strategy of U-Net is better in labelling cars and artifacts.

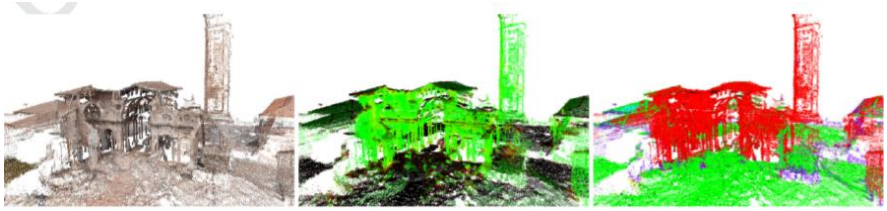

Figure 2.7: Semantic labelling of photogrammetric data. The left one is RGB, the middle one is depth composite and the right one is prediction map (Boulch, et al., 2017).

Besides, this approach was tested on photogrammetric point clouds for two experiments. The first one is transferring the semantic-8 network directly to photogrammetry. The full semantic-8 training set is labelling's network. The majority of labelling errors were concentrated on high vegetation and ground classes. The error of mislabelling on ground classes was due to the rubbles covered on the ground and the small inclination of a rooftop. The mislabel of high vegetation on destroy parts is because noise estimation conflicts with the class of building. The second one was using new classes to fine-tune for labelling. The point cloud was labelled by using rubble and non-rubble classes. The RGB and composite networks were fine-tuned by replacing the two new classes in the last classes.

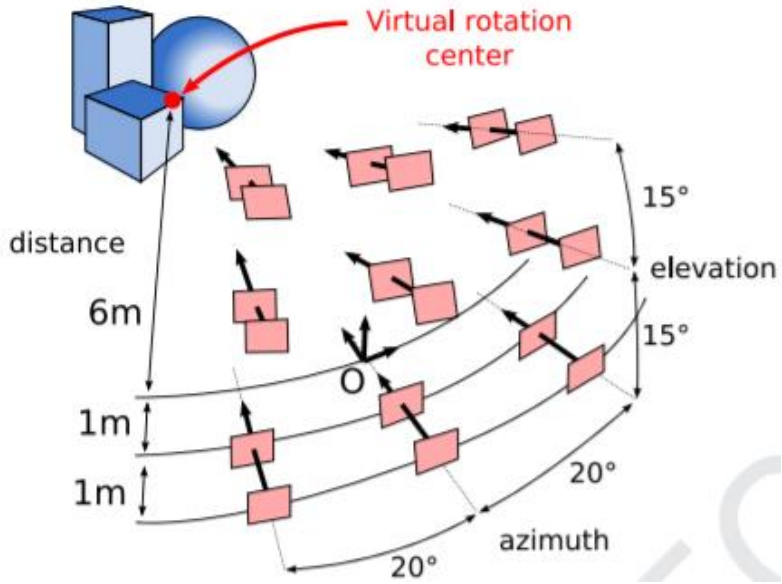


Figure 2.8: View generation strategy for RGB-D data (Boulch, et al.,2017).

Last but not least, this approach is tested with the point clouds generated by RGB-D cameras. SUN RGB-D dataset was used, and the dataset consisted of more than 10000 images from numerous RGB-D sensors. Firstly, an image inpainting technique was used to enhance the image space by filling the missing parts of the acquisition, resulting in smoother and more persistent point clouds. The meshing process was carried out, and the elongated face at different depths was removed. Figure 2.8 shows the view generation strategy for RGB-D data. For the view generation, many RGB-D sensors were required to snapshot the whole scene because of the small field of view of the sensors. The virtual rotation point was set at 6m from the origin, and generate the view at 8m from the virtual rotation point. The cameras were placed in front with an angle of 20° in both left and right. The process was repeated for angles 0°, 15°, and 30° to produce the 18 views per scene. Each class should be weighted to overcome the class imbalance problem of the SUN RGB-D dataset. The post-processing step was to fill the holes of the prediction map by nearest neighbour propagation. The mean accuracy of SnapNet is 67.4 % which is the highest among the methods, including SceneNet and SUNRGBD.

## 2.4 Manual 3D Point Cloud Labelling

Manual 3D point cloud labelling enables users to complete the labelling with the aid of a user interface. The user interface can load the point clouds and enable users to add labels on the 3D point clouds. There are two types of labelling user interfaces, including screen-based and VR-based (Li et al., 2020). A screen-based system can visualize the point cloud on a computer, whereas a VR-based system is dependent on a VR device. Then, for the screen-based labelling system, it can be further classified into web-based and PC-based. For the web-based system, the labelling process can be performed using a web browser, whereas, for the PC-based system, the software is needed to be installed to perform the labelling process.

### 2.4.1 PointAtMe

Wirth et al. (2019) stated that the 3D point cloud could be directly labelled without converting into a 2D image using a virtual reality device. A VR-based labelling tool, PointAtMe was created by Wirth et al. (2019) to complete the

data labelling. The label is a cuboid label type that is suitable to label cuboid-like objects, such as vehicles, pedestrians, etc. The machine learning community was allowed to access the tool and build a new community-labelled dataset for autonomous driving. Besides, an annotation benchmark was planned to be set up by the commercial annotation companies.

PointAtMe consists of a VR device and Oculus Rift Touch Controllers, used to annotate the data. The advantage of using PointAtMe is the speed of the data annotation process is relatively fast. The DoF of 3D bounding boxes created by this tool is nine, which had extra two DoF compare to the standard tool. During the annotation process, a transparent dummy bounding box with nine DoF appears between the controller. The nine DoF's position and scale have been defined by using red anchors, which are installed directly to the virtual controller. The color axes are installed on the right controller to grab the attention of the annotator. The created bounding box can suit the actual size of the object in the real world using this annotation tool. Thus, the image and point cloud can be visualized in Unity, either measured from a moving platform or a stationary platform.


Figure 2.9: The left and right Oculus Touch Controller (Wirth, et al., 2019).

Figure 2.9 shows the left and right Oculus Touch Controller. The left controller functions to control, understand and manage the scenes, whereas the right controller functions for data annotation. The left controller consists of 5 components with different functions. The components are hand trigger, index

trigger, thumbstick, button X, and Button Y. Hand trigger was used to move the point cloud with 6 DoF. Index trigger was used to release two rotational DoF, such as the box and point cloud. Switching the scene can be done by using a thumbstick. Button Y functions to show four images with different directions at the current scene, whereas button X functions to switch the scene's scale to label the objects with different scales. There are four components for the right oculus touch controller, such as index trigger, thumbstick, button A and button B. The functions of the index trigger and thumbstick of the left and right controller are different. The index trigger for the right controller was used to accept the dialogs. The dialogs contained the metadata will pop up while meeting a new box. The thumbstick for a right controller was used to switch the tracks to either left or right. By pressing button A and button B, a new track with dialogue and a new box with dialogue can be created, respectively. Besides, users can add information about label quality to the new box with dialogue (Wirth et al., 2019).

Wirth et al. (2019) used the KITTI 3D Object Detection scenes as the datasets. This tool's annotation speed can be determined by measuring the average speed used by inexperienced annotators to complete labelling process. The annotators are given 40 minutes to learn the technique to use the labelling tool. Then, the annotators start labelling four KITTI scenes, and the annotation time was recorded. Besides, the annotation accuracy is determined by comparing to Ground Truth offered by KITTI. The developers generated the Ground Truth as well because the labels of KITTI are slightly different. The average time for labelling one object in four scenes is 37.5 seconds.

This annotation tool is useful in the automated robot application, which performs in the mobile platform. Thus, the labelling process can be done directly to the 3D objects rather than 2D images using this kind of method to label the 3D point clouds. The result will be more accurate because the created bounding box is direct to the objects in 3D form, which results in a more accurate label, and the likelihood to mislabel objects will decrease. Wirth et al. (2019) could make further improvements to improve the label's quality. The further step is to visualize the points directly on the created bounding box.

### 2.4.2 3D BAT

3D BAT is a 3D point cloud labelling tool proposed by Zimmer, Rangesh, and Trivedi (2019). The proposed labelling system is efficiently used for 3D localization and detection. The tool was allowed the user or annotator to label the objects in 3D point clouds with a toolbox's aid. Besides, this is a web-based tool that can be opened by using the browsers in the computer. This project works on both 2D images and a 3D point cloud of the street scenes. This is because the user will label the objects directly in 3D point clouds, and the labels will be transferred back to the images. Therefore, the user is able to see the labels in the images. The data labelling will be done on the Laser data with rough bounding primitives, and the labels will be transferred to the images by using a geometric model. Figure 2.10 compares the functions of 3D BAT to the other open-source and commercial labelling tools. 3D BAT showed the highest score among all the labelling tools, which had all the functions from F1 to F14 shown in Figure 2.10.

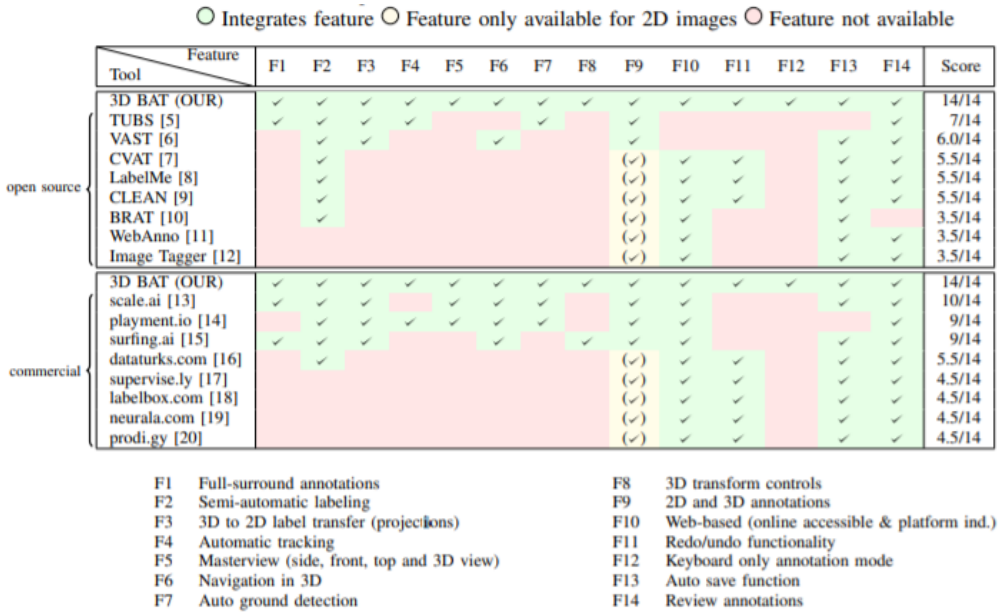


Figure 2.10: Comparison of the labelling tools (Zimmer, Rangesh and Trivedi, 2019).

The system architecture of 3D BAT was described below. Web Graphics Library (WebGL) is JavaScript API, which allows collaborative labelling in this research. Multiple labels on the objects at the same time are

prevented. Firstly, the user interface will display all the camera images from different views. Thus, the user was able to view the objects in the images from multiple cameras. Then, the frames between the first and last frame will be labelled using the semi-automatic labelling method.

The labelling time decreases by implementing the semi-automatic interpolation technique. Then, the labels were transferred from 3D point cloud to 2D images by using a 3D and projective geometry model. This model is very useful because the labels done on the 3D point clouds will be projected automatically to the 2D images. Therefore, the annotator can label the point clouds by referring to the camera images, which will reduce the labelling time and increase the label's accuracy as the projection of 3D to 2D in real-time. There are a front view, side view, and bird-eye view of 3D point clouds provided for the user. The user was allowed to translate, rotate and scale the targeted object.

The datasets used for the labelling tool are nuScenes, and LISA-T and the dataset obtained using a full-surround sensor configuration. Five classes can be labelled, which are car, truck, motorcycle, bicycle, and pedestrian. The colours of labels for car, truck, motorcycle, bicycle, and pedestrian are green, yellow, red, pink, and blue, respectively. For the LISA-T datasets, the result obtained was evaluated by comparing to ground truth in terms of 3D IoU, precision, recall, and F1-score. Precision and recall are used to evaluate the accuracy of the labels. The harmonic mean of recall and precision is F1-score. The results of user annotations were obtained by taking the average results from 3 users. The average IoU for the growth truth is 0.138, and for the user, annotations are 0.066. The average precision, recall, and F1-score are 0.0117, 0.0113, and 0.0377, respectively. For nuScenes dataset, the average IoU for ground truth is 0.0748, and for a user, annotations are 0.0393. The average precision, recall, and F1-score are 0.007, 0.0067, and 0.007, respectively.

### 2.4.3 LATTE

Bernie et al. (2019) had designed an open-source web-based labelling tool, called LATTE and the labelling process can be done by using a mouse. This tool is designed to address the challenges faced in 3D LiDAR point cloud labelling. There are three challenges listed, including low resolution of LiDAR

point cloud, complex annotating operations, and sequential correlation. The LiDAR point cloud resolution is relatively low compared to the resolution of images captured by the camera, which will make it difficult for the annotator to recognize the objects in point clouds. Besides, 3D bounding box labelling makes a cuboid label on an object in 3D point clouds and collects the points that belong to a target object. 3D bounding box labelling is more complex than 2D bounding box labelling. This is because drawing a 2D bounding box is considered two dimensions. The user can create labelling by drawing two corners, whereas drawing a 3D bounding box is needed to consider the three dimensions, centre position, and 3D rotation. Besides, sequential correlation is the annotations that might be repeated due to the point cloud data are collected in sequences and consecutive frames, which results in high correlation.

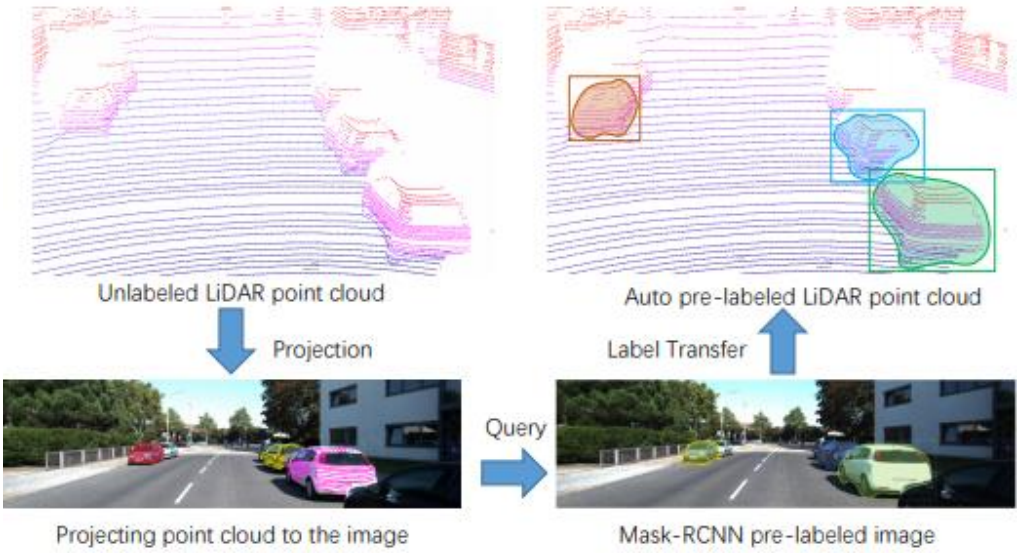


Figure 2.11: The pipeline of sensor fusion (Bernie et al.,2019).

To improve this labelling tool's performance, sensor fusion, one-click annotation, and tracking function are added. In this research, the point cloud labelling can be done with the aid of image-based detection algorithms. This is because image-based detection algorithms are more sophisticated than LiDAR-based detection algorithm. The LiDAR point clouds are collected by using the combination of cameras and the LiDAR sensor. Each point in the collected point clouds can be projected to the corresponding pixel image. According to Figure 2.11, semantic segmentation was conducted on the image.

Mask R-CNN was used to obtain the labels for each pixel in the image and eventually transfer them to the 3D point clouds. Thus, the pre-labels for the point clouds are generated automatically. Mask R-CNN helps annotators recognize the object in a 3D point cloud and adjust the labels. Besides, one-click annotation is applied to reduce the complexity of the annotation process. The label is a 3D bounding box, which is more simple to use comparing to point-wise labelling. In the perspective view, the 2D bounding boxes are shown on the objects' top, eventually to one-click annotation. One-click annotation enables a user to label the objects in point clouds by clicking one point in the object. One-click annotation uses clustering algorithms to find all target object's points and display a 2D bounding box on the object. By using one-click annotation, effectively reduces the labelling time. Lastly, a tracking algorithm is used to transfer the labels from one frame to another. Kalman filtering is used to track the centre of the bounding box of the target object. When the annotators annotate the first frame in a sequence, the tracking algorithm will estimate bounding boxes' centres for the next frame.

| Method | IoU (%) | IoU(%) w/ KITTI | Time (s) | #ops |
|---|---|---|---|---|
| Ground Truth | 100.0 | 86.0 | - | - |
| Baseline | 85.5 | 82.3 | 9.51 | 3.76 |
| Sensor fusion | 86.3 | 82.5 | 3.88 | 2.88 |
| One-click annotation | 86.2 | 82.9 | 2.55 | 1.29 |
| Tracking | 86.4 | 83.5 | 2.41 | 1.53 |
| Full features | 87.5 | 84.7 | 1.53 | 1.02 |

Figure 2.12: The results of basline, sensor fusion, one-click annotation, tracking and full features in term of IoU, time and operation counts (Bernie et al., 2019).

The labelling speed of this labelling tool is measured by taking the average labelling time of users to label the LiDAR point cloud scene, taken from the KITTI dataset. The volunteers are given sufficient time to practice, and the results are recorded when the volunteers are confident to start labelling. The volunteers use the three features, such as sensor fusion, one-click annotation, and tracking algorithm, to label a sequence of 5 frames, and the

labels were compared to the ground truth labels provided by an expert. The results are evaluated in terms of IoU between the ground truth and the bounding boxes created by volunteers. According to Figure 2.12, the IoU of the methods is between 85.5 % to 87.5 % when comparing to the expert's ground truth. The IoU is between 82.3 % to 84.7 % when comparing ground truth provided by KITTI. The baseline method is for the user to label the point clouds without using the three features. Therefore, the time used to label one object is much longer than other methods. Full features are users used all three features to complete the labelling. It is much more accurate and efficient as the IoU, time, and operation counts are 87.5 %, 1.53 s, and 1.02 counts, respectively.

### 2.4.4 POINTS

Li et al. (2020) had developed a 3D point cloud labelling tool for autonomous driving datasets. POINTS is a screen-based labelling system and web-based. The major challenges faced to develop a 3D point cloud labelling system are convenient user interface, scalable annotation tools, and geometric data units. In this paper, visualization modules, interactive tools for 3D labelling, and transfer methods were developed to address the challenges.

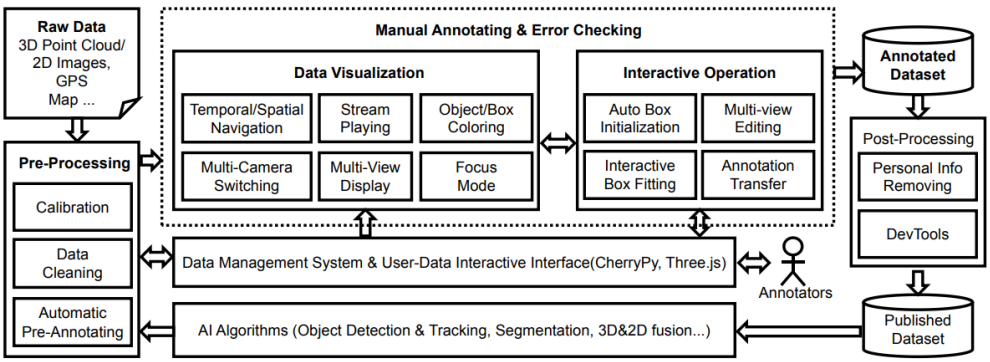


Figure 2.13: The pipeline of POINTS (Li et al., 2020).

Firstly, the raw data should be prepared and undergo pre-processing. In the data pre-processing step, the sensor parameters are estimated, and the parameters are calibrated. Then, AI algorithms will generate the initial results of labelling. The data management system and user-data interactive interface programmed with CherryPy and Three.js are responsible for managing the

data after the pre-processing stage. CherryPy is an object-oriented web framework. Python is used as the programming language, and Three.js is a JavaScript library and API used to perform 3D computer graphics in a web browser by using WebGL. The data visualization module integrated with an interactive operation to perform manual annotating and error checking with the functions listed in Figure 2.13.

Data visualization modules can review the labelling results in terms of 3D bounding boxes. The primary user interface consists of one main view, three projective views, two photo contexts, a context menu, and a fast-floating toolbox. The main view is displaying whole point clouds, and three projective views show the selected objects' top, front and side views. One of the photo contexts is resizable and switched automatically among multiple cameras, and another one focused photo context is automatically selected for the chosen object. A context menu provides tools for context operation, and a fast-floating toolbox provides tools for adjusting 3D bounding boxes. Besides, there are many functions available to add benefits to the user interface, including focus mode, camera auto-switching, object colouring, navigation, stream play, and object locking. When turning on the focus mode, the selected object will be zoomed and centered in the main view. Camera auto-switching is switching the camera automatically, which is more relevant to the objects. According to the classes, object colouring will highlight the boxes and enclosed points in the box in different colours. Apart from that, navigation enables the users to check the labelling results by changing the selected objects, point of view, and several frames. Stream play displays the sequential data, and the object locking is locking and highlighting the object while turning on the stream play.

On the other hand, 3D box initialization, 3D box editing in perspective view and projective sub-views, interactive box fitting, and transfer method are interactive operations, enabling the user to use the tool and finish the annotations effectively. For the 3D box initialization, when the user creates a new 3D bounding box, the box's initial orientation is upward along z axis of the main view. The bounding box will automatically shrink and fit the enclosed points object. The automatic shrink and fitting is an interactive box fitting function, which will be reviewed later. Besides, the object type will be recognized and automatically selected according to the dimension of the

objects. Next, POINTS enables users to edit the 3D bounding boxes in both perspective view and projective sub-views. It is more convenient to adjust the bounding boxes in projective sub-views compare to in perspective view. This is because editing in perspective view needs to change the viewpoint frequently while editing in projective sub views adjusts the boxes in the top, front and side views without changing the viewpoint.

Furthermore, interactive box fitting will find the minimal box, which is enclosing all the objects' points to resize the bounding box to fit the bounding box to the object automatically. With this algorithm, the labelling process is much easier to be done by drawing a rectangular box that encloses all the objects' points and sometimes needs to rotate 3D bounding boxes. However, this algorithm objects' point sets of ground are present. Lastly, the transfer method can transfer the annotations among the frames. This algorithm is efficient for autonomous datasets as the datasets always consist of independent data frames. It needs a combination of a 3D data registration algorithm and a 3D object tracking algorithm to perform automatic annotation transfer. 3D registration algorithm is used to adjust the boxes using the objects' sizes and orientation from the previous frame. A 3D object tracking algorithm can build the correspondence between the target objects in a current frame and reference objects from the previous frame. The 3D data registration algorithm will eventually figure out the analogous geometric transform between reference objects and target objects. The adjustment will be conducted on the boxes in the target frame.

| **Method** | **Ours (POINTS)** | **PointAtME [1]** |
|---|---|---|
| Time/Object (sec.) | $24 \pm 6$ | $55.2 \pm 12.1$ |
| Errors (Points/Object) | $<1$ | $28.7 \pm 5.1$ |
| FP ratio / Object | 0 | 7.16% |
| FN ratio / Object | $<1\%$ | 3.27% |

Figure 2.14: Comparison of results between POINTS and PointAtME (Li et al., 2020).

The performances of POINTS are evaluated in terms of annotation speed and annotation errors. The annotation accuracy is determined by comparing the labour-intensive annotation results and ground truth. The results

were obtained by three inexperienced users. According to Figure 2.14, the labelling time used in POINTS is much shorter compared to PointAtMe. Besides, the errors, false-positive ratio, and false-negative ratio are less than 1 %. It can be concluded that POINTS can produce more accurate labels in a shorter time. This is because POINTS is convenient to use and easy to understand the operation.

## 2.5 Summary

In conclusion, 3D point cloud labelling is very useful in object recognition and computer vision. Nowadays, more and more technologies adapt 3D point cloud labelling techniques in many applications, such as robotics, self-driving car, urban planning, disaster management, and so on. Some of the applications require object recognition skills to perform efficiently, and 3D point cloud labelling contributes much to these sectors. Therefore, there are many labelling tools perform labelling operation by using different methods. There are two types of labelling, including automatic and manual 3D point cloud labelling.

For automatic labelling, by using the voxel-based CNN technique, the overall accuracy is 93 %, which is relatively high. However, it may take a longer time to finish the process and poor in detecting an extra-large object. The accuracy in recognizing large objects is relatively low. The advantage of using the voxel-based CNN technique is the technique neglects the segmentation. Apart from that, the semantic labelling process of SnapNet is done in 2D images. When all the 2D images are labelled, the images will be projected back to the original 3D point cloud. This process may cause the loss of detailed information, and the results obtained will not accurate. Next, patch context analysis combined with multiscale processing will produce a result, which is more accurate compare to 2D deep segmentation network methods. The overall accuracy and average class accuracy are 88 % and 75.6 %, respectively.

For manual labelling tools, it can be classified into two categories, including screen-based and VR-based. PointAtMe is a VR-based system, whereas 3D BAT, LATTE, and POINTS are screen-based systems. The VR-based system can provide a better experience feeling, but motion sickness and inaccurate operation will affect labels' quality and accuracy. It is quite difficult

to control Oculus touch controllers to label the objects in VR world. Therefore, a screen-based system is preferable. In the screen-based system, it can be classified into PC-based and web-based systems. The PC-based system needs software installation, whereas web-based systems enable users to open the user interface through web browsers. 3D BAT, LATTE, and POINTS are web-based systems. The accuracy of these three user interfaces is quite high. Among three user interfaces, 3D BAT takes longer time than LATTE and POINTS to load all the point clouds and images. Besides, LATTE and POINTS have the fast annotation features. One-click annotation of LATTE and interactive box fitting of POINTS effectively reduces the annotation time and reduces labelling objects' complexity in a 3D point cloud. Apart from that, both LATTE and POINTS can transfer the annotations in a sequence of frames. However, the 3D box initialization algorithm of POINTS can recognize the classes of the labelled objects by determining the objects' dimensions. In addition, POINTS contains more functions in visualization modules compare to LATTE. The features in visualization module will ease the operation of user interface. The accuracy of labels depends on the annotators.

In this project, manual 3D point cloud labelling tools will be selected to achieve the aim and objectives of this project, which is providing a convenient user interface for users to traverse in a 3D environment and label the points or regions of point clouds. Among four manual labelling tools, POINTS will be adopted to continue this project as it is convenient and easy to use, and the accuracy of labels is quite high. The other user interfaces will be set up to make the comparison between the user interfaces' performances. Further improvements can be made to obtain maximized labels' accuracy within minimum time and make it easier for inexperienced users to use.

# CHAPTER 3

# METHODOLOGY AND WORK PLAN

## 3.1 Introduction

This chapter will discuss the methods used to complete and achieve the objectives of this project. This project's methodology will be described properly by providing all the necessary details to conduct the project. Besides, a work plan is concerned about planning the project with the activities identified and explained well. Gantt charts can describe the work plans for the part I and part II projects. The duration of the whole project is one year. Therefore, the part I project had been conducted in May semester 2020, and the part II project had been carried out in January semester 2021.

This project aims to compare the user interface for labelling 3D point clouds. Before starting the software development, accessing other user interfaces on point cloud labelling is important to get the ideas. The first two tools that had been tried out are CloudCompare and Meshlab. Some raw point clouds had been downloaded from the webpage and opened by using CloudCompare and Meshlab. According to the literature review, manual open-source 3D point cloud labelling tools will be selected to achieve this project's aim and objectives. This project was decided to be done by using a screen-based system instead of a VR-based system. This is because VR device is relatively expensive, and therefore less accessible. Screen-based system allows the users to label the objects in 3D point clouds on a PC by using a mouse and keyboard.

## 3.2 Tools for Implementation of Algorithms

In this project, POINTS was adopted to be modified. An IDE will be used to run the algorithm. Visual Studio Code was selected as the IDE for this project. Visual Studio Code is a free and open-source code editor, which can run in Windows, Linux, and macOS (Visual Studio Code, n.d.). This IDE supports IntelliSense, run and debug, built-in Git, and a lot of extensions. It also supports the programming languages used in POINTS, including Python, JavaScript, CSS, and HTML. Visual Studio Code aims to provide simple tools

that are needed for the code-build-debug cycle, which is much easier to use than complex workflows of full-featured IDE, such as Microsoft Visual Studio. Besides, POINTS is a web-based labelling tool. Web browsers, including Mozilla Firefox, Google Chrome, etc., are necessary to open the user interface. Most of the web browsers support this user interface.

## 3.3 Software Development

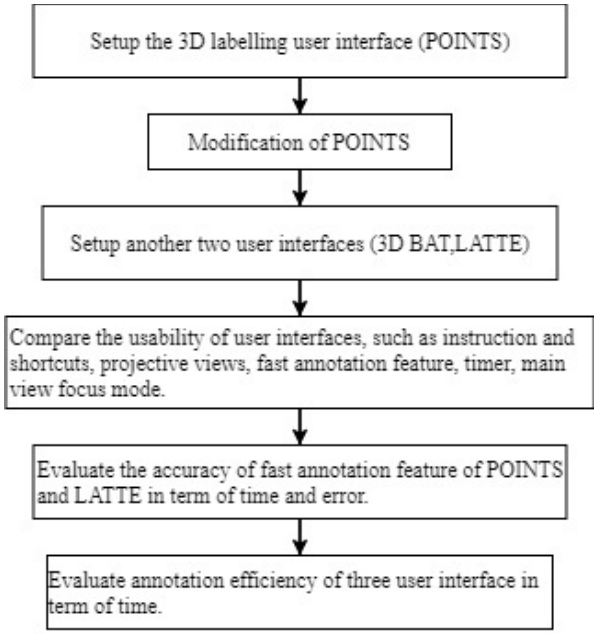


Figure 3.1: The flows to conduct the project.

Figure 3.1 shows the flow to conduct the overall project. First of all, POINTS will be set up. POINTS needs a pre-trained model. The 'deep_innotation_inference.h5' file, which consists of the DNN-based interactive point cloud annotation algorithm, is downloaded. This algorithm is created by modifying the code of PointNet, which the library used to build up the algorithm is TensorFlow 2.1. PointNet uses a single symmetric function and max-pooling to handle an unordered input set (Qi et al., 2017). The network will select and encode the point cloud points by learning a set of optimization functions. The final full network layers will combine the optimal values learned from the network and stored them in the global descriptor for shape segmentation or shape classification. Before starting the operation of PointNet, a data-dependent spatial transformer network will be built, which can canonicalize the data. If each point is transformed independently, the input

format is relatively simple to apply affine transformations, which will improve the performance of PointNet. After install the pre-trained model, the required packages were installed with the command ‘pip install -r requirement.txt’. Ensure that the Python, CherryPy, and TensorFlow versions are larger than 2.1.

The web applications of POINTS are built using CherryPy. CherryPy is an object-oriented HTTP framework, which allows developers to build the web application easily. CherryPy is written in Python. The developer will spend lesser time developing smaller source code. In this project, JavaScript, HTML, and CSS are used to add the user interface's functions or tools and design the user interface. CherryPy provides support to serve the static content, including JavaScipt, HTML, CSS, and images, to end-users. When running the main python code, the server will start to set up the web application. As the application is ready to be used, users can access the application on the address http://127.0.0.1:8080 by using web browsers.

This user interface consists of two main components: visualization modules and 3D bounding boxes interactive tools. Visualization modules can review the labelling results in terms of 3D bounding boxes, distinguish data of different objects, and display projective views of an object in top, front, and side view. All the user interface tools were written in JavaScript, and the design of the user interface was written in HTML with CSS. The index.html is used to set up the user interface structures, and the main CSS is used to describe the presentations of structures, such as fonts, colours, layout, size, and so on, from index.html. The scene_reader.py is functioned to load the point clouds and images. The file format of point clouds and images that can be loaded is PCD and jpg format, respectively. As the annotator finishes the annotations, it will read the annotations and save the annotations in a JSON file. The main.py will import the scene_reader.py. After running the main.py, the user interface displays visualization modules as the CherryPy will load the index.html and scene_reader.py. The visualization modules consist of the main view, projective views, images, point clouds, context menu, and fast-floating toolbox.

On the other hand, the 3D interactive tools mainly consist of 3D box initialization, interactive box fitting algorithm, and 3D box editing in perspective view and projective view. 3D interactive tools are used to ease the

labelling process for the user. 3D box editing in perspective view and projective view enables the user to edit the bounding boxes in two ways. As the bounding box is selected, the projective views will show the bounding box's top, front and side views. The adjustment made to the bounding box in a projective view will affect the perspective view and vice versa, which means both projective views and perspective view are correlated. 3D box initialization is working with interactive box fitting. As a bounding box is created to enclose an object in point clouds, the box's orientation is upward. Then, all of the object's points will be predicted by using the Euclidean distance-based growing algorithm. After that, an interactive box fitting algorithm will be used to reduce the box size to enclose only the objects' points.

A good user interface enables users to understand and use it intuitively. The user interface must be simple but contain sufficient tips and tricks to help the operation of users. Some improvements will be carried out to ease the annotations for inexperienced users. The initial design of POINTS only consists of one main view. Inexperienced users might not know how to control the user interface to make labels on the objects. Therefore, one tab was added. There are three tab contents, including main UI, instruction, and shortcut keys. The tab is created using combination of HTML, JavaScript and CSS, which can refer to APPENDIX A. HTML is used to design the structures of tab, while CSS is used to style the colour, font, position, etc. JavaScript is written to provide function to the tab. The main UI tab consists of the original content of the POINTS. For the instruction tab, a teaching video provides a guide to the novices to master this user interface in a short time without going to the GitHub webpage to search the teaching video. This instruction video was recorded by using a screen recording of a laptop. Then, after editing the video and inserting captions to the video, the video was uploaded to Youtube. The <iframe> tag was used to import the video from Youtube to this tab. Besides, the shortcut keys of keyboard and operation of mouse for perspective view and the projective view were listed in two tables. The tables' structures were done using HTML, while the design of the structures, such as font, colour, size, and so on, was done using CSS. The time needed to learn and understand the user interface, and the annotation time will be significantly decreased by adding

both instruction and shortcut keys tabs. Apart from that, for the projective views, the original user interface shows only the point clouds in three views without mention the type of views. Therefore, the words 'top view', 'front view', and 'side view' were added to the specific views to make inexperienced users recognize the projective views. Furthermore, a timer function was added to the 'Main UI' tab. The HTML, CSS and JavaScript codes of timer function can refer to APPENDIX A. This built-in timer will start as the users open the user interface. This will make it convenient for users who want to record the labelling time because an external time keeper is unnecessary. These extra features may make the user interface more convenient to use and can be evaluated in terms of usability, which can achieve the objective of this project.

After the modification of POINTS, 3D BAT and LATTE were set up, respectively. The first step to set up 3D BAT is cloning the repository using HTTPS from GitHub to a local computer. This is because clone a repository will make it easy to push or pull the changes to GitHub. Then, NPM with Node.js was installed in order to ease the JavaScript developers to share, reuse and update the code. An IDE with the integrated web server, PHP Storm, was downloaded and installed to run the code of 3D BAT. Some point clouds from the NuScene datasets were downloaded and extracted to the 'input' folder. Lastly, the required packages for the code of 3D BAT were installed by using the 'npm install' command. When all the things had been successfully installed, the 3D BAT user interface can run on localhost by using PHP storm.

Once 3D BAT was successfully set up, the LATTE was set up. The first step is the same as 3D BAT, which is cloning the repository from GitHub. The pre-trained model, 'mask_rcnn_coco.h5' file, was downloaded and extracted to the 'Mask_RCNN' folder. After that, the command' pip3 install -r requirements.txt' was inputted, the dependencies will be installed. All the libraries needed are in the packages. After successfully installing the required packages, the user interface can work by running the 'app.py' Python code. Once the system was ready, the user can open the user interface on a web browser through http://127.0.0.1:5000/ address.

Once the three user interfaces were successfully set up, a comparison between POINTS, 3D BAT, and LATTE, which are web-based 3D point cloud labelling tools, is carried out to evaluate user interfaces performance. The

same point cloud scene from KITTI datasets in BIN file format was used as reference point clouds. POINTS and 3D BAT can load the point clouds in PCD format files instead of BIN format files. Therefore, conversion of the BIN to PCD format is necessary. A BIN to PCD algorithm was applied to convert the BIN file to a PCD format file. The Python code for the conversion can refer to APPENDIX A. After the conversion was done, the comparison between POINTS, 3D BAT, and LATTE was conducted. The performances of these user interfaces are evaluated in terms of usability and labelling speed. The usability includes instruction and shortcuts, projective views, fast annotation feature, timer, and main view focus mode. For the labelling speed, evaluation of results is in terms of time. Thus, the objectives of this project will be achieved. Screen recording on a laptop can record the labelling process in user interfaces. The accuracy of a user interface can be determined by comparing the labelling result to ground truth.

## 3.4 Schedule of Project Activities

Project planning is an important criterion to ensure that the project can be done on time. Schedule planning is a critical component of time management. Gantt chart is a great way to show the schedule in a simple form. Allocation of time wisely helps the completion of a project become more efficient. In this project, the whole project duration is one year and separated into parts I project and part II project. The part I project had been conducted in May semester 2020, and the part II project had been carried out in January semester 2021.

Table 3.1: The Gantt chart of the activities in part I project for 14 weeks.

| No. | Project Activities | W 1 | W 2 | W 3 | W 4 | W 5 | W 6 | W 7 | W 8 | W 9 | W 10 | W 11 | W 12 | W 13 | W 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M1 | Project Planning | | | | | | | | | | | | | | |
| M2 | Literature review | | | | | | | | | | | | | | |
| M3 | Setup the environment | | | | | | | | | | | | | | |
| M4 | Report writing | | | | | | | | | | | | | | |
| M5 | Oral presentation | | | | | | | | | | | | | | |

Table 3.1 shows the schedule planning for 14 weeks of May semester 2020. It is mainly consisted of five activities to complete part 1 of the final year project. For the first two weeks, project planning was carried out with the aid of the supervisor, Dr. Ng Oon-Ee. Dr. Ng had clearly defined the aim and objectives of this project and the desired outcome to help in effective project planning. Besides, Dr. Ng had given some suggestions for allocating time to finish the project on time. With the aid of Dr. Ng, the project planning was done within two weeks.

In weeks 2, literature reviews were carried out. The literature reviews are conducted to master the knowledge about the 3D point cloud. Some articles or journals about 3D point cloud labelling were being found from google scholar and the UTAR library. The article or journal discussed in-depth the problem encountered in 3D point cloud labelling and proposed methods to accomplish the 3D point cloud labelling and overcome the problems. The literature reviews were conducted within nine weeks. Among all the articles or journals, three automatic 3D point cloud labelling tools and four manual 3D point cloud labelling tools had been reviewed. Three automatic 3D point cloud labelling tools are voxel-based 3D CNN, Patch Context Analysis and Multiscale Processing, and SnapNet. In contrast, four manual 3D point cloud labelling tools are PointAtMe, 3D BAT, LATTE, and POINTS. The literature reviews had been done in week 10.

Besides, to set up the environment, research was carried out. The environment was planned to be set up from week 3 to week 7. However, the plan had been delayed until week nine and done on 10 August 2020. This is because it is necessary to do revision for midterm tests and do the assignment, which causes a lack of time. The next activity, report writing, was started from week 7. The report consists of four parts, including introduction, literature review, methodology and work plan, as well as problems and recommended solution. The introduction describes the general knowledge about 3D point cloud and object labelling in the 3D point cloud and defines the aim and objectives of this project as well as states the current problem in 3D point cloud labelling. The literature review part was done with the discussion on the articles or journals in-depth about the methods used to label the point clouds in 3D space. The methodology and work plan part describe the methods used to

achieve this project's aim and objectives and the detail of the project planning and managing. The report had been done in week 13, on 9 September 2020, and submitted on time. Lastly, preparation of oral presentation was carried out in week 13, and the presentation was conducted on 15 September 2020. All of the activities were done on time, except setting up the environment had been delayed for two weeks. Then, the part II project was conducted in January semester 2021, and the duration was 14 weeks. This project will be fully done in part II project.

Table 3.2: The Gantt chart for the activities in part II project for 14 weeks.

| No. | Project Activities | W 1 | W 2 | W 3 | W 4 | W 5 | W 6 | W 7 | W 8 | W 9 | W 10 | W 11 | W 12 | W 13 | W 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M1 | Software development | | | | | | | | | | | | | | |
| M2 | Preliminary testing | | | | | | | | | | | | | | |
| M3 | Result and discussion | | | | | | | | | | | | | | |
| M4 | Evaluation of the performance | | | | | | | | | | | | | | |
| M5 | Poster design | | | | | | | | | | | | | | |
| M6 | Report writing | | | | | | | | | | | | | | |

Table 3.2 shows the Gantt chart for the part II project for 14 weeks of January semester 2021. It is mainly consisted of six activities to be accomplished in the part II project. Firstly, the software development was carried out starting from week 1. The duration for software development will be longer. Therefore, nine weeks are given to complete software development due to the modification and improvement of code that will be carried out. The works described in section 3.3 are software development, in which the code of POINTS was adopted and improved by some modifications. Software development had been done on 18 March 2021, which follow the planned schedule. Then, the preliminary test was carried out from week 4 to week 6. By carrying out the preliminary test, the software's performance can be evaluated, and the problems can be recognized earlier. Therefore, the software can be improved to overcome the problems encountered and boost the user interface's performance. The testing was done on week 6. The results and

discussion were conducted in week 6. The discussion on the obtained results was carried out to analyse the user interface's performance.

Besides, evaluation of the designed user interface's performance was carried out along with results and discussion. The duration of this activity is seven weeks, which was starting from week 6. The performance of POINTS was compared to 3D BAT and LATTE, which are open-source web-based labelling tools. The evaluation of the labelling tools' performance was done on the same open-source point cloud scene from KITTI datasets. This activity was conducted until week 12 as the designed user interface may be modified or improved, which may affect the performance. All the results had been obtained before 10 April 2021. The deadline for the submission of the poster was on week 12. The final design of the poster had been done on 8 April 2021, before the deadline on week 12. Finally, the report writing started from week 8 to ensure the time is sufficient to complete the report properly. The report was successfully done on 17 April 2021. All the activities had been completed follow the Gantt chart in Table 3.2. Upon the final report had been submitted, the presentation had been prepared.

## 3.5 Problem Faced and Solutions

In this project, three user interfaces were set up for comparison. During setting up POINTS, 3D BAT, and LATTE, there are some problems encountered, which cause the user interface cannot to work well.

### 3.5.1 Setup POINTS

First of all, in order to set up POINTS, Python, CherryPy, and TensorFlow libraries was necessary. The versions of these libraries must be larger than 2.1. Initially, the Python version on a local computer was 3.7.4. Then, the latest version of CherryPy 18.6.1 and TensorFlow 2.4.1 was successfully installed by input the commands "pip install cherrypy" and "pip install tensorflow" in the terminal. Then, the command "pip install -r requirement.txt" was inputted to install all the required libraries to set up POINTS. However, there was an error encountered during the installation of the packages. The showing error is "Fix the pip error: Couldn't find a version that satisfies the requirement". Therefore, all the versions of libraries in the package were checked in order to

determine the required version of Python for the libraries in packages. However, it is quite difficult to check all the libraries in the package, as it consists of more than 80 libraries in the package. Therefore, the Python version was upgraded to Python 3.8.8, and it can successfully install all the libraries in the package. This is because some of the libraries in the package need Python version 3.8 to support. After running the main.py function, the server will be ready to use, and the web application can be opened through http://127.0.0.1:8080 by using web browsers.

### 3.5.2 Setup LATTE

During the installation of required libraries in the package of LATTE, one library called Matplotlib with 3.0.3 version was unable to be built and installed. The version of Matplolib is specific as the developer uses this version to build up the algorithm. The errors show FreeType and Libpng are not found. Thus, wheels for the Matplotlib 3.0.3 version were downloaded and installed. After the installation was done, the error was still there. This is because Matplotlib needs some dependencies, including Python, Numpy, setup tools, cycler, Dateutil, Kiwisolver, Pillow, and Pyparsing, with specific versions or larger than the specific versions. After researching Matplotlib 3.0.3, pip is trying to build from sources as it allows to provide wheels for Python 3.5 to 3.7 version instead of 3.8 version. Therefore, the Python version was downgraded to Python 3.7.9. After downgrade the Python version, LATTE was successfully to be setup.

### 3.5.3 Setup 3D BAT

There was no error detected during setting up 3D BAT. However, if the index.html of 3D BAT is directly open in the browser, it cannot load the PCD files. The user interface is displaying the images and some tools. This is because the user interface will occupy large CPU usage and relatively slow to load all the PCD files. This user interface works well on GPU instead of CPU. It is necessary to use localhost to open the user interface.

In the beginning, the tool used to open 3D BAT is Visual Studio Code. Visual Studio Code does not consist of a built-in web server. Thus, Visual Studio Code unable to open the index.html file using localhost. After installing

the NPM and Node.js, the PCD files are unable to be loaded as well. Thus, another solution was implemented to run 3D BAT. PHP Storm was used instead of Visual Studio Code. This is because PHP Storm has a built-in web server. After that, 3D BAT is able to load all the point clouds, images and tools. However, there is a limitation for PHP Storm. This IDE is not free of charge. It provides a duration time for trial only. If the duration time is past, the PHP Storm can work for 30 minutes. Since 3D BAT is used for comparison, hence it will not affect much.

### 3.5.4 Conversion of KITTI Dataset File Format

Besides, during the comparison of three user interfaces, all user interfaces are working with different file format types of point clouds. In this project, point cloud scene from KITTI datasets in BIN file format were used as reference point clouds. Due to POINTS and 3D BAT are only able to work with point clouds in PCD format. Thus, the point clouds from KITTI datasets are necessary to be converted into PCD format. The first try is using CloudCompare to do the conversion of BIN file to PCD file. However, the PCD file format after conversion is not suitable for 3D BAT and POINTS. Then, another algorithm was built in Python. There are two libraries needed, including struct and open3d. Struct library was used to convert Python values into a string of bytes and vice versa, whereas open3d is a library used to perform 3D data processing. Firstly, the struct.unpack() function was used to convert the string of binary representations into the original representations with the specified format in terms of x, y, z, and intensity. After the conversion, the results of x, y, and z were appended into the NumPy array. The NumPy array will be passed to open3d.o3d.geometry.PointCloud() function to obtain the geometry of point clouds. After that, open3d.utility.Vector3dVector() function was used to convert the NumPy matrix to 3D vectors. Lastly, o3d.io.write_point_cloud() function is used to write the point cloud into a PCD file. The original point clouds in BIN file format were successfully converted into PCD file format using this algorithm, and both POINTS and 3D BAT can load the converted PCD files.

## 3.6 Summary

POINTS is the selected user interface to be adopted in this project. After modification of POINTS, the other two open-source user interfaces, 3D BAT and LATTE, are set up as well. A comparison between three user interfaces will be conducted to achieve the objectives. The results will be evaluated in terms of usability and labelling speed of user interfaces. All of the project activities are considered done on time and followed the planning in the Gantt chart. Lastly, the problems faced during set up POINTS, 3D BAT and LATTE, and the conversion of the point clouds file format are successfully overcome.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Introduction

This research aims to compare the user interfaces for 3D point cloud labelling, which enables users to traverse the 3D environment and label volumes of points in a 3D point cloud. By adopting an open-source user interface from the web and modifying it, the results obtained will be analysed by comparing the result with other open-source user interfaces. The results were collected using three different user interfaces to label the same point cloud scene from the KITTI dataset.

## 4.2 Comparison of User Interface

Due to the 3D point cloud labelling tools in the marketplace are pretty expensive, the comparison of open-source user interfaces will be made to select a free and convenient user interface, which can perform well. Comparison of the user interfaces' useful features can achieve the objective in this project, which is evaluating usability between the user interfaces. An open-source user interface with useful features to ease the labelling process can be determined. If the open-source 3D point cloud labelling tool consists of some useful features as the 3D point cloud labelling tool in the marketplace, the problem can be overcome.

This project's aim and objectives were achieved by setting up three different open-source user interfaces, including POINTS, 3D BAT, and LATTE, because the comparison can be made between these three user interfaces. POINTS, 3D BAT, and LATTE are screen-based and web-based user interfaces, which can run using a web browser on the computer. All three user interfaces allow users to traverse in a 3D environment. The labels can be made in perspective views. The labels for POINTS and 3D BAT are 3D bounding boxes, whereas the labels for LATTE are 2D bounding boxes. The bounding boxes for different classes will be assigned with different colours in POINTS and 3D BATS, whereas the colour of bounding boxes in LATTE for different classes are the same. The points of objects enclosed by bounding

boxes will be highlighted in the same colour as bounding boxes in POINTS. The bounding boxes and object colouring will make the user easy to recognize the classes for the objects. The evaluation of usability was conducted by comparing user interfaces' valuable features, including instruction and shortcuts, projective views, editable in projective views, fast annotation feature, timer, and main view focus mode.

### 4.2.1 Instruction and Shortcuts

Table 4.1: Instruction and shortcuts of user interface.

| User interface | Instruction and Shortcuts |
|---|---|
| POINTS | ✔ |
| 3D BAT | ✘ |
| LATTE | ✘ |


Figure 4.1: Tabs for user interface.

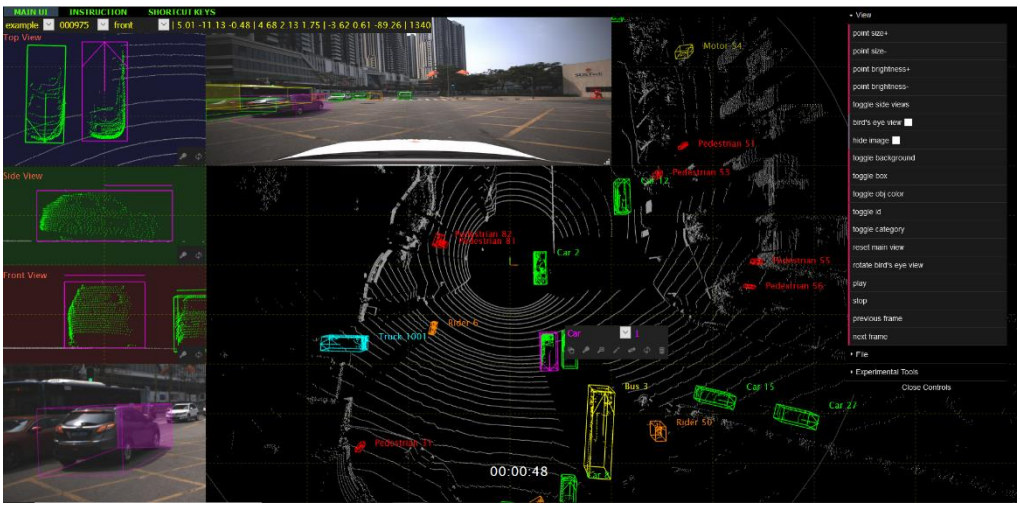


Figure 4.2: Tab contents of main UI tab.

According to the 3D point cloud labelling tool, Amazon Sagemaker Ground Truth in the marketplace (AWS, n.d.), it consists of instructions and shortcuts tabs. In the beginning, there is no instruction and shortcuts embedded in POINTS. There is a problem faced if asking an inexperienced user to make a label to point clouds intuitively in the user interface. As an example, if the user

interface is set up by person A and person B was the user who asked to label the point clouds. Person B may have no idea how to operate without guidance. Therefore, instructions and shortcuts features were added in POINTS. Three tabs in horizontal were added in POINTS.

Figure 4.1 shows three tabs, including main UI, instruction, and shortcut keys. The main UI tab was set to default open. Figure 4.2 shows the contents of the main UI, which consists of point clouds and images with various features for the user to label the point clouds. The tab contents of instruction and shortcut keys are new features added. Figure 4.3 shows the tab contents of the instruction tab. In the instruction tab, there is a video showing the steps to labels an object in point clouds. The instruction video is teaching the labelling steps in both perspective view and projective views. Therefore, users can refer to the instruction video as a guidance without online searching for teaching instruction from the developer.


Figure 4.3: Tab contents of instruction tab.


Figure 4.4: The tab contents of shortcut keys tab.

In addition, the last tab is shortcuts. According to Figure 4.4, this tab consists of two tables showing the keyboard and mouse functions for both main view and projective view. Figure 4.5 and Figure 4.6 below shows the clear version of tables of keyboard shortcuts and mouse functions for the perspective view and projective views in shortcuts tab.

With the aid of both instruction video and shortcut lists, the user will be more understanding the user interface's operation. The user can refer to the video and tables in this user interface when forget some of the user interface functions to operate. Among all three user interfaces, POINTS is the only one that provides an instructional video and the list of shortcuts. The instructional video and the list of shortcuts will provide enough information for the user to make labels on the point clouds and further reduce the time to master the user interface's operation.

| Keys | Description |
|---|---|
| Mouse scroll up/down | Zoom in/out |
| Mouse left key hold & move | Rotate (change main view) |
| Mouse right key hold & move | Pan |
| Left click on box | Select |
| Left click on a selected box | Show transform control |
| Left click on non-box area | Hide transform control if present/ unselect box |
| Ctrl+mouse drag | Add a new box |
| -/= | Adjust point size |
| V | Switch transform modes among resize/translate/rotate |
| Z/X/C | Turn on/off X/Y/Z axis |
| Ctrl+S | Save current frame |
| Ctrl+D/del | Remove selected box |
| 1,2 | Select previous/next box |
| 3,4 | Show previous/next frame in current scene |
| 5,6,7 | Show camera helper box of sideviews |
| Space | Pause/Continue stream play |

Figure 4.5: Keyboard shortcuts and mouse function for perspective view in shortcuts tab.

| Keys | Description |
|---|---|
| A | Move box left |
| S | Move box down |
| D | Move box right |
| W | Move box up |
| Q | Rotate box counterclockwise |
| E | Rotate box clockwise |
| R | Rotate box counterclockwise with box auto-fitting |
| F | Rotate box clockwise with box auto-fitting |
| G | Reverse heading direction (rotate by PI) |
| T | Reset box |
| Double click on center | Auto-shrink box by adjusting all borders to nearest inner point |
| Double click on border | Auto-shrink by adjusting the borders to nearest inner point |
| Double click on corner | Auto-shrink box by adjusting the corresponding borders to nearest inner point |
| Drag border/corner /center | Move border/corner/box |
| Ctrl+Drag border/corner | Move border/corner/box with box auto-fitting |

Figure 4.6: Keyboard shortcuts and mouse function for projective views in shortcuts tab.

### 4.2.2 Projective Views of User Interface

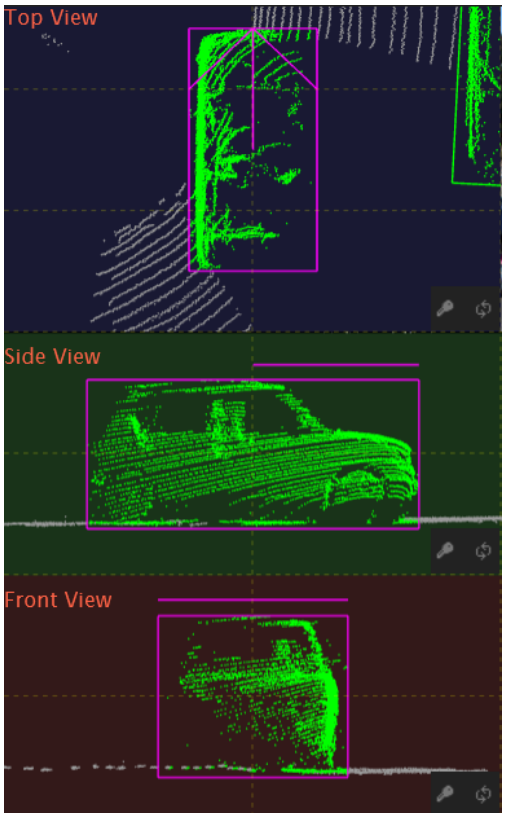


Figure 4.7: Projective view of POINTS.

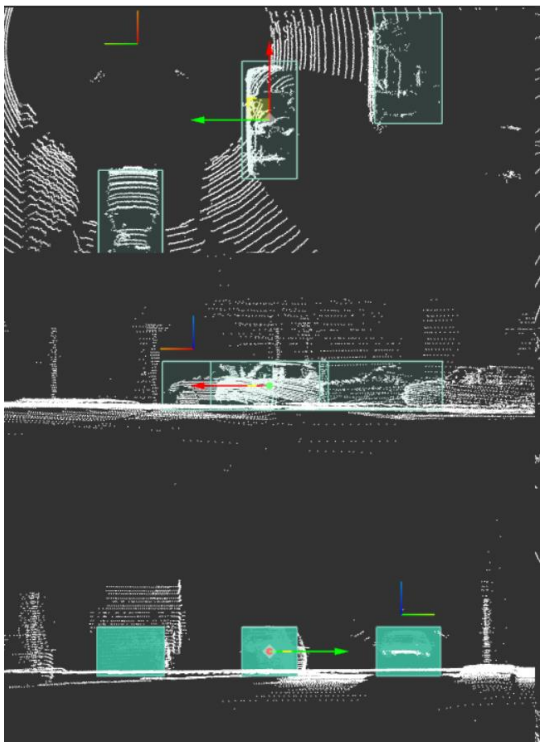

Figure 4.8: Projective views of 3D BATS.

Table 4.2: Projective views of user interface.

| User interface | Projective view | Editable in projective view |
|---|---|---|
| POINTS | ✔ | ✔ |
| 3D BAT | ✔ | ✘ |
| LATTE | ✘ | ✘ |

According to the 3D point cloud labelling tools in the marketplace, Amazon Sagemaker Ground Truth, Playment.io (Playment, n.d.), and Supervise.ly (SUPERVISELY, n.d.), the user interfaces consist of projective views. When the bounding box is created, the projective views, including the top, front and side views, are displayed. Besides, the 3D point cloud labelling tools in the marketplace enables the user to edit in projective views as well.

Figure 4.7 and Figure 4.8 show the projective views of POINTS and 3D BAT, respectively. According to Table 4.2, among the three user interfaces, POINTS and 3D BATS consist of projective views to show the bounding box's top, side, and front views, whereas LATTE does not have the projective views. The projective views of POINTS are much clearer than the projective views of 3D BAT. This is because the projective views of POINTS are focused on the selected object. The projective views of 3D BAT are quite difficult to see as there are several bounding boxes displayed in the projective views. With projective views, the bounding box with point clouds is clearly displayed in three views, and it will ease the inspection of the labelling result. The changes in perspective view and projective views are simultaneously. In contrast, without projective views, the user needs to rotate the perspective view to inspect and make sure a bounding box had enclosed all the object' points. This will increase the labelling time and become inefficient.

Among the three open-source user interfaces, POINTS is the only one enables the user to edit and adjust the bounding box in projective views. It is convenient to adjust the box in projective views as the user no need to change the viewpoint in perspective view from time to time. Editing in projective views is much easier than editing in perspective view, as the projective views' adjustment is based on two dimensions view. The height, width, and length of the bounding box can be adjusted in the projective views accordingly. Thus,

this feature will help the user complete the labels effectively as the bounding box's adjustment in projective views is relatively simple.

### 4.2.3 Fast Annotation Feature

Table 4.3: Fast annotation feature of user interfaces.

| User interface | Fast annotation feature |
|---|---|
| POINTS | 3D interactive box fitting algorithm |
| 3D BAT | ✕ |
| LATTE | One click annotation |

According to Table 4.3, 3D interactive box fitting algorithm and one-click annotation are applied in POINTS and LATTE, respectively, whereas 3D BAT does not have any fast annotation feature. The function of the fast annotation feature of POINTS and LATTE is different.

The interactive box fitting algorithm was described below. The inputs of interactive box fitting algorithm:

$$P \in R^{n\times 3}$$
$$b = (p, s, r)$$
$$o = (r', s')$$
$$p, s, r \in R^3$$

where

$P = \ point\ cloud$

$b = \ original\ box$

$o = \ operation$

$p, s, r = \ position, scale, and\ rotation, respectively$

$r', s' = \ rotation\ angles\ changed\ and\ scale\ change, respectively$

Firstly, all of the points, $R$, which inside $b$ will be found. Then, a coordinate system, $B'$, is built out with original $P$ and $r + r'$, and a coordinate range, $c$, which is represented x, y, and z coordinate ranges is composed by applying $s'$on $s$, as the operation had been carried out. A new point cloud, $P'$, was computed to represent the point cloud, $P$, in the coordinate system $B'$. In order

to find $R'$, two conditions will be tested. If the points in $P'$ consist of the original points, $R$, the points are added to $R'$. Besides, the points will be added to $R'$ as well, as the points in $P'$ are within the coordinate range $c$. Then, the coordinate range, $d$, of $R'$ can be computed to find the maximum and minimum values of x, y, and z coordinate axis. After that, the position, $\delta p$ and scale, $s''$ can be computed in coordinate system, $B'$, according to coordinate range, $d$. $\delta p'$ will be figured out to represent the $\delta p$ of original coordinate system. Thus, the new box, $b'$ obtained is:

$$b' = p + \delta p', s'', r + r'$$

For the 3D interactive box fitting algorithm, the user needs to drag a rectangular to create a bounding box on an object in point clouds. The created bounding box should be slightly larger than the object. Then, it will automatically recognize the object in the bounding box and shrink the bounding box to fit the object. Besides, it will automatically assign the desired classes for the labelled objects by changing the colour of the bounding box. This depends on the dimensions of the objects.

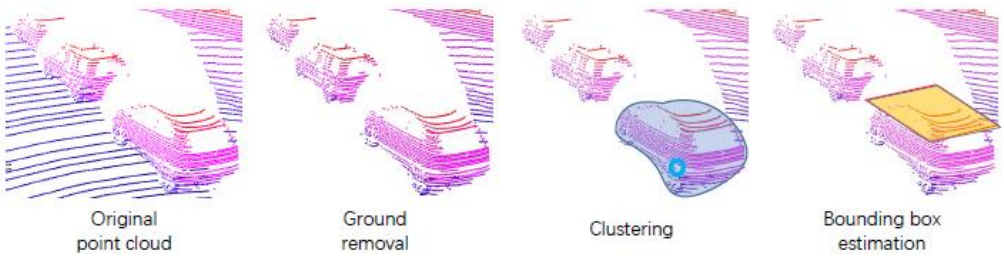


Figure 4.9: Pipeline of one-click annotation feature (Wang et al., 2019).

Figure 4.9 shows the pipeline of one-click annotation. For one-click annotation, three steps are required, including removing ground, clustering the points of an object, and estimating bounding box. Ground removal removes the noise in point clouds. After the ground removal process, a clustering algorithm is used to find the cluster. The clustering algorithm is based on DBSCAN. When a human clicks a point, the algorithm will find the nearby cluster to the point. The cluster is considered as an object. As the clustering

algorithm finds a cluster of the point, a 2D bounding box is estimated and displayed on the object's top. The bounding box estimation is based on the search-based rectangular fitting algorithm. The user only needs to right-click on the object's point in point clouds without dragging a box. After right-click the points, a two-dimension rectangular will be automatically displayed on the top of the object. The bounding box in LATTE is two dimensions without the thickness of the box.

Theoretically, one-click annotation is faster 3D interactive box fitting algorithm, as it just needs one click only on a point of an object. However, it is very dependent on the CPU and GPU of the computer. When the labelling process is carried out in POINTS, CPU and GPU usage is not more than 10 %, and the bounding box will come out immediately. When one click annotation is used in LATTE, CPU and GPU usage is around 45 % and 60 %, respectively. There is a delay of about 10 seconds to display the bounding box. Therefore, the one-click annotation may cause the user interface to crash if the specification of CPU and GPU of the computer is too low. Both features will add benefit for users to complete the labelling in a shorter time. However, the required specifications of computer for POINTS are low, and the performance of 3D interactive fitting box is fast and smooth.

### 4.2.4 Timer

Table 4.4: Timer of user interfaces.

| User interfaces | Timer |
|---|---|
| POINTS | ✔ |
| 3D BAT | ✔ |
| LATTE | ✘ |

According to Table 4.4, an embedded timer is present in POINTS and 3D BAT. For 3D BAT and POINTS, a timer will start to count when the user interface is opened. Both timers will count the time in terms of hours, minutes, and seconds. It is pretty convenient for the user to record labelling time as the external timer is unnecessary. For example, the point clouds contain many objects, and users spend much time completing the labels. However, the user

forgot to record the time for the whole labelling process. It is pretty time-consuming and wasting of time to label all the objects again and record the time. With the internal timer in the user interface, this kind of human error can be neglected as time had been recorded. The timer is much more convenient for researchers or developers who wish to figure out the user interface's labelling speed.

### 4.2.5 Main View Focus Mode

Table 4.5: Main view focus mode of user interfaces.

| User interfaces | Main view focus mode |
|---|---|
| POINTS | ✔ |
| 3D BAT | ✘ |
| LATTE | ✘ |

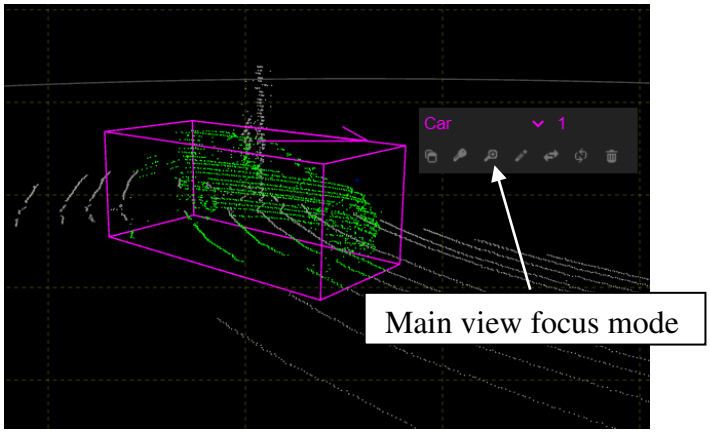


Figure 4.10: Main view focus mode in POINTS.

According to Table 4.5, POINTS has a main view focus mode, whereas 3D BAT and LATTE do not have a main view focus mode. Main view focus mode functions to focus on the selected bounding box. Figure 4.10 shows a car with a bounding box in focus mode. The main view focus mode can be activated in the fast-floating toolbox as a bounding box is selected. When the focus mode is turned on, only the points enclosed by the bounding box will be automatically displayed at the centre of the main perspective view. The user interface will zoom in on the selected object with most of the background hidden. The object colouring feature of POINTS will colour the points in a

bounding box. With the combination of main view focus mode, it allows the user to clearly inspect the label. The user can check the details of the selected object. This feature will ease the inspection of errors, which the user can check the labels of objects one by one. Thus, the accuracy of the labels in POINTS will increase. Without this feature, the user needs to change the viewpoints in perspective view from time to time to inspect the labels' quality. It is pretty time-consuming, and the user may be lost in the 3D environment as it needs to rotate the perspective views to inspect.

## 4.3 Evaluation of Fast Annotation Feature in terms of Time and Error

Table 4.6: The time used and errors to label all the 12 objects of KITTI datasets for POINTS and LATTE using fast annotation feature.

| User Interface | Time for all labels | Average time for one object (s) | Error |
|---|---|---|---|
| POINTS | 1min 17s | 6.42 | 2 |
| LATTE | 1min 23s | 6.92 | 5 |

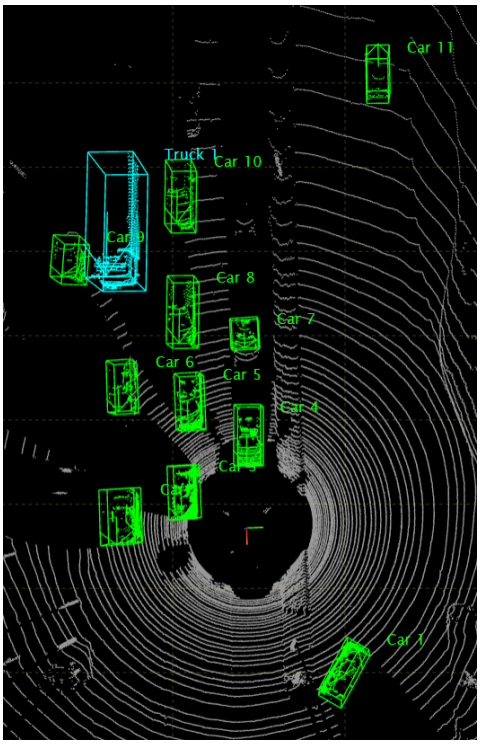


Figure 4.11: Labelling using 3D interactive box fitting algorithm without adjustment of boxes.

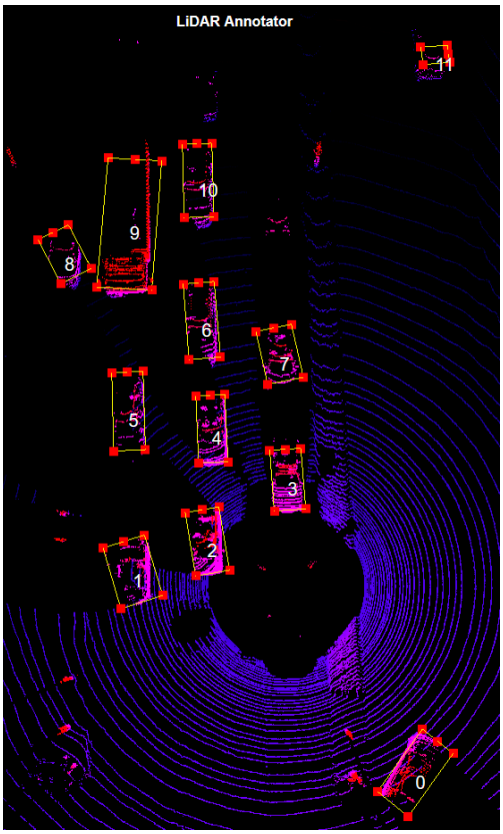


Figure 4.12: Labelling using one-click annotation without adjustment of boxes.

The comparison was made between POINTS and LATTE only, as 3D BAT does not have the fast annotation feature. The point cloud scene used is taken from KITTI datasets. The objective, evaluation of labelling speed of user interfaces, can be achieved by comparing the labelling time to finish all the labels in the point cloud scene. According to Table 4.6, the total time used to label all the objects in the point cloud for POINTS and LATTE is 1 min 17 s and 1 min 23 s, respectively. The average time for each label on an object of POINTS and LATTE is 6.42 s and 6.92 s, respectively. Thus, the annotation speed of POINTS is slightly faster than LATTE, which the average time for each label is 0.5 s faster. All of the labels are finished by one operation count without adjustment. Theoretically, the annotation speed of one-click annotation should be faster than the 3D interactive box fitting algorithm. The labelling time for each object of LATTE is 2.55 s (Wang, 2019). However, the laptop's CPU and GPU used to run the user interface are not strong enough, so there are some delays for the one-click annotation.

On the other hand, according to Figure 4.11 and Figure 4.12, there are two major errors in POINTS and five major errors in LATTE. The errors of POINTS are occurred on car 7 and car 9 in Figure 4.11, whereas the errors of LATTE have occurred on car 1, car 7, car 8, truck 9, and car 11 in Figure 4.12. The errors in POINTS are occurred due to the Lidar point cloud is too sparse, so the algorithm cannot recognize all the points enclosed in the objects, which results in a short length of bounding boxes for both car 7 and car 9. The errors of POINTS are short of length only, whereas most LATTE errors are the labels slightly crooked. In short, the 3D interactive box fitting of POINTS is faster and more accurate compared to the one-click annotation of LATTE.

## 4.4 Evaluation of Annotation Efficiency of Three User Interfaces in terms of Time

Table 4.7: Total time to finish all labels and average time for each object of three user interfaces.

| User Interface | Time to finish all labels | Average time for one object (s) |
|---|---|---|
| POINTS | 2mins 8s | 10.66 |
| 3D BAT | 4mins 18s | 21.5 |
| LATTE | 2mins 42s | 13.5 |

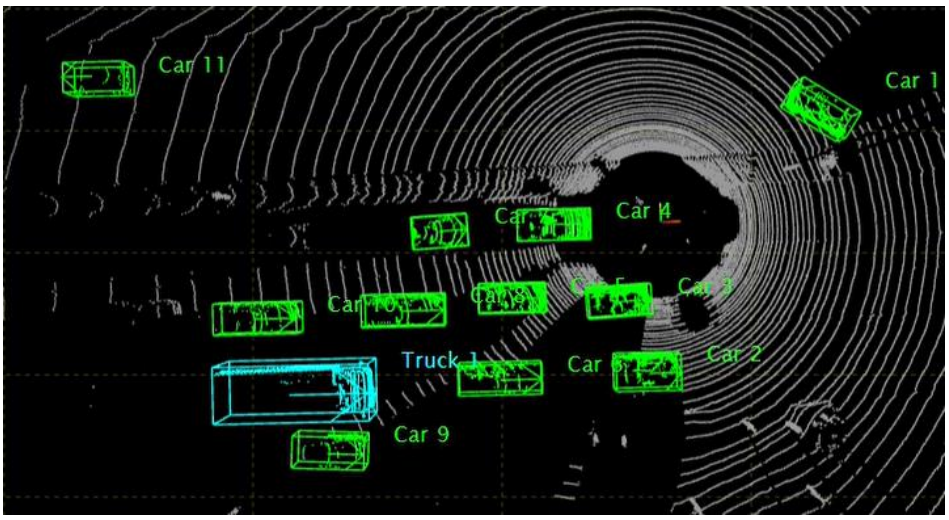


Figure 4.13: Final labelling results of POINTS.

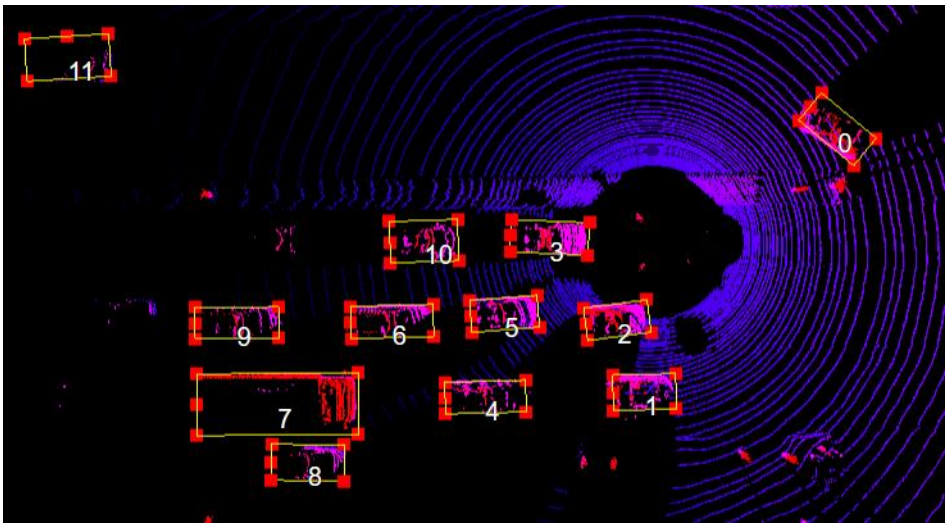


Figure 4.14: Final labelling results of LATTE.

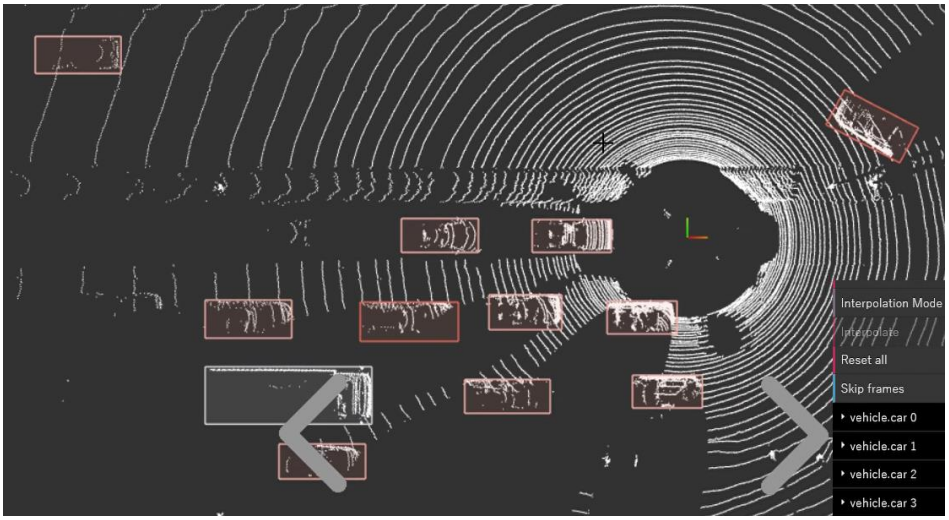


Figure 4.15: Final labelling results of 3D BAT.

The results obtained are the total time used to label all the objects in the point cloud scene. The objective of this project, evaluation of labelling speed between user interfaces, can be achieved by comparing the labelling time for each user interface to finish all the labels. The point cloud scene used for the comparison comes from KITTI datasets. This point cloud is a LiDAR point cloud and consisted of 12 objects. The labelling process of POINTS and 3D BAT were recorded by using screen recording of a computer. Screen recording is not suitable for LATTE as it consumes many CPU and GPU spaces, where LATTE cannot work simultaneously with screen recording. The point cloud labelling of POINTS and LATTE was done with the aid of fast annotation feature. The errors were adjusted until the labels are just right, which were

compared to ground truth. The evaluation metric for 3D point cloud labelling tools is difficult to evaluate, since it does not have a standard benchmark metrics (Li et al., 2020). Basically, the evaluation of accuracy is based on IoU, FP ratio and FN ratio. IoU is the intersection over union of the labels from users and the ground truth. FP ratio is the ratio of points not belong to the objects being labelled and total points of the objects. In contrast, FN ratio is the ratio of points belong to the objects not being labels and total points of the objects. Figure 4.13, Figure 4.14 and Figure 4.15 show the final labelling results for POINTS, LATTE and 3D BAT, respectively. In this project, all of the labels for three user interfaces are considered true compared to the ground truth.

According to Table 4.7, the total time used for POINTS, 3D BAT, and LATTE to label all the objects is 2 mins 8 s, 4 mins 18 s, and 2 mins 42 s, respectively. The average time used to label one object in the point cloud is 10.66 s, 21.5 s, and 13.5 s, respectively. Thus, the labelling speed of POINTS, LATTE, and 3D BAT is fastest to slow according to the sequences. The main reason why the labelling speed of 3D BAT is slow is that 3D BAT does not have a fast annotation feature. Besides, editing a 3D bounding box in perspective view is pretty difficult. The rotation and translation of the bounding box should be done by the combination of keyboard and mouse. For LATTE, labelling speed is shorter than 3D BAT. The main reason for LATTE leads 3D BAT is because of the one-click annotation feature. The bounding box is two dimensions, which is much easier to adjust compared to 3D BAT. However, the labelling speed for LATTE is slower than POINTS. 3D interactive box fitting algorithm reduces much time to adjust the bounding box. As both POINTS and LATTE have fast annotation features, the labelling speed is shorter than 3D BAT. POINTS is the fastest because the bounding box can be adjusted in projective views, whereas the adjustment of the bounding box in LATTE should be made in the perspective view.

## 4.5 Summary

3D point cloud labelling is valuable and important. However, the convenient user interfaces for 3D point cloud labelling in the marketplace are pretty expensive. A comparison was conducted between open-source labelling tools

to select the most convenient user interface, which is free of charge and easy to use to overcome the problem. All objectives had been fulfilled as the comparison between open-source user interfaces, and evaluation of usability and labelling speed had been done.

Table 4.8: The availability of useful features for three user interfaces.

| | POINTS | 3D BAT | LATTE |
|---|---|---|---|
| Instruction and shortcut | ✔ | ✘ | ✘ |
| Projective views (Top, Side, Front) | ✔ | ✔ | ✘ |
| Editable in projective views | ✔ | ✘ | ✘ |
| Fast annotation feature | 3D Interactive box fitting | ✘ | One-click annotation |
| Timer | ✔ | ✔ | ✘ |
| Main view focus mode | ✔ | ✘ | ✘ |

Among the three user interfaces, the performance of POINTS is the best. Table 4.8 shows that POINTS consists of all the listed useful features. The functions of instruction and shortcuts, projective views, editable in projective views, fast annotation feature, timer, and main view focus mode are beneficial for users to master the user interface's operation quickly, and convenient to use as the operation is pretty simple. The features aid the inspection of labels' as well.

Besides, the labelling speed was evaluated in terms of labelling time. The average labelling time of one object without adjustment using fast annotation features of POINTS and LATTE is 6.42 s and 6.92 s, respectively. The average labelling time of one object for POINTS, 3D BAT, and LATTE is 10.66 s, 21.5 s, and 13.5 s, respectively. Among the three user interfaces, the labelling time for POINTS is the shortest. Thus, POINTS is the best open-source and accessible user interface among all three user interfaces. POINTS

is a powerful and useful user interface for labelling 3D point clouds, which can perform well as or better than the 3D labelling tools in the marketplace.

# CHAPTER 5

# CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Conclusions

A convenient user interface for 3D point cloud labelling means the user interface is easy to operate and the design of user interface is simple. The user interface allows user to traverse in 3D environment and make label in point cloud. Thus, the user interfaces, POINTS, 3D BAT, and LATTE, were successfully set up. PointAtMe unable to be set up is because the user interface is VR-based system, which needs to work with a VR device.

By setting up three open sources user interfaces, the comparison between POINTS, 3D BAT and LATTE was conducted. The three user interfaces' performances were evaluated in terms of usability and labelling speed. The usability is evaluated by comparing the useful features of user interfaces, which ease the operation of labelling and increase the labelling speed. With the aid of extra features, such as POINTS's 3D interactive box fitting algorithm and LATTE's one-click annotation, the labelling speed will improve a lot. According to the results, the extra features, including instruction and shortcuts, projective views, editable in projective views, fast annotation feature, timer, and main view focus mode, in POINTS allows user to understand the operation of user interface in short time and finish labelling all the labels in a shorter time compared to 3D BAT and POINTS. POINTS is a powerful and convenient user interface for labelling 3D point clouds. POINTS is able to perform well as or better than the 3D labelling tools in the marketplace.

## 5.2 Recommendations for Future Work

The scene of point cloud is 3D view. During the labelling process, users may need to zoom in and rotate the point clouds to label the objects. 3D environment contains more data compared to 2D images. For the inexperienced users, it is easy to get lost in 3D environment. They may not return back to original position. The only method can back to original position

is refreshing the web page. Thus, an algorithm is needed to restore the point cloud to the original position. The algorithm should obtain the original X, Y, and Z position of point clouds, which is the default position of the point clouds as the point clouds are selected. Then, a feature, such as button or symbol, can be added with the algorithm in POINTS. As the users click on the button or symbol, the scene of point clouds will return back to the original position.

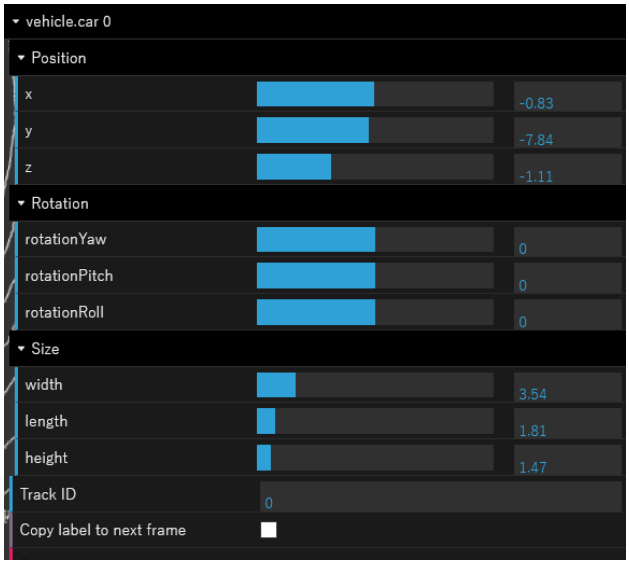


Figure 5.1: Example list for vehicle 0 in 3D BAT.

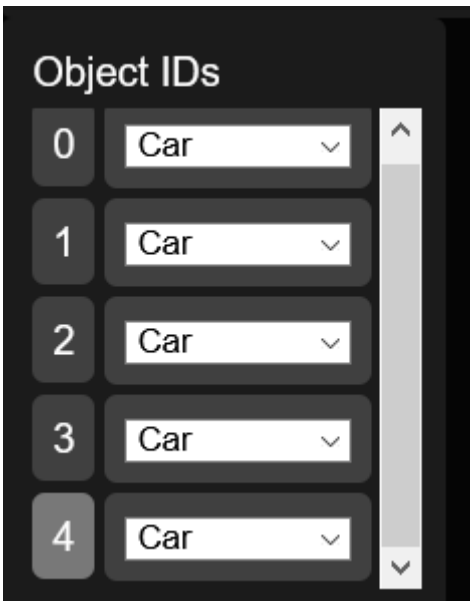


Figure 5.2: Example list for vehicle 0 to vehicle 4 in LATTE.

Figure 5.1 and Figure 5.2 show the example list of 3D BAT and LATTE. Besides, one feature from 3D BAT and LATTE can be added into POINTS. When the labels are made, there is a list shows the labels with name and detailed information. For 3D BAT, the lists show the objects' name and enable user to adjust the position and size of bounding box in the list. In contrast, for LATTE, the list shows the classes of objects and assigned with specific number. The modification can be made to combine both advantages. For example, when the user creates a bounding box on a car in point clouds and label the bounding box with car 1, the list will show the name, car 1 with the dimensions of bounding box, including length, width, and height of the car. The list shows the information only and the users are not allowed to change the information in the list. This is because POINTS provides a feature for users to assign the bounding box's number and the adjustment of bounding box can be made in perspective view and projective views. The function of list is to show the information of labelled objects. Besides, by clicking the name of car in the list, POINTS will automatically move to the position of the car. The users can observe and inspect the labelled object conveniently.

Lastly, more AI-based algorithms can be added to ease the operation of POINTS. 3D object tracking algorithm can be added to work with annotation transfer algorithm of POINTS to obtain high quality labels. Firstly, 3D object detector detects the oriented 3D bounding boxes from point clouds (Wang et al., 2020). Then, state estimation and similar feature matching will be done using 3D Kalman filter and reidentification, respectively. Lastly, Hungarian algorithms work to do data association. This 3D object tracking algorithm is based on deep learning. Thus, by taking the benefit of this algorithm and combining with annotation transfer algorithm of POINTS, high quality of labels will be obtained.

# REFERENCES

Aws, n.d. Amazon SageMaker Ground Truth Features. [online] Available at: <https://aws.amazon.com/sagemaker/groundtruth/features/> [Accessed 14 April 2021].

Babahajiani, P., Fan, L., Kämäräinen, J.K. and Gabbouj, M., 2017. Urban 3D segmentation and modelling from street view images and LiDAR point clouds. *Machine Vision and Applications*, 28(7), pp.679–694.

Bello, S.A., Yu, S., Wang, C., Adam, J.M. and Li, J., 2020. Review: Deep learning on 3D point clouds. *Remote Sensing*, 12(11), pp.1–35.

Boulch, A., Guerry, J., Le Saux, B. and Audebert, N., 2018. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers and Graphics (Pergamon)*, [online] 71, pp.189–198. Available at: <https://doi.org/10.1016/j.cag.2017.11.010>. [Accessed 8 July 2020].

Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. and Beijbom, O., 2020. nuScenes: A Multimodal Dataset for Autonomous Driving. (March), pp.11618–11628.

Daniel,S., n.d. Intelligent 3D world building from mobile terrestrial LiDAR point clouds. [online] Available at: <https://syldan.wordpress.com/research/intelligent-3d-world-building-from-mobile-terrestrial-lidar-point-clouds/> [Accessed 2 August 2020].

FME Community, 2020. What is a point cloud? What is LiDAR? [online] Available at: <https://community.safe.com/s/article/what-is-a-point-cloud-what-is-lidar> [Accessed 2 August 2020].

GitHub, 2019. nuScenes devkit. [online] Available at: <https://github.com/nutonomy/nuscenes-devkit> [Accessed 22 August 2020].

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. and Bennamoun, M., 2019. Deep Learning for 3D Point Clouds: A Survey. [online] pp.1–27. Available at: <http://arxiv.org/abs/1912.12033>. [Accessed 12 July 2020].

Hu, S.M., Cai, J.X. and Lai, Y.K., 2020. Semantic labeling and instance segmentation of 3d point clouds using patch context analysis and multiscale processing. *IEEE Transactions on Visualization and Computer Graphics*, 26(7), pp.2485–2498.

Jing, H. and You, S., 2016. Point cloud labeling using 3D Convolutional Neural Network. *Proceedings - International Conference on Pattern Recognition*, 0, pp.2670–2675.

Kim, D.I. and Sukhatme, G.S., 2014. Semantic labeling of 3D point clouds with object affordance for robot manipulation. *Proceedings - IEEE International Conference on Robotics and Automation*, pp.5578–5584.

Koh, J.H., 2018. Object detection with LiDAR point cloud algorithm. [online] Available at: <https://medium.com/@jhkoh/object-detection-with-lidar-point-cloud-algorithm-94a241fd3f49> [Accessed 2 August 2020]

Koppula, H.S., Anand, A., Joachims, T. and Saxena, A., 2011. Semantic labeling of 3D point clouds for indoor scenes, in Proc. Neural Inf. Process. Syst., pp. 244–252.

Li, E., Wang, S., Li, C., Li, D., Wu, X. and Hao, Q., 2020. SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv), pp.1108–1115.

National Ocean Service, 2020. What is lidar? [online] Available at: <https://oceanservice.noaa.gov/facts/lidar.html#:~:text=Lidar%2C%20which%20stands%20for%20Light,variable%20distances)%20to%20the%20Earth.> [Accessed 2 August 2020].

Qi, C.R., Su, H., Mo, K. and Guibas, L.J., 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January, pp.77–85.

Rehman, Y., Uddin, H.M.A., Siddique, T.H.M., Haris, Jafri, S.R.U.N. and Ahmed, A., 2019. Comparison of Camera and Laser Scanner based 3D Point Cloud. *2019 4th International Conference on Emerging Trends in Engineering, Sciences and Technology, ICEEST 2019*.

SUPERVISELY, n.d. 3D point cloud labelling. [online] Available at: <https://supervise.ly/lidar-3d-cloud> [Accessed 14 April 2021].

Visual Studio Code, n.d. Code editing. Redefined. [online] Available at: < https://code.visualstudio.com/#powerful-debugging> [Accessed 20 March 2021].

Wang, B., Wu, V., Wu, B. and Keutzer, K., 2019. LATTE : Accelerating LiDAR Point Cloud Annotation via Sensor Fusion , One-Click Annotation , and Tracking. 2.

Wang, D., Huang, C., Wang, Y., Deng, Y. and Li, H., 2020. A 3D Multiobject Tracking Algorithm of Point Cloud Based on Deep Learning. *Mathematical Problems in Engineering*, 2020.

Wang, Y., Ji, R. and Chang, S.F., 2013. Label propagation from imagenet to 3D point clouds. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.3135–3142.

Wirth, F., Quchl, J., Ota, J. and Stiller, C., 2019. PointAtMe: Efficient 3D point cloud labeling in virtual reality. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019-June(Iv), pp.1693–1698.

Zimmer, W., Rangesh, A. and Trivedi, M., 2019. 3D BAT: A Semi-Automatic, Web-based 3D annotation toolbox for full-surround, multi-modal data streams. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019-June, pp.1816–1821

# APPENDICES

## APPENDIX A: Coding

```
<div class="tab">
    <button class="tablinks" onclick="openTab(event, 'UserInterface')" id="defaultOpen"><b>Main UI</b></button>
    <button class="tablinks" onclick="openTab(event, 'Instruction')"><b>Instruction</b></button>
    <button class="tablinks" onclick="openTab(event, 'Shortcut')"><b>Shortcuts</b></button>
</div>
```

Figure A-1.1: HTML code for tab.

```
function openTab(evt, cityName) {
  var i, tabcontent, tablinks;
  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }
  document.getElementById(cityName).style.display = "block";
  evt.currentTarget.className += " active";
}
document.getElementById("defaultOpen").click();

var timerVar = setInterval(countTimer, 1000);
var totalSeconds = 0;
```

Figure A-1.2: JavaScript code for tab.

```
<div id="Instruction" class="tabcontent">
    <iframe width="1000" height="650" src="https://www.youtube.com/embed/xQp5C7-dWek"
    title="YouTube video player" frameborder="0"
    allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
    allowfullscreen></iframe>
</div>
```

Figure A-1.3: HTML code for instruction tab contents.

```
<div id="Shortcut" class="tabcontent" >
    <div id="wrapper">
      <div class="content-area">
        <div class="container-fluid">
          <div class="main">
            <div class="row">
              <div class="column">
                <div class="col-md-7">
                  <table class="blueTable">
                    <thead>
                        <tr><b style="color: rgb(255, 255, 255);">Perspective View</b></tr>
                    </thead>
                    <thead>
                      <tr>
                        <th>Keys</th>
                        <th>Description</th>
                      </tr>
                    </thead>
                    <tbody>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Mouse scroll up/down</b>
                        </td>
                        <td>
                          <b style="color: rgb(0, 0, 0);">Zoom in/out</b>
                        </td>
                      </tr>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Mouse left key hold & move</b>
                        </td>
                        <td>
                          <b style="color: rgb(0, 0, 0);">Rotate (change main view)</b>
                        </td>
                      </tr>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Mouse right key hold & move</b>
                        </td>
                        <td>
                          <b style="color: rgb(0, 0, 0);">Pan</b>
                        </td>
                      </tr>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Left click on box</b>
                        </td>
                        <td>
                          <b style="color: rgb(0, 0, 0);">Select</b>
                        </td>
                      </tr>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Left click on a selected box</b>
                        </td>
                        <td>
                          <b style="color: rgb(0, 0, 0);">Show transform control</b>
                        </td>
                      </tr>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Left click on non-box area</b>
                        </td>
                        <td>
                          <b style="color: rgb(0, 0, 0);">Hide transform control if present/ unselect box</b>
                        </td>
                      </tr>
                      <tr>
                        <td>
                          <b style="color: rgb(43, 116, 21);">Ctrl+mouse drag</b>
                        </td>
                        <td>
```

Figure A-1.4: HTML code for shortcuts tab contents (1).

```
                    <b style="color: rgb(0, 0, 0);">Add a new box</b>
                  </td>
                </tr>
                <tr>
                  <td>
                    <b style="color: rgb(43, 116, 21);">-/=</b>
                  </td>
                  <td>
                    <b style="color: rgb(0, 0, 0);">Adjust point size</b>
                  </td>
                </tr>
                <tr>
                  <td>
                    <b style="color: rgb(43, 116, 21);">V</b>
                  </td>
                  <td>
                    <b style="color: rgb(0, 0, 0);">Switch transform modes among resize/translate/rotate</b>
                  </td>
                </tr>
                <tr>
                  <td>
                    <b style="color: rgb(43, 116, 21);">Z/X/C</b>
                  </td>
                  <td>
                    <b style="color: rgb(0, 0, 0);">Turn on/off X/Y/Z axis</b>
                  </td>
                </tr>
                <tr>
                  <td>
                    <b style="color: rgb(43, 116, 21);">Ctrl+S</b>
                  </td>
                  <td>
                    <b style="color: rgb(0, 0, 0);">Save current frame</b>
                  </td>
                </tr>
                <tr>
                    <td>
                      <b style="color: rgb(43, 116, 21);">Ctrl+D/del</b>
                    </td>
                    <td>
                      <b style="color: rgb(0, 0, 0);">Remove selected box</b>
                    </td>
                  </tr>
                  <tr>
                    <td>
                      <b style="color: rgb(43, 116, 21);">1,2</b>
                    </td>
                    <td>
                      <b style="color: rgb(0, 0, 0);">Select previous/next box</b>
                    </td>
                  </tr>
                  <tr>
                    <td>
                      <b style="color: rgb(43, 116, 21);">3,4</b>
                    </td>
                    <td>
                      <b style="color: rgb(0, 0, 0);">Show previous/next frame in current scene</b>
                    </td>
                  </tr>
                  <tr>
                    <td>
                      <b style="color: rgb(43, 116, 21);">5,6,7</b>
                    </td>
                    <td>
                      <b style="color: rgb(0, 0, 0);">Show camera helper box of sideviews</b>
                    </td>
                  </tr>
                  <tr>
                    <td>
                      <b style="color: rgb(43, 116, 21);">Space</b>
                    </td>
                    <td>
                      <b style="color: rgb(0, 0, 0);">Pause/Continue stream play</b>
                    </td>
                  </tr>
                </tbody>
              </table>
            </div>
          </div>
```

Figure A-1.5: HTML code for shortcuts tab contents (2).

```
24      <div id="timer-count">
25          <p id = "timer"></p>
26      </div>
661     var timerVar = setInterval(countTimer, 1000);
662     var totalSeconds = 0;
663
664     function countTimer() {
665           ++totalSeconds;
666           var hour = Math.floor(totalSeconds /3600);
667           var minute = Math.floor((totalSeconds - hour*3600)/60);
668           var seconds = totalSeconds - (hour*3600 + minute*60);
669               if(hour < 10)
670                   hour = "0"+hour;
671               if(minute < 10)
672                   minute = "0"+minute;
673               if(seconds < 10)
674                   seconds = "0"+seconds;
675                   document.getElementById("timer").innerHTML = hour + ":" + minute + ":" + seconds;
676           }
```

Figure A-1.6: HTML and JavaScript code for timer.

```
428   /* Style the tab */
429   .tab {
430       overflow: hidden;
431       border: 1px solid rgb(0, 0, 0);
432       background-color: #000000;
433     }
434
435     /* Style the buttons inside the tab */
436     .tab button {
437       background-color: inherit;
438       float: left;
439       border: none;
440       outline: none;
441       cursor: pointer;
442       padding: 0px 16px;
443       transition: 0.3s;
444       font-size: 12px;
445       color: rgb(78, 253, 9);
446     }
447
448     /* Change background color of buttons on hover */
449     .tab button:hover {
450       background-color: #345;
451     }
452
453     /* Create an active/current tablink class */
454     .tab button.active {
455       background-color: #345;
456     }
457
458     /* Style the tab content */
459     .tabcontent {
460       display: none;
461       position: absolute;
462       border: 0px solid #ccc;
463       border-top: none;
464     }
```

Figure A-1.7: CSS code for tab.

```css
table.blueTable {
  font-family: "Times New Roman", Times, serif;
  border: 5px solid #0a0891;
  background-color: #D2D2D2;
  width: 100%;
  text-align: left;
  border-collapse: collapse;
}

table.blueTable td,
table.blueTable th {
  border: 1px solid #160091;
  padding: 3px 2px;
}

table.blueTable tbody td {
  font-size: 15px;
  font-weight: bold;
}

table.blueTable tr:nth-child(even) {
  background: #E7E7E7;
}

table.blueTable thead {
  background: #1dafc9;
  background: -moz-linear-gradient(top, #55bcd6 0%, #33afce 66%, #1db5c9 100%);
  background: -webkit-linear-gradient(top, #55bcd6 0%, #33afce 66%, #1db5c9 100%);
  background: linear-gradient(to bottom, #55bcd6 0%, #33afce 66%, #1db5c9 100%);
  border-bottom: 2px solid #444444;
}

table.blueTable thead th {
  font-size:20px;
  font-weight: bold;
  color: #242BA6;
  text-align: center;
  border-left: 1px solid #003780;
}

table.blueTable thead th:first-child {
  border-left: none;
}

table.blueTable tfoot td {
  font-size: 14px;
}
```

Figure A-1.8: CSS code for the tables in shortcuts tab.

```css
/* Style the timer*/
#timer-count {
    position: absolute;
    resize:both;
    overflow:hidden;
    top: 90%;
    left: 48%;
    width: 40%;
    height: 30%;
    padding: 0px;
    margin: 0px;
}
```

Figure A-1.9: CSS code for timer.

```python
import numpy as np
import open3d as o3d
import struct

size_float = 4
list_pcd = []
with open ("0000000004.bin", "rb") as f:
    byte = f.read(size_float*4)
    while byte:
        x,y,z,intensity = struct.unpack("ffff", byte)
        list_pcd.append([x, y, z])
        byte = f.read(size_float*4)
np_pcd = np.asarray(list_pcd)
pcd = o3d.geometry.PointCloud()
v3d = o3d.utility.Vector3dVector
pcd.points = v3d(np_pcd)
o3d.io.write_point_cloud("000000.pcd", pcd)
```

Figure A-1.10: Python code for conversion of BIN file format to PCD file format.