UC Riverside
UC Riverside Electronic Theses and Dissertations

Title
A Novel Method for Classifying Pedestrians & E-Scooter Users in Roadside Point Cloud Data

Permalink
https://escholarship.org/uc/item/5fx5r3v5

Author
Gunasekar, Joy Mathew

Publication Date
2024

Copyright Information
This work is made available under the terms of a Creative Commons Attribution-ShareAlike License, available at https://creativecommons.org/licenses/by-sa/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

A Novel Method for Classifying Pedestrians & E-Scooter Users in Roadside Point Cloud Data

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Joy Mathew Gunasekar

September 2024

Thesis Committee:
Dr. Guoyuan Wu, Chairperson
Dr. Matthew Barth
Dr. Jiachen Li

Copyright by
Joy Mathew Gunasekar
2024

The Thesis of Joy Mathew Gunasekar is approved:

Committee Chairperson

University of California, Riverside

ABSTRACT OF THE THESIS

A Novel Method for Classifying Pedestrians & E-Scooter Users in Roadside Point Cloud Data

by

Joy Mathew Gunasekar

Master of Science, Graduate Program in Electrical Engineering
University of California, Riverside, September 2024
Dr. Guoyuan Wu, Chairperson

In recent years, riding e-scooters as a hobby has evolved into one of the popular forms of transportation in our city traffic. As it is becoming increasingly popular, micromobility-related accidents have also increased. To reduce such casualties, this thesis proposes a novel methodology to differentiate between e-scooter riders and pedestrians in urban environments. The Intelligent Transportation Systems Joint Program Office (ITS JPO) team that plans out safety features for city traffic will be able to utilize this algorithm to collect data on e-scooter user's frequency in general traffic. This data will allow city planners to plan out special road strips in places that have registered high e-scooter usage, thus enabling safer commutes for e-scooter riders. Although research on Vulnerable Road Users has increased in the past years, there has been very little research done to ensure the safety of e-scooter users. This work is among the first few to present a perception solution that works on point cloud data. The novelty lies in harnessing “Elasticity” and “Spatial” data from the actors

and using them as features to train ensemble learning models like Random Forest, Gradient Boost, and XG Boost. Both the terms “Elasticity” and “Spatial” are well explained in the methodology chapter of the thesis. All the tests were conducted in a private dataset recorded during the peak rush hours on the college campus.

Contents

List of Figures

List of Tables

Chapter 1 Overview

1.1 Introduction

On 13th December 2023, the World Health Organization published that approximately 1.9 million people lose their lives every year because of accidents in road traffic. Over half of those people are classified as vulnerable road users. Twenty-three percent of the tragic sum are listed as pedestrians. Drivers of micromobility devices like e-scooters make up 3% and cyclists come up to 6%. Two or three-wheeled vehicles account for 21% [1]. Technological improvement has made micro-mobility devices such as electric scooters more reliable and widely used. In the state of California, electric scooters are recognized as equals to motor vehicles and are required to follow motor vehicle rules (CVC §21221) [125]. This means they are not allowed to drive on sidewalks (CVC §21235(g)) [125]. However, this is rarely followed, especially around college campuses, thus putting pedestrians at risk. This is because rough road surfaces are unideal for riding e-scooters. Also, if a vehicle speeds beside an e-scooter rider it can knock them off balance. Thus, in reality, it is riskier to ride with the general traffic. The first step to solving this problem is designing special road strips that would allow for safer e-scooter travel. Designing a perception algorithm that can differentiate between pedestrians and micromobility users is probably the first step to solving this issue. As the data obtained from the sensors will give city planners a broader picture of e-scooter traffic in specific regions. This thesis aims to solve this perception problem with point cloud data obtained from roadside LiDARs.

1.2 Motivation

1.2.1 Vulnerable Road User Safety

Road users that are not covered by an outside shield are referred to as Vulnerable Road Users [145]. Thus pedestrians, cyclists, motorcyclists, scooterists, wheelchair users, skateboarders, e-scooterists, and all similar users fall under this category. As mentioned previously, VRU users approximately account for 1.9 million road accident casualties. Three percent of the total accounts for micromobility-related accidents. The number is shocking because e-scooters are not yet widely adopted. As more people decide to utilize micromobility devices like e-scooters the number of accidents is only expected to rise. Below attached is a graph that shows the public perception of e-scooters.

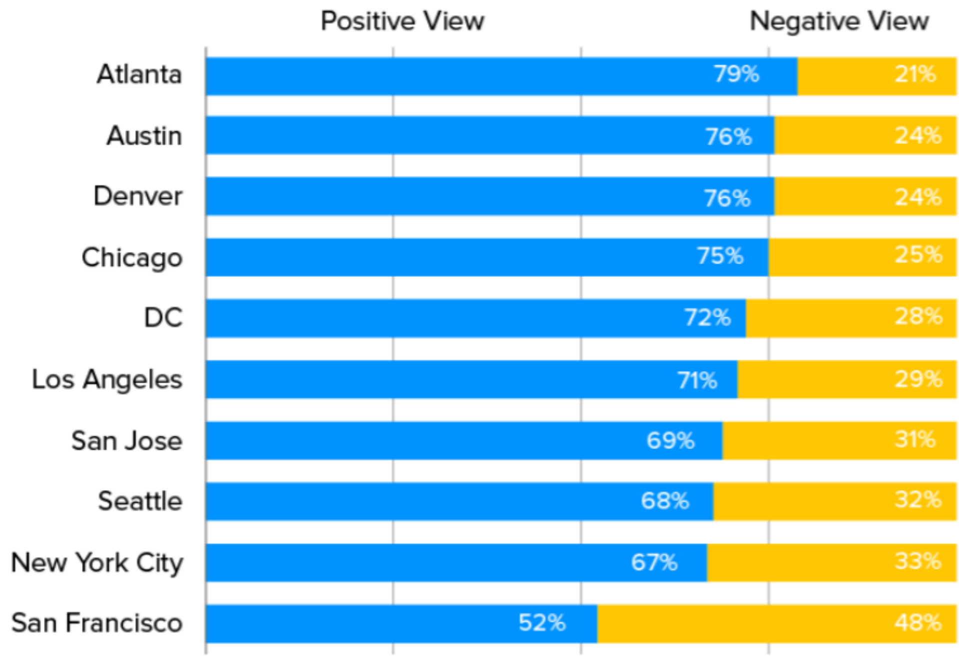


Figure 1.1 Public Perception of E-Scooters [28]

Another indication of wide public acceptance of e-scooters comes from the data shared by North American Bike Share and Scooter Share companies, which showed that they have registered 52 million e-scooter trips [27]. Since e-scooters are easy to handle, eco-friendly, low maintenance, and budget-friendly they are seen as a viable option for transportation. This thesis uses this as one of the motivations to improve safety for e-scooter users.

1.2.2 Transportation Equity

The law states that regardless of socio-economic status, race, gender, ethnicity, age, or disability safe transportation resources must be accessible [136]. Here, accessibility refers to walking paths, biking paths, roadways, and public transit. It also involves making transportation easily accessible for disabled individuals. The equity lens allows city planners to make safe roadways for all the people. With increase in micromobility users and the increase in micromobility-related accidents raises concerns about improving existing roadways to accommodate such users safely into traffic.

1.2.3 LiDAR Sensor

LiDAR is an acronym for Light Detection And Ranging, it is a laser-based remote sensing technology. This technology was shortly introduced after the invention of lasers in 1960. LiDAR's architecture consists of laser beam sources and photodetectors, depending on the resolution requirements the laser beam sources increase. 1, 16, 32, 64, and 128 are the prevalent channels of resolution configurations. The instrument works based on two measurements, the first being the distance measured with respect to its location and the second being the position of the sensor in the environment (onboard or roadside) [137]. Table 2.1 mentions other possible perception sensors with their pros & cons. Comparatively LiDAR sensors have higher resolution, wide range, and are versatile. It is also the best choice for operating in changing weather conditions. With the adoption of 850 nm wavelength standards instead of the regular 905 & 1,550 nm, the point clouds are clearer in rainy, foggy, and snowy conditions, it also reduces power consumption. As technology improves, we are seeing a drop in prices. In 2015 the price of a single LiDAR unit was approximately marked around $75,000 [138]. Fast forward to today's market in 2024, a seventh generation top-of-the-line autonomous driving grade LiDAR from Ouster is priced at $25,999 (OS2-32) & $26,999 (OS2-128). Lower-end models of Ouster like OS1-32 and OS1-128 are priced at $8,000 and $18,000 respectively [69]. This trend of falling prices indicates improvements in manufacturing technology and the introduction of more competition. Thus, it is safe to assume that in the future LiDARs will be more advanced, affordable, and prevalent in perception space and will likely be integrated with traffic lights and infrastructure for safety and data collection purposes. Keeping the wide application potential of LiDARs in mind, this research was developed to identify classes on the point cloud data.

1.2.4 Artificial Intelligence in Intelligent Transportation System

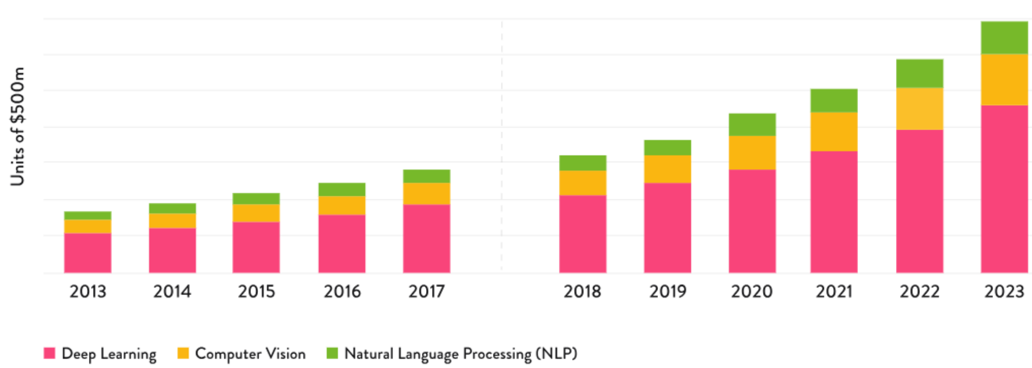


Figure 1.2 AI adoption in the global transportation industry. [60]

As AI is becoming more prevalent it is being more quickly absorbed by the transportation market to make commuting faster, eco-friendly, and safer. Figure 1.2

shows the increase in AI usage in the transportation industry. The graph includes government institutions like Intelligent Transportation System Joint Program Office (ITS JPO) which was developed to improve roadway safety and travel quality. This has been possible due to the availability of data concerning pedestrians and vehicular traffic in an area. Depending on the data roadways are modified to accommodate them. Some of the algorithms utilized to reduce traffic congestion are Ant Colony Optimizer (ACO), Genetic Algorithm (GA), Fuzzy Logic Model (FLM), Simulated Annealing (SA), and Artificial Neural Network (ANN). All the algorithms listed are optimization algorithms that can be used to optimize traffic speeds [62]. The algorithm proposed in this thesis can be used by roadside LiDARs to identify e-scooter users and collect data on their frequency to adjust road designs to accommodate them.

1.3 Gaps in Related Research

As explained previously, this thesis is amongst the very few to present a perception solution to classify e-scooter users in point cloud data. The existing methods use models like PointPillars [10] to train models to identify them. In 2D space, perception solutions to identify e-scooters in digital images have emerged in previous years. Apuruv and his team [76] are probably the first to bring a computer vision solution to this problem. They proposed to utilize YOLO V3 [77] architecture that has been pre-trained on the [78] COCO dataset. This algorithm essentially points out all the humans in each frame. To check if they were riding an e-scooter, the authors proposed a MobileNetV2 classifier [79] as the second module that would expand all the bounding boxes around the human class to see if they were riding on an e-scooter. This classifier was trained on a special dataset called the "IUPUI CSRC E-Scooter Rider Detection Benchmark Dataset." The data pool has 10,749 digital images of e-scooter riders and 10705 images of non-e-scooter riders. That is a total of 21454 images in the collection. Although the study stated that it achieved validation results of 0.9, it did not mention if the network would be able to generalize on relatively new data [76]. Using the same YoloV3 architecture, research was done to identify e-scooters, but the authors proposed a novel approach to separately identify the riders in an independent class. It is achieved by dividing the image into grids and relating parallel bounding boxes of the classes of interest. The study concluded that it had a validation accuracy of over 0.9 but the network was only trained on 140 images and tested on 60 images [80]. All images used in this research were obtained through internet image search. Since this study used cherry-picked images to test and train, questions about the algorithm's applicability in real-life situations are being raised. The study also did not detail if their network could tackle occlusion or generalize on a foreign dataset. Based on the methodology proposed by Apuruv [76], a new study was released to make the model occlusion resistant. The model used two networks: COCO dataset trained CenterNet-Hourglass 104 [83] to detect pedestrians and ResNet101 [82] classifier to detect e-scooters. They also released a chart to differentiate the performance of different classifiers trained on the same benchmark dataset [81].

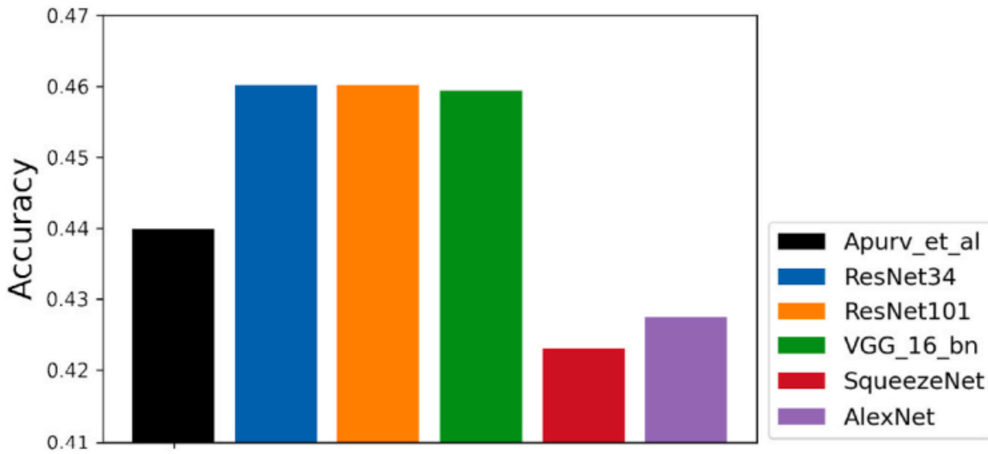


Figure 1.3 Differentiating Classifiers [81]

The data from the experiment concluded that the ResNet101 [82] and ResNet34 [146] test accuracy was 0.460 compared to the method proposed by Apuruv which scored 0.439 [81]. Provided here [84] is a comparison of different Yolo architecture's performance in detecting e-scooters in urban landscapes. In the latest, a study was conducted that resulted in this methodology that can identify any micromobility user using YOLOX [85] and Flow Guided Feature Aggregation (FGFA) [86]. Essentially, these algorithms extract the spatiotemporal information of an actor to determine its class. Results also show that it is effective against blurry and occluded images as this architecture uses data from previous frames for continuous object classification. The algorithm was tested in a private dataset consisting of 4000 bicycles, 2500 skateboards, and 2000 electric scooters [87]. The results showed that the Average

Precision (AP) for bicycles was 45.0, 23.2 for skateboards, and finally 47.6 for electric scooters, thus having a 38.6 mean average precision (mAP) for the model. Since the test was conducted in a private dataset there is no way of telling if this methodology is better than the previous methodologies listed above. Although there has been significant research done in the last few years, to date there has been no perception algorithm uniquely developed to identify e-scooters in point cloud data. This thesis most likely will be the first to do so.

1.4 Major Contributions

To achieve class recognition, the thesis proposes to harness two types of data "elasticity" and "spatial", which can only be extracted from 3D point cloud data. These two distinct features are then used to train ensemble learning algorithms. Both elasticity and spatial data are well defined in the third chapter of this thesis. To further inform the readers about this topic, this thesis includes a literature survey and methods to further improve the effectiveness of the proposed algorithm.

1.5 Road Map

This thesis is split into five major chapters. The first chapter gives the introduction and overview of the thesis. Followed by motivation, gaps in the literature survey, and the major contribution presented in this thesis. The second chapter gives all the background information for the topics presented in this thesis. This allows readers across different technical backgrounds to understand the works presented in this study. Chapter three gives the methodology of the proposed algorithm's architecture it also defines the terms "elasticity" and "spatial" data. The fourth chapter presents the readers with the results of the proposed architecture, it also points out some of the difficulties faced when working with a real-life dataset. The final chapter gives the conclusions of the work and points toward possible methods to improve the algorithm's effectiveness in real-life scenarios.

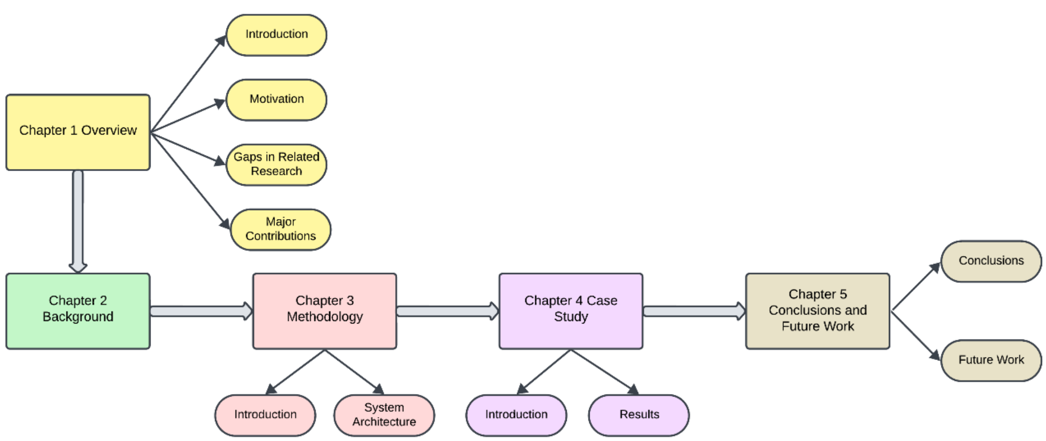


Figure 1.4 Road Map

Chapter 2 Background

This chapter is put together to give background information on some of the prevailing topics discussed in this thesis. As mentioned in Chapter 1, the World Health Organization estimates that approximately 1.9 million road traffic-related deaths happen every year. An estimated sum of US $1.8 trillion is forecasted to compensate for all the road accident injuries caused between 2015 & 2030 [2]. Statistics like these have compelled researchers to focus more on vulnerable road users. Fig 2.1 shows the increase in VRU research publications in recent years, the collective sum is indicated by the red line.

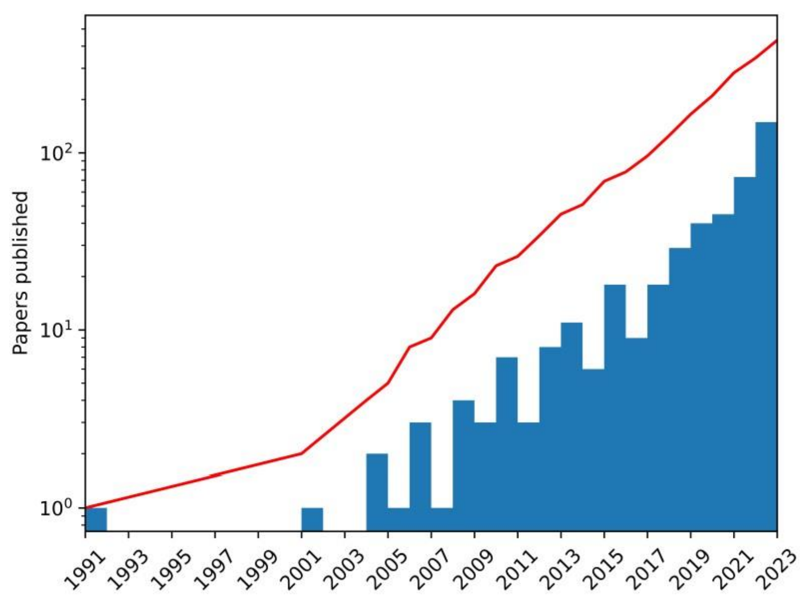


Figure 2.1 VRU research publications in IEEE Xplore. [3]

Most of the Intelligent Transportation Systems (ITS) being developed to make city traffic safer start with the most crucial step, perception. All the widely used perception sensors are listed, and their abilities are compared in Table 2.1.

Feature	Visible camera	Thermal camera	Radar	LiDAR	Ultrasonic	Acoustic	UWB
Night vision capability	Low	High	High	Medium	Low	Medium	Medium
Image resolution	High	Medium	Low	High	Low	Low	Low
Color perception	High	Low	Low	Low	Low	Low	Low
Detection range	Medium	High	High	High	Low	Medium	High
Field of view	Wide	Medium	Narrow	Medium	Narrow	Wide	Wide
Weather resistance	Low	High	High	Medium	Medium	Low	High
Cost	Medium	High	High	Very High	Low	Low	Medium

Table 2.1 Perception sensors. [3]

In the upcoming subtopics, all the research undertaken to detect VRU is presented, along with the prevalent datasets available in this domain.

2.1 Working of LiDAR Sensor

Before we dive deep into perception topics in point cloud it is important to have a good understanding of the LiDAR's working. This subsection goes through the basic work principles of LiDAR. As mentioned in the previous chapter, the instrument works based on two measurements, the first being the distance measured with respect to its location and the second being the position of the sensor in the environment (onboard or roadside). The distance can be measured by utilizing pulsed laser to measure the time of flight, which is essentially the time taken by the pulse to reach the photoreceptor after reflecting from a surface.

$$D = C \cdot \triangle T/2 \quad (2.1)$$

With D being the distance measured, C being the speed of light and $\triangle T$ the time of flight. The equation above is used to measure the distance between the source and the object. The equation shows that the system is limited only by the returning pulse, meaning that miles of distance can be accurately mapped with high-powered lasers.

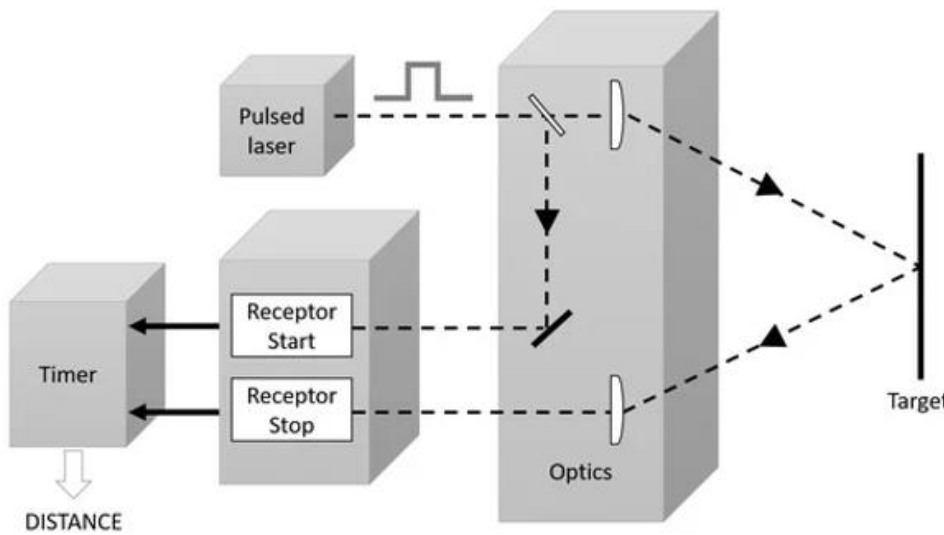


Figure 2.2 Time of flight measurement [137].

The figure presented above explains the concept of time-of-flight measurement in LiDARs. Another approach to calculate the distance would be to measure the phase by utilizing Amplitude Modulated Continuous Waveform (AMCW) lasers. The phase shift between the returning pulse and the incident pulse employed is used to calculate the distance.

$$D = \frac{c}{2} \cdot \triangle \phi/(2 \cdot \pi \cdot f_M) \quad (2.2)$$

As in the previous equation, D refers to the distance calculated and C represents the speed of light. The modulation frequency is denoted by f_M and phase shift by $\triangle \phi$. Unfortunately, the maximum range that can be measured precisely is approximately 100 m (328.08 ft).

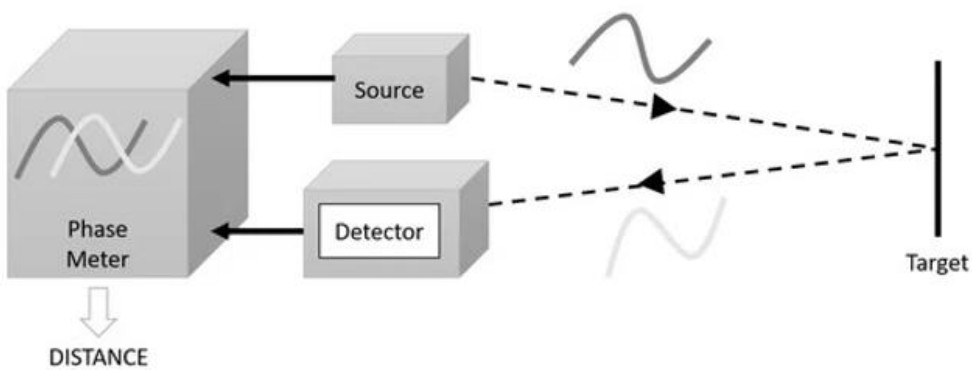


Figure 2.3 Time of flight phase measurement principle [137].

LiDAR sensors are widely being adopted into robotics, ITS, and self-driving applications. Table 2.1 mentions other possible perception sensors with their pros & cons, it is clear that LiDARs have a natural advantage over the others because they utilize lasers, which in return makes them quicker and more accurate.

2.2 Dataset

This subtopic presents Table 2.2 to give the readers a brief idea of all the available VRU datasets that contain point cloud data, it also details the other sensors used, the nature of the data (real or synthetic), the types of classes present and if it was captured by sensors onboard of a vehicle or sensors fixed in a structure. As more complex models are being developed the need for accurately annotated datasets becomes more prevalent. To mitigate this issue synthetic datasets are being synthesized. Softwares like SUMO (Simulation of Urban MObility) [40], OpenCDA [39] CarMaker [108], Matlab, and CARLA (CAR Learning to Act) [41] feature tools to generate synthetic Lidar point cloud data. Since LiDAR data does not contain complex data like texture, lighting, and color [38] 3D rendering software like Unreal Engine 4 can replicate LiDAR data. But since the real-world data contains noises and occlusion which are usually not found in synthetic datasets the usage is limited. Figure 2.4 shows the perfection that comes with virtual datasets, and Figure 2.5 shows a real point cloud frame. Notice the stark contrast of perfection in their geometry and 3D rendering between those scenarios. To account for that factor CARLA has introduced a toolbox that allows users to add imperfections. Users can change the atmosphere attenuation rate (measures K loss per meter), droop-off off general rate (probability of points being randomly eliminated), drop-off intensity limit (probability of points with K above a limit is untouched), drop-off intensity (probability of points with zero K value being dropped) and noise standard deviation [m] (standard deviation of in-built noise model to affect points). All the synthetic datasets are marked in red for easy spotting. To prove the hypothesis in this thesis a private dataset with high-resolution point clouds and a high number of e-scooter users was used. The combination of these two features was not available in open datasets.

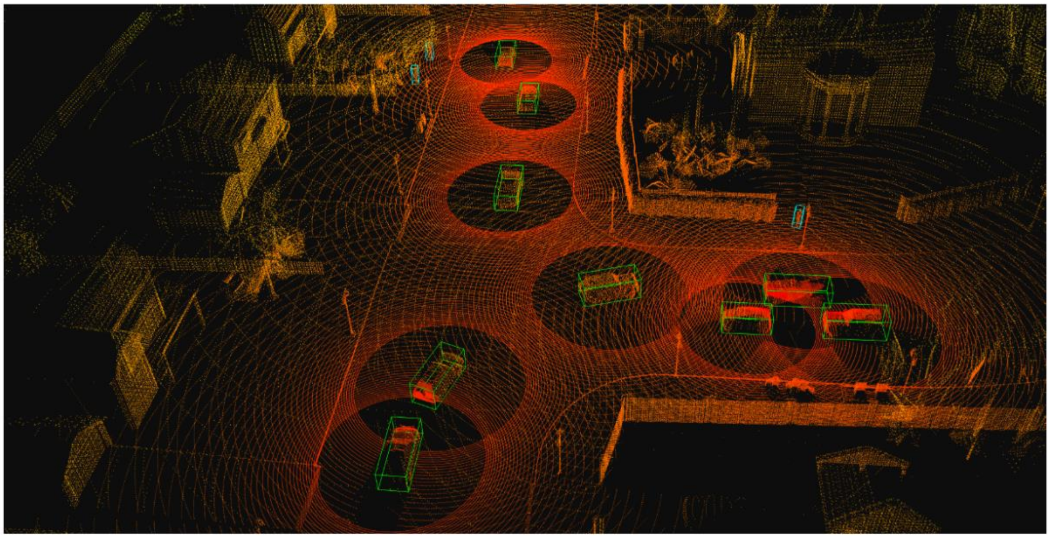


Figure 2.4 Frame from CODD synthetic dataset [67]

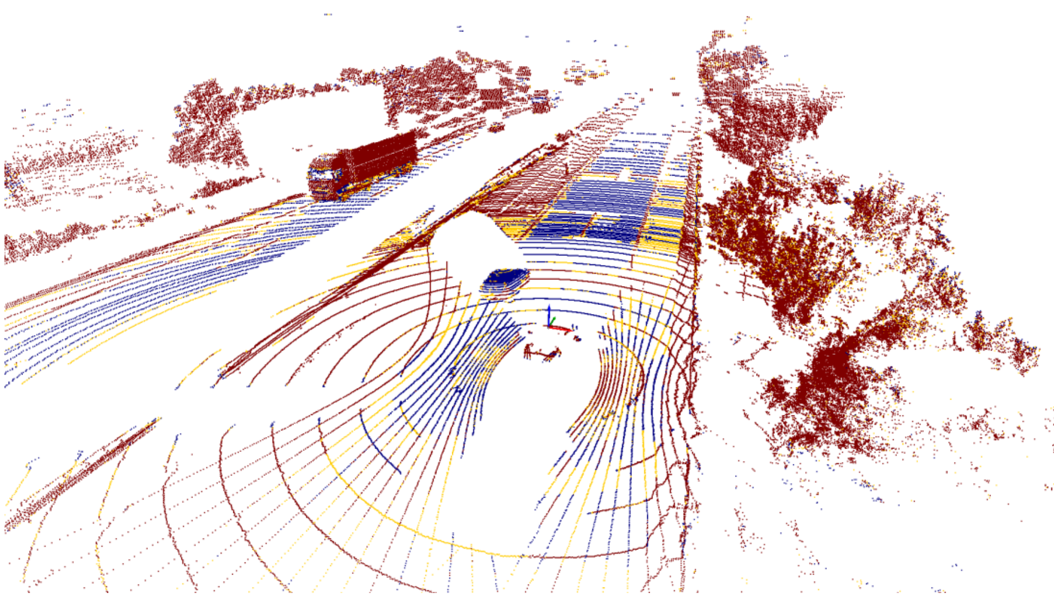


Figure 2.5 Frame from Zenseact Open Dataset [68]

Dataset	Sensor	Real/Simulation	Onboard/Roadside	VRU Class Present
KITTI [17]	Camera & LiDAR	Real	Onboard	Pedestrian & Cyclist
Oxford RobotCar [20]	Camera, LiDAR & Radar	Real	Onboard	Pedestrian & Cyclist
Astyx [21]	Camera, LiDAR & Radar	Real	Onboard	Pedestrian, Motorcyclist & Cyclist
Dense [22]	Camera, LiDAR & Radar	Real	Onboard	Pedestrian
Rope3D [126]	Camera & LiDAR	Real	Onboard/Roadside	Pedestrian & Cyclist
TUM Traffic [129]	Camera & LiDAR	Real	Roadside	Pedestrian, Motorcyclist & Cyclist
RCooper [127]	Camera & LiDAR	Real	Roadside	Pedestrian, Motorcyclist, Tricycle & Cyclist
DeepAccident [128]	Camera & LiDAR	Synthetic (CARLA)	Onboard/Roadside	Pedestrian, Motorcyclist & Cyclist
Argoverse 1 [31] & 2 [32]	Camera & LiDAR	Real	Onboard	Pedestrian, Moped, Stroller, Motorcyclist& Cyclist
nuScenes [19]	Camera, LiDAR & Radar	Real	Onboard	Adult Pedestrian, Child Pedestrian, Personal Mobility, Police, Construction Worker, Wheelchair, Stroller, Motorcyclist& Cyclist
MulRan [34]	LiDAR & Radar	Real	Onboard	Pedestrian
SemanticPOSS [47]	LiDAR	Real	Onboard	Pedestrian & Cyclist
WADS [36]	Camera & LiDAR	Real	Onboard	Pedestrian
BAAI-VANJEE [49]	Camera & LiDAR	Real	Roadside	Pedestrian, Motorcyclist, Cyclist & Tricycle

Waymo Open Dataset [18]	Camera & LiDAR	Real	Onboard	Pedestrian, Motorcyclist & Cyclist
ONCE [37]	Camera & LiDAR	Real	Onboard	Pedestrian & Cyclist
RADIATE [35]	Camera, LiDAR & Radar	Real	Onboard	Pedestrian, Group of Pedestrian, Motorcyclist & Cyclist
CODD [74]	Camera & LiDAR	Synthetic (CARLA)	Onboard	Pedestrian
V2X-Sim [121]	Camera & LiDAR	Synthetic (CARLA-SUMO)	Roadside	Pedestrian
DAIR-V2X [72]	Camera & LiDAR	Real	Onboard & Roadside	Pedestrian & Cyclist
DOLPHINS [123]	Camera & LiDAR	Synthetic (CARLA)	Onboard & Roadside	Pedestrian
OPV2V [75]	Camera & LiDAR	Synthetic (OpenCDA & CARLA)	Onboard	Pedestrian
View-of-Delft [46]	Camera, LiDAR & Radar	Real	Onboard	Pedestrian, Cyclist & Moped
IPS300+ [48]	Camera & LiDAR	Real	Roadside	Pedestrian, Motorcyclist, Cyclist & Tricycle
V2X-ViT [122]	LiDAR	Synthetic (CARLA & OpenCDA)	Onboard & Roadside	Pedestrian
SynLIDAR [120]	LiDAR	Simulation (Unreal Engine)	Onboard	Male, Female, Kid, Motorcyclist & Cyclist
Deliver [119]	Camera & LiDAR	Synthetic (CARLA)	Onboard	Pedestrian & Two-Wheeler
Zenseact [42]	Camera, LiDAR & Radar	Real	Onboard	Pedestrian, Cyclist, Motorcyclist, Stroller, Wheelchair & Personal Transporter
REHEARSE [110]	Camera, LiDAR & Radar	Real & Synthetic	Onboard	Pedestrian
TWICE [109]	Camera, LiDAR & Radar	Real & Synthetic (CarMaker)	Onboard	Pedestrian & Cyclist
IMPTC [50]	Camera, LiDAR & UWB	Real	Roadside	Pedestrian, Cyclist, Wheelchair, E-Scooter & Stroller

WiDEVIEW [45]	Camera, LiDAR & UWB	Real	Onboard	Pedestrian
V2V4Real [44]	Camera & LiDAR	Real	Onboard	Pedestrian
IAMCV [43]	Camera & LiDAR	Real	Onboard	Pedestrian & Cyclist
V2X-Real [73]	Camera & LiDAR	Real	Onboard & Roadside	Pedestrian, Cyclist, Motorcyclist & Scooter

Table 2.2 All the available VRU datasets featuring LiDAR data.

2.3 LiDAR Point Cloud Pre-Processing Methods

Like digital images, cloud data also comes with noises and there is a variety of methods to handle it. This subtopic is aimed at explaining some of the popular preprocessing techniques. Starting with Frequency Based Noise filtering [90], which can increase clarity and boost signal accuracy by eliminating noises in targeted frequency. This might not be beneficial to combat distortions or noises that are non-frequency related. Variational Mode Decomposition [91] allows for marking important LiDAR echoes by strained decomposition of all the available signals. The success rate very much depends on parameter tuning which can get more complicated with more signals involved. Although computationally expensive with the inherent risk of increasing noise if not properly tuned the Richardson Lucy Deconvolution [92] method improves cluster delineation by adjusting blurring effects. Studies often use Adaptive Noise Reduction via PCA [93] as it can denoise with less computation power while preserving the data. However, it is observed to have poor performance if the point clouds are sparse and irregular. Real-time CNN for segmentation [94] might require specific hardware tools like (FPGA) Field Programmable Gate Array with NVIDIA deep learning accelerator. But it is rated to be fast and efficient with low power requirements. Similar to this, the Gaussian Decomposition for FPGA [95] will need FPGA hardware, but it yields faster processing speeds that allow it to work in real-time.

2.4 Single LiDAR setup

Each LiDAR point cloud data has four values associated with it, (X, Y, Z, K). The first three values represent the coordinates, and the last value indicates the intensity. Usually, LiDAR point clouds contain millions of points per scan. Ouster LiDAR series OS0, OS1 & OS2 give an output of 5.02 M, 5.02 M & 2.62 M points per scan [69]. Before the data is fed into the perception algorithm it is usually down-sampled to reduce computational complexity. Uniform downsampling, farthest point, and nearest neighbor are some of the algorithms widely used to downsample the data. Algorithms like Point Net [14] [15] [16] strongly advocate for directly processing point clouds as data transformation renders the resulting data unnecessarily voluminous — while also introducing quantization artifacts that can obscure natural invariances of the data.

Method	Modality	Speed (Hz)	mAP	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar & Img.	2.8	N/A	86.02	76.90	68.49	N/A	N/A	N/A	N/A	N/A	N/A
Cont-Fuse [15]	Lidar & Img.	16.7	N/A	88.81	85.83	77.33	N/A	N/A	N/A	N/A	N/A	N/A
Roarnet [25]	Lidar & Img.	10	N/A	88.20	79.41	70.02	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN [11]	Lidar & Img.	10	64.11	88.53	83.79	77.90	**58.75**	**51.05**	**47.54**	68.09	57.48	50.77
F-PointNet [21]	Lidar & Img.	5.9	65.39	88.70	84.00	75.33	58.09	50.22	47.20	75.38	61.96	54.68
HDNET [31]	Lidar & Map	20	N/A	89.14	**86.57**	78.32	N/A	N/A	N/A	N/A	N/A	N/A
PIXOR++ [31]	Lidar	35	N/A	**89.38**	83.70	77.97	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet [33]	Lidar	4.4	58.25	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
SECOND [30]	Lidar	20	60.56	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
PointPillars	Lidar	**62**	**66.19**	88.35	86.10	**79.83**	58.66	50.23	47.19	**79.14**	**62.25**	**56.00**

Table 2.3 PointPillar detection on BEV detection benchmark Kitti Dataset (mAP). [10]

To combat point cloud sparsity voxels were used to represent data to algorithms that work in 3D convolutional space [4] [5] [6] [7] [8]. Voxels divide the point cloud data into three-dimensional cubes, encompassing the points within it. Voxel sizes are predefined depending on the required spatial resolution and available computational resources.

Unlike point and voxel representations, the Bird's Eye View (BEV) type of representation is usually used for algorithms that work in 2-D convolutional space. Since LiDAR points cannot overlap in a given frame its top view is projected onto a horizontal plane. This technique allows for the utilization of algorithms developed for 2-D images to detect objects in point cloud data [9] [10] [11] [12] [13]. Above attached is Table 2.3 that shows the effectiveness of PointPillar (an architecture utilizing voxels to combat data cloud sparsity) on the BEV dataset. Table 2.4 shows its effectiveness in normal point cloud dataset.

Method	Modality	Speed (Hz)	mAP	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar & Img.	2.8	N/A	71.09	62.35	55.12	N/A	N/A	N/A	N/A	N/A	N/A
Cont-Fuse [15]	Lidar & Img.	16.7	N/A	82.54	66.22	64.04	N/A	N/A	N/A	N/A	N/A	N/A
Roarnet [25]	Lidar & Img.	10	N/A	**83.71**	73.04	59.16	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN [11]	Lidar & Img.	10	55.62	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
F-PointNet [21]	Lidar & Img.	5.9	57.35	81.20	70.39	62.19	51.21	**44.89**	40.23	71.96	56.77	50.39
VoxelNet [33]	Lidar	4.4	49.05	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
SECOND [30]	Lidar	20	56.69	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointPillars	Lidar	**62**	**59.20**	79.05	**74.99**	**68.30**	**52.08**	43.53	**41.49**	**75.78**	**59.07**	**52.92**

Table 2.4 PointPillar detection on 3D detection benchmark Kitti Dataset (mAP). [10]

2.5 LiDAR Fusion

LiDAR fusion is a new field of research that allows for fusing two types of sensors for perception tasks. The subsections below talk about the different types of LiDAR fusion research available for VRU detection.

2.5.1 Multi LiDAR Fusion

Due to limitations such as sparse point clouds, occlusions, noise, and the flexible nature of the pedestrian class, researchers found it difficult to implement their algorithms. The research trend slowly changed as researchers believed denser LiDAR points may lead to better detection performance [51]. Therefore, some researchers have tried to fuse data from multiple independent LiDARs. Sensor fusion was believed to be the answer to most of the technical difficulties mentioned above. Primarily there are two methods of sensor fusion strategies, pre-classification & post classification [52], measurement integration done at the feature level or at raw data level falls into pre-classification. Measurement integration done after processing of the raw data falls under post-classification. Score & rank are widely sought-after features to be fused under this class [59]. To enable better perception for autonomous cars, it was proposed that a multi-LiDAR setup would be able to better determine the geometric features of the roadways thus enabling the vehicle to ride on road and off-road trails, also on low lighting conditions [53] [54] [55]. The basic philosophy behind dual multi-LiDAR setup is to create an overlapping space that is expected to have more denser point clouds thus enabling better perception [56]. The same principle was tried on robots to ensure better mobility in both indoor & outdoor environments [57]. This setup when utilized for pedestrian detection yielded better results [58]. The suggested framework surpasses conventional raw data fusion algorithms in various scenarios. Its adaptability enables the utilization of diverse classification algorithms prior to fusion. Nevertheless, detection accuracies noticeably diminish when pedestrians are partially obscured. Challenging settings, particularly adverse weather conditions like rain, snow, and airborne particles, significantly influence algorithm performance. As Table 2.5 below shows, this multi-LiDAR setup makes perception faster and more accurate.

Method	3D Detection AP (%)			BEV Detection AP (%)			Times (s)
	Easy	Moderate	Hard	Easy	Moderate	Hard	
AVOD [25]	36.10	27.86	25.76	42.58	33.57	30.14	0.08
Complex-YOLO [29]	17.60	13.96	12.70	21.42	18.26	17.06	0.06
BirdNet [10]	12.25	8.99	8.06	20.73	15.80	14.59	0.11
TopNet-HighRes [24]	10.40	6.92	6.63	19.43	13.50	11.93	0.10
Ours	33.75	26.64	23.34	49.27	37.96	33.83	0.026

Table 2.5 Detection results (Kitti Dataset) [58]

2.5.2 LiDAR and Camera Fusion

Although LiDAR's point clouds contain valuable depth information it lacks data on texture and color found in digital images. Fusing point clouds with digital images is an option to compensate for LiDAR's data structure. But this also means that the model must process noises occurring in the digital images. Gaussian blur is usually preferred to eliminate noises that might be present. Models like the flat earth model [64] exploit the assumption that relevant pedestrians are situated on a flat surface, such as the road or walkway, thus making it faster to find the region of interest. This model calculates the image region corresponding to the ground based on camera geometry, assuming a flat ground in the vehicle's frontal view. Figure 2.6 below shows the comparison data. The relaxed free world model refers to having a small threshold of tolerance, since in the ideal world the surface is not completely flat, thus, to account for irregularities the relaxed flat world model was introduced. Although it makes the model faster, LiDAR-only detection methods tend to outperform multi-sensor fusion methods in public benchmarks [63]. This is because practically during fusion, data from separate modalities tends to generalize at independent rates, and in some scenarios, they overfit [65]. PointPainting [66] a sequential fusion method was able to bridge the gap. PointPainting utilizes cameras to obtain digital images and performs semantic segmentation. After completion, the digital image will contain pixel-wise scores. The point clouds are allowed on the scene and now undergoes a point cloud detection pipeline. Depending on the two scores a decision is made. As mentioned in the earlier topic, this type of fusion is called post-classification [52]. Below, Figure 2.7 is a detailed image explaining this process. Figure 2.8 shows the improvement in detection results in the Kitti dataset, a total increase of 6.3 mAP can be observed in the Painted PointPillar ++ across ten different classes [66]. The existing performance on pedestrian detection can be improved by being capable of performing fusion at various levels [19]. Semantic Voxels utilizes semantic augmentation on the point cloud, the process involves encoding raw point data into pillars for geometric features and semantic point data into voxels for semantic features. These features are effectively fused. The experimental results on the KITTI test set demonstrate that Semantic Voxels achieves state-of-the-art performance in both 3D and bird's eye view pedestrian detection benchmarks. Below attached is a table showing the difference in performances as different fusion is undertaken. For both 3D and bird's eye view (BEV) detection tasks, early fusion results in the most substantial mean average precision (mAP) improvement, with a gain of 3.2 for 3D and 2.82 for BEV. Researchers are also exploring other fusion setups like thermal camera and LiDAR [70] & radar and LiDAR [71]. But to date are yet to provide any significant results that can compare with the previous state-of-the-art perception results in this domain.

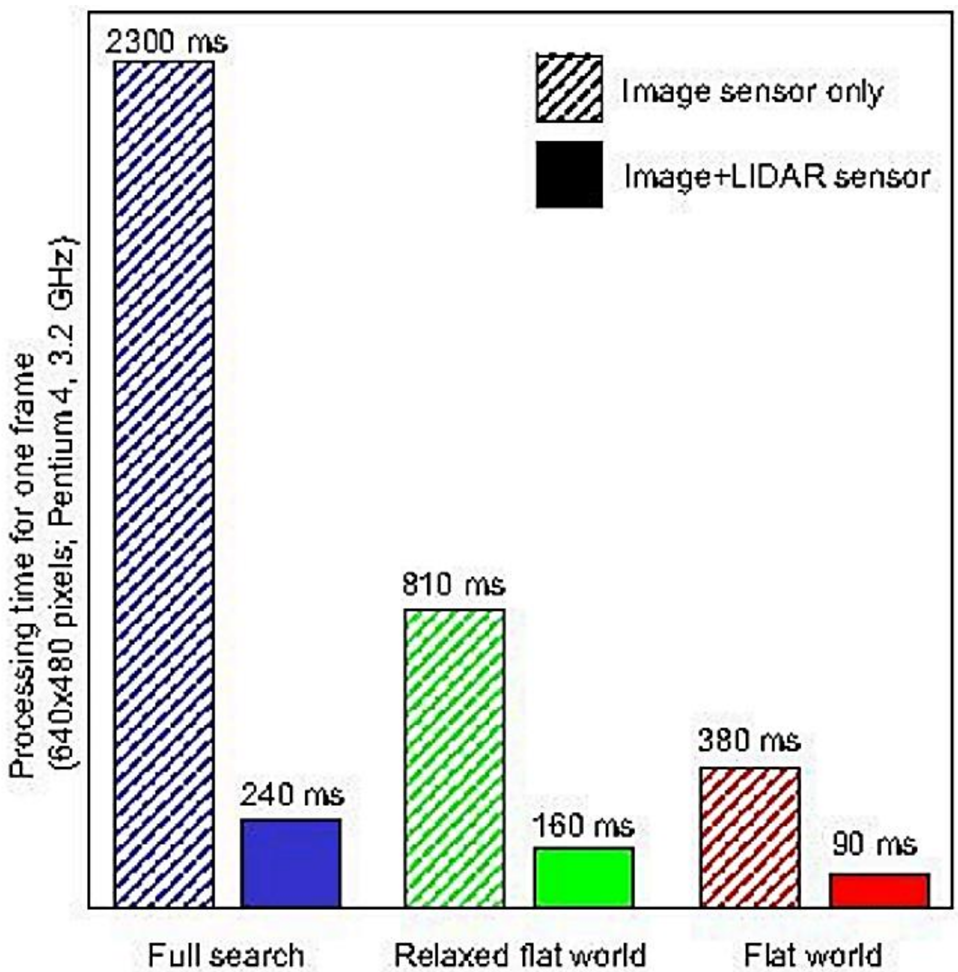


Figure 2.6 Pedestrian detection system with and without LiDAR-based ROI detection. [64]

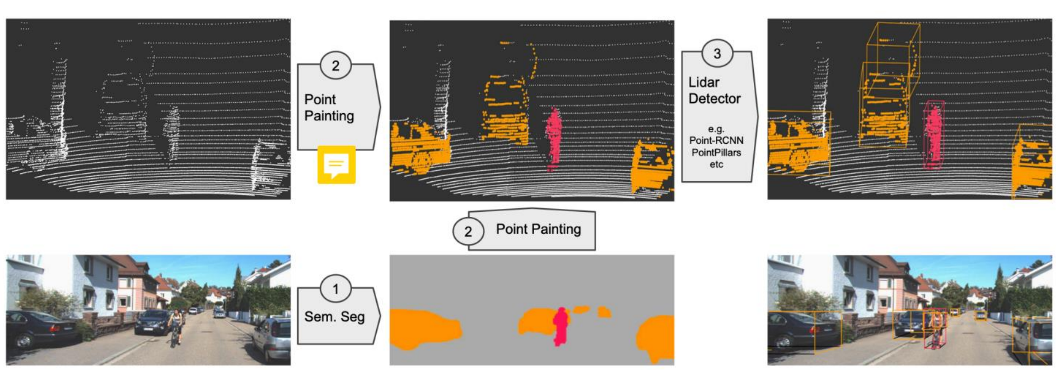


Figure 2.7 PointPainting overview [66]

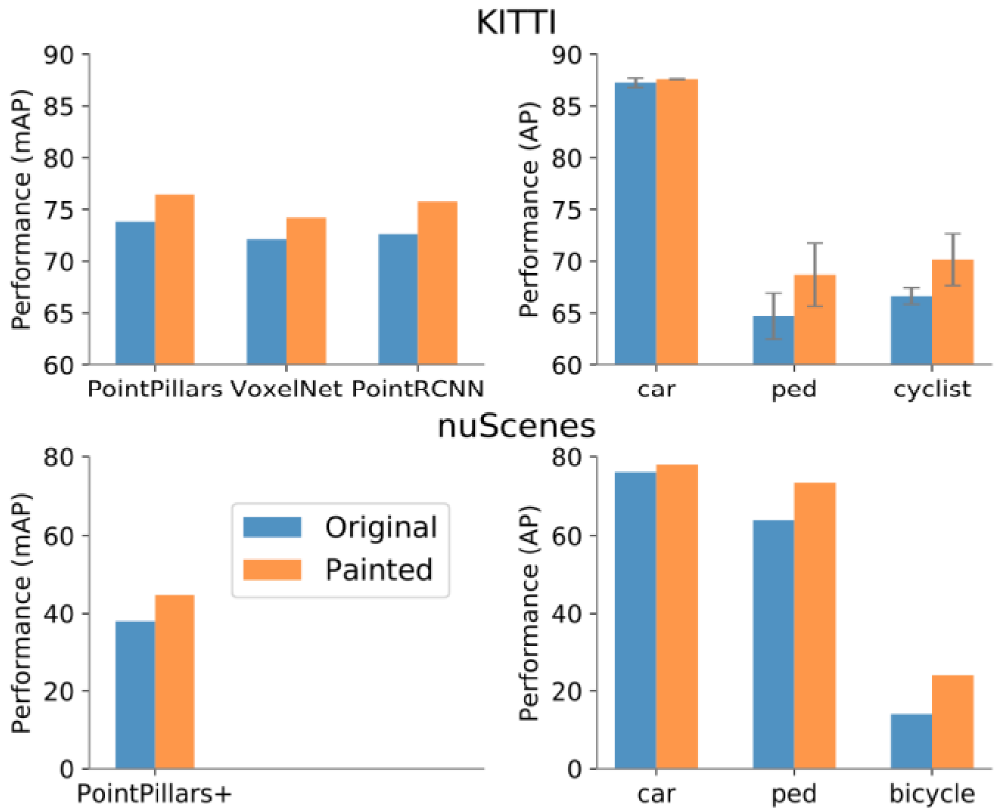


Figure 2.8 PointPainting results [66]

Method	AP_{3D} Easy	AP_{3D} Moderate	AP_{3D} Hard	mAP_{3D}	AP_{BEV} Easy	AP_{BEV} Moderate	AP_{BEV} Hard	mAP_{BEV}
PointPillars	67.69	60.94	56.02	61.55	73.42	68.89	63.32	68.54
Late Fusion	68.58	62.62	59.19	63.46	73.72	69.86	64.29	69.29
Delta	+0.89	+1.68	+3.17	+1.91	+0.30	+0.97	+0.97	+0.75
Middle Fusion	68.15	63.31	59.15	63.54	74.31	70.81	**66.54**	70.55
Delta	+0.46	+2.37	+3.13	+1.99	+0.89	+1.92	**+3.22**	+2.01
Early Fusion	**69.71**	**64.47**	**60.07**	**64.75**	**76.31**	**71.56**	66.21	**71.36**
Delta	**+2.02**	**+3.53**	**+4.05**	**+3.20**	**+2.89**	**+2.67**	+2.89	**+2.82**

Table 2.6 Performance comparison of pedestrian detection. [63]

2.6 Evaluation Metrics

As we analyze and compare results across different studies it is important to know which metric we are comparing. Starting with precision, which is defined as the true value in a set of values that the algorithm predicts as true. Mathematically it can be expressed as,

$$precision = \frac{TP}{TP+FP} \quad (2.3)$$

True Positive (TP) refers to values that are marked as rightly assumed true values and False Positive (FP) refers to values that are incorrectly assumed as true values. False Negatives (FN) are true values that are wrongly assumed as false values.

Recall, another widely used term refers to the number of times the algorithm finds the true value in all the positive values of a dataset. Mathematically it can be represented as,

$$recall = \frac{TP}{TP+FN} \quad (2.4)$$

Average precision (AP) is defined by the area under the precision-recall curve. The precision-recall curve is essentially the relation between the precision and recall values at separate thresholds. A model with high AP indicates that it is able to hold high recall and precision in most cases. They are also used to evaluate the robustness and accuracy of 3D models. AP is mathematically represented as,

$$AP = \int_0^1 p(r)dr \quad (2.5)$$

In this equation, p(r) represents a function of precision with respect to recall. [130] Intersection over Union (IoU) also known as the Jaccard Index in the context of object detection is a metric that evaluates the performance of object detecting algorithms. Generally, the IoU of two finite sets is expressed as,

$$IoU(A,B) = \frac{A \cap B}{A \cup B} = A \cap B \,/\, (|A|+|B|-A \cap B) \quad (2.6)$$

A and B are independent finite sets. In the context of 2D object detection IoU is expressed as,

$$IoU\left(B_g\,, B_d\right) = \frac{Area\ of\ overlap\ B_g\ and B_g}{Area\ of\ union\ B_g\ and\ B_d} \quad (2.7)$$

B_g refers to the bounding box of the ground truth and B_d refers to the bounding box of the predicted class. For objects in point cloud data IoU is decided in 3D space, it is standard practice to omit pitch and roll in the 3D space. It is assumed that all the bounding boxes lie flat on the ground. Thus, only yaw is taken into consideration. IoU for 3D bounding boxes is calculated using the following equation,

$$IoU_{3D} = \frac{Area_{overlap} \times h_{overlap}}{Area_g \times h_g + Area_d \times h_d - Area_{overalp} \times h_{overlap}} \quad (2.8)$$

$Area_g$ represents the area of the ground truth and $Area_d$ refers to the area of the predicted bounding box, $h_{overlap}$ & h_{union} refers to the intersection & union in the z-axis (height).

Multiple Object Tracking Accuracy (MOTA) [88] is a measurement used to evaluate computer vision models that can track objects. It is mathematically written as,

$$MOTA = 1 - \frac{(FN+FP+IDSW)}{GT} \quad \in (-\infty, 1] \quad (2.9)$$

As mentioned previously, FN and FP hold the same meaning. GT here refers to the ground truth. ID refers to the unique code given to each object tracked. ID switches, if an object tracked is given a new ID despite already being assigned an ID, then it is called an ID Switch. IDSW in the equation refers to the sum of times ID switches occurred. Ideally, a tracking algorithm's ID Switch should be null. Some of the classical metrics used in multiple object tracking are explained below.

False trajectories are predicted trajectories that fail to match with the ground truth. Mostly Tracked (MT) trajectories refer to the predicted trajectories that align with the ground truth at least in 80% of the frames. If predicted trajectories only match the ground truth in 20% of the frames, then those are marked as Mostly Lost (ML). In some publications, Multiple Object Tracking Percentage (MOTP) is preferred which is essentially the percentage report on MOTA. It is mathematically put together as,

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t C_t} \quad (2.10)$$

Here, $d_{t,i}$ refers to the overlaps of the bounding box between the hypothesis i with their marked ground truth. C_t is the count of matches in t frame [88].

Chapter 3 Methodology

3.1 Introduction

This chapter discusses the methodology used to create the algorithm proposed in this thesis. It starts with the algorithm's architecture and progresses to elaborate on each of the system's blocks individually.

3.2 Architecture

Figure 3.1 gives a brief description of the algorithm through the block diagram. In total, it contains five crucial subsystems (plane elimination, clustering, fitting convex hull, tracking & observing the elasticity, and spatial data exhibited) which will be elaborated further in the subtopics below. The algorithm is designed to take in segments of raw point clouds from the BEV perspective. This allows the algorithm to function more efficiently as background elimination algorithms do not have to be used, but ground plane elimination using RANdom SAmple Consensus (RANSAC) is done on the frame [97] [98]. It is further clustered using Density-Based Spatial Clustering of Application with Noise (DBSCAN) to avoid noises [96] [99] and identify all other actors that might be present in the frame. All the clusters are then fitted with a convex hull to find the volume. The architecture proposed in this thesis can classify classes by analyzing the elasticity and spatial data of an actor. The movement of the physical body in order to generate motion is defined as elasticity in this work. Since humans must physically relocate their limbs even to generate the slightest momentum, elasticity is significantly observed, compared to the e-scooter users who must remain still and balanced while traveling.

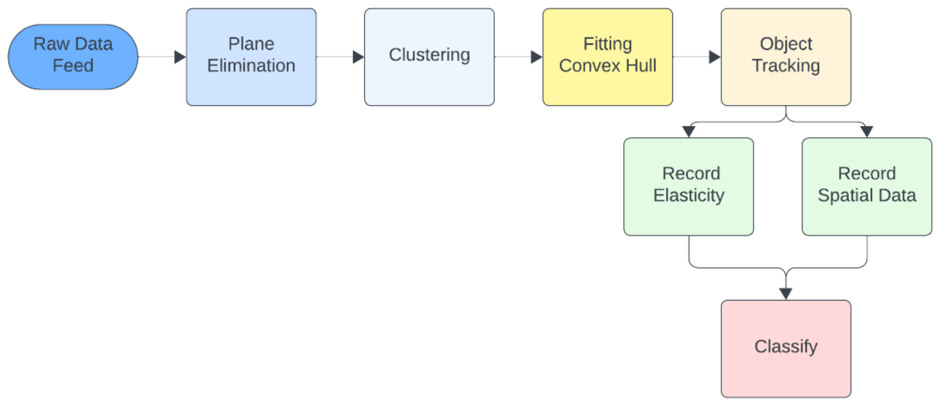


Figure 3.1 Proposed algorithm's block diagram

This elasticity is calculated by covering the object of interest with a convex hull and recording its volume in every frame as the cluster starts to gain momentum. So essentially, the algorithm tracks the cluster of interest and records the volume of the convex hull for *t* amount of time. Simultaneously spatial data is also recorded. Spatial data is the accumulation of a cluster's point cloud over *t* amount of time. After every successful accumulation the volume of the cluster as a whole is recorded. Since e-scooter users travel faster while maintaining little to no movement of their bodies the algorithm can differentiate them with the criteria mentioned above. Both of these terms are mathematically explained in section 3.7.

3.3 Ground Plane Elimination

Plane elimination allows for the segmentation of the actors in interest. Since the proposed algorithm requires the calculation of the volume it is vital that the ground plane is eliminated before processing it further. To segment geometry in a point cloud, the algorithm must be based on any of the following methodologies: Region-Based, Edge Based, Graph-Based, Model-Based, or Attribute Based [100]. Starting with the region-based method, this methodology uses data from similar neighboring points to identify isolated regions and continually finds differences between regions. Comparatively, noise is resistant to the edge-based method but is known to under-segment or over-segment in some cases. This method is further divided into Seeded Region and Unseeded Region. To use the Seeded Region method seed points must be predefined. Each region will then develop from the predefined seed points and grow by adding more neighboring points that comply with the set threshold [101].

It is imperative to choose optimal seed points as this approach is very dependent on the predefined seed points and is very time-consuming. The Unseeded Region method does not require any predefined points, instead, it gathers all the points into a set and starts to divide into smaller sets. It continues until it can no longer fit the number of set points beyond the set threshold. This method can't be practically used in complex scenes with unknown parameters. It requires a lot of prior knowledge like the number of regions, object models, etc. [102]. The edge-based method primarily relies on locating points with sudden changes in K to determine boundaries. This was originally inspired by the work of Bhanu and his team. Sparsity, noise, and uneven collection of point clouds make it challenging to apply this in real-life scenarios [103].

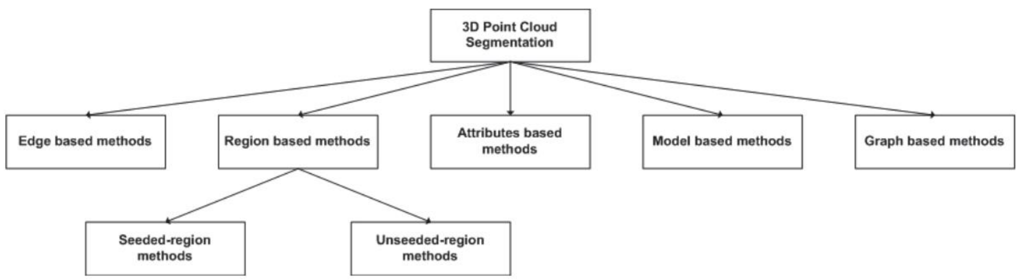


Figure 3.2 3D point cloud Segmentation methods. [100]

Model-based methods utilize pre-defined geometric shapes like squares, circles, etc. to segment shapes in a given point cloud frame. This idea is combined with the RANSAC model to segment point clouds. Since RANSAC was initially designed to identify mathematical shapes, it was an instant match to work in 3D segmentation. This method was further improved to segregate point cloud data and mesh. It was also able to identify primitive shapes in unorganized point cloud data [104]. Expanding on this work, new research was published detailing the detection of shapes that are translationally and rotationally symmetrical called Slippable shapes. Linear extrusion, plane, helix, surfaces of revolution, sphere, and cylinder fall under this category [105].

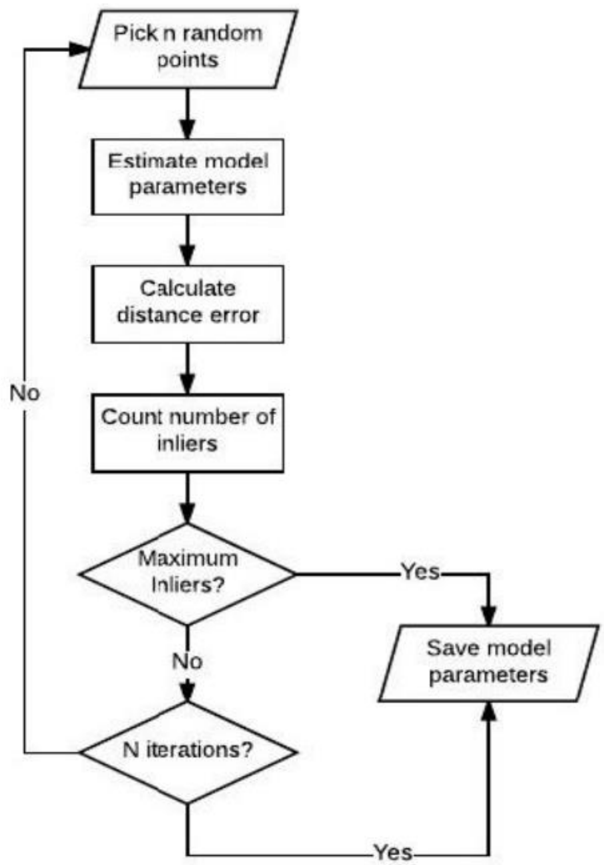


Figure 3.3 RANSAC flowchart. [106]

Graph-based methods are popular in autonomous navigation projects due to their robustness and efficiency. It works by using radius-based approaches or KNN [107] to make a graph from the point clouds. Graph Laplacian and clustering are further performed to segment planes. Usually, it is not used in real-time and requires special sensors and cameras to run this methodology [100]. Finally, attribute-based methods segment based on the available clustering features in a point cloud scene. Each cluster with a unique feature will be segmented. This methodology mostly relies on the clarity of the points nearby and is very time-consuming when working with multidimensional features of large point cloud scenes.

3.4 Clustering

Clustering allows for isolating clusters of interest and tracking them to observe their elasticity and spatial data. Given below are some of the most widely used clustering algorithms in the point cloud domain.

K-means is a widely used clustering algorithm that works by iteratively combining points to the neighboring cluster's center point and continues to update the midpoints based off the mean of the points assigned [111]. Density-Based Spatial Clustering of Applications with Noise (DBSAN) [99] performs clustering operations by grouping points based on their density and identifies the outlier points as noise. Epsilon is the predefined search radius and min points is the number of minimum points that should constitute a cluster. This value is also predefined, the optimum value is found by the trial-and-error method.

Considered an extension of DBSCAN, the Ordering Points To Identify the Clustering Structure (OPTICS) algorithm [115] classifies points by their cluster density, and based on the density it maintains a hierarchical order. Unlike DBSCAN this does not need epsilon value, it adapts to varying density using a reachability plot which makes it computationally expensive. Utilizing a similar methodology, the Mean Shift Clustering algorithm locates dense point clouds and clusters them. It then continues to iteratively move the points to the mean of their local diffusion until it converges. The hierarchical Clustering algorithm iteratively combines similar clusters until all available points belong to one cluster [112]. Spectral clustering finds the eigenvalues and eigenvectors of the matrix to cluster the point clouds, thus making it computationally expensive [113]. Fuzzy C-Means [114] allows for points to be a part of numerous clusters with different degrees of membership points in each cluster. The membership points and cluster centers are iterated. By representing the point clouds as a mixture of Gaussian distribution and computing the parameters of the diffusion through the expectation-maximization algorithm the probabilistic Gaussian Mixture Model [116] is able to cluster points in the point cloud. Balanced Iterative Reducing and Clustering using Hierarchy (BIRCH) clusters points by creating a hierarchical tree of subclusters. It continues to iterate points in the cluster using a hierarchical clustering algorithm and combines the subclusters, creating big clusters [117] [118].

3.5 Convex Hull Construction and Volume Calculation

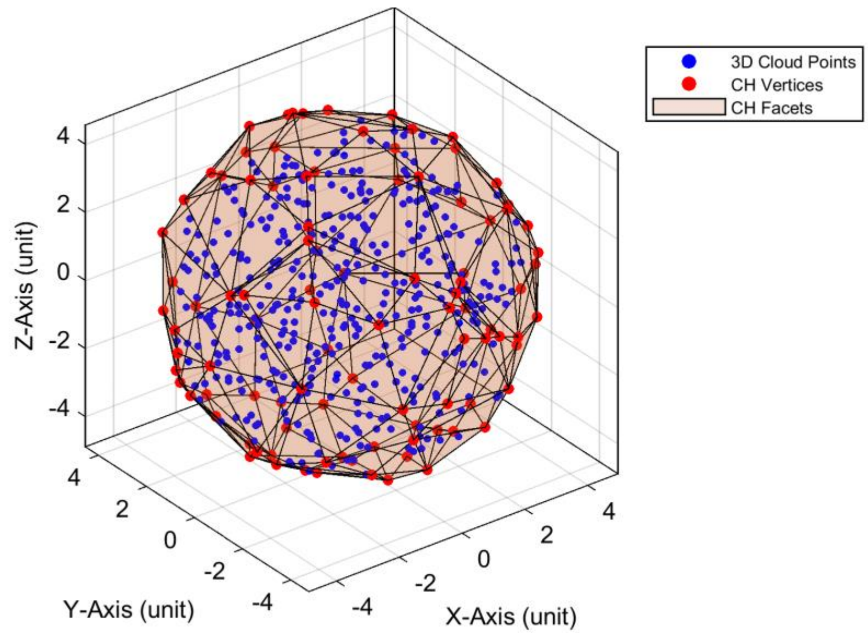


Figure 3.4 Convex Hull [134]

After clustering the point of interest, a convex hull fits over the point clouds, thus encompassing all the available points within it. Theoretically, a convex hull is the smallest polygon (polyhedron in 3D space) that can enclose a fixed number of points in a given plane. This method allows for the calculation of volume to compute elasticity and spatial data. Furthermore, to a certain extent, this method is resistant to issues caused by point cloud sparsity. Figure 3.4 shows a convex hull encompassing all the points within it and thus turning into a sphere construct a convex hull. Graham Scan [141] is one of the widely used algorithms to compute convex hull. It works by finding any one vertex in the convex hull and sorting the rest of the points as scanned from that vertex. Let the leftmost point of the convex hull be denoted by x_0 and mark the points left by angle from x_0 moving in counterclockwise encompassing $x_1, x_2, \dots, x_{n-1}$. Let $x_n = x_0$, if x_j is eliminated then for $i < j < k$ the points $x_i \rightarrow x_j \rightarrow x_k$ will form a right turn. Thus, x_j is enveloped inside the triangle (x_i, x_j, x_k) and not on the convex hull. The figure below visualizes this concept.

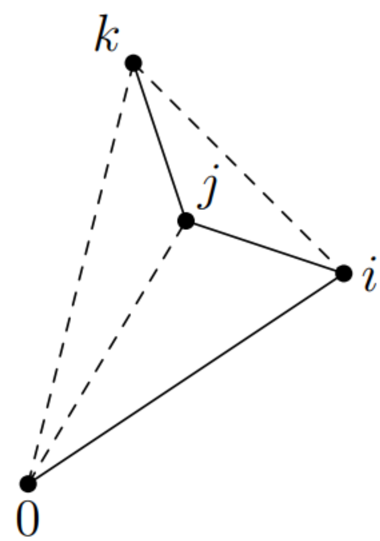


Figure 3.5 2D Convex Hull [141]

Every time the algorithm's while loop is activated a point is stacked or eliminated. At the highest the loop is activated $2n$ times as a point is only looked at once. The syntax of Graham Scan is attached below.

Graham Scan

1. Sort points by angle from x_0
2. Push x_0 and x_1. Set i=2
3. While $i \leq n$ do:

    If x_i makes left turn w.r.t. top 2 items on stack

      then { push x_i; i++ }

      else { pop and discard }

Figure 3.6 Syntax of Graham Scan [141]

The wrapping algorithm, also known as Jarvis March [139] [141] works by seeking out points in the order in which they appear. Let the leftmost point be x_0 and x_1 be the first point when viewed from x_0 in counterclockwise route. Similarly, let x_2 be the first point when viewed from $x_1$in counterclockwise route and so on.

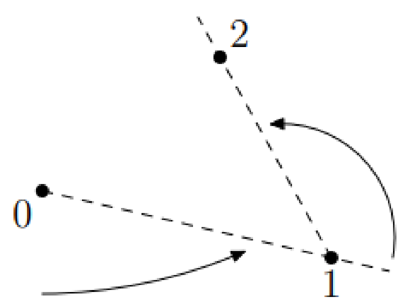


Figure 3.7 Jarvis March [141]

It takes linear time to find x_{i+1} and at the maximum, the while loop is activated h times, here h refers to the number of vertices on the convex hull, the algorithm is very robust but gets slow if there are too many points to process. The syntax of Jarvis March is attached below.

Jarvis March

$i = 0$

while not done do

  x_{i+1} = first point counterclockwise from x_i

Figure 3.8 Jarvis March syntax [141]

After fitting the convex hull, the volume is calculated by computing the volume of all the available tetrahedrons, as it is proved that convex polyhedra can always be tetrahedralizable [23]. Since the surface of the convex will always be made of triangular facets, an arbitrary point Q is fixed near the middle of the convex hull and imaginary lines from each vertex of the triangular facet are drawn to Q, thus creating a tetrahedron. This process is repeated for every triangular facet on the surface of the convex hull. Figure 3.4 shows a convex hull fitted over a spherical cluster of point clouds, notice the triangular facets making up the surface of the sphere.

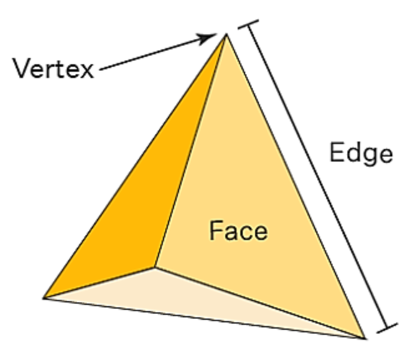


Figure 3.9 Tetrahedron [140]

Assuming Q as the arbitrary point and B, C, D as the vertexes of the facet, the volume of the tetrahedron can be calculated using,

$$v_1 = \frac{1}{6} |\{(\overrightarrow{QB} \times \overrightarrow{QC}) . \overrightarrow{QD}\}| \quad (3.1)$$

The final volume V of a single convex hull is calculated by summing up all the volumes of the available tetrahedrons.

$$V = v_1 + v_2 + v_3 + v_4 + \cdots \quad (3.2)$$

3.6 Tracking

After fitting the convex hull, the point cloud cluster in interest is tracked to observe their elasticity. Usually, tracking is done by extracting features and creating a 3D search map [143] or finding the cosine similarity between the template and search branch [142]. But in this thesis tracking is achieved by finding the centroid of a convex hull and finding the Euclidean distance between the convex hull's centroid in the successive frame. The algorithm finds Euclidean distance with all the clusters encompassed with convex hull and selects the cluster that has the least distance.

The centroid is the mean of all the coordinates in the 3D space, it can be mathematically represented as,

$$C_x = \frac{1}{n}\sum_{i=1}^{n} x_i \, , C_y = \frac{1}{n}\sum_{i=1}^{n} y_i \, , C_z = \frac{1}{n}\sum_{i=1}^{n} z_i \quad (3.3)$$

$$C_1 = (C_x, C_y, C_z) \quad (3.4)$$

C_x, C_y, C_z refers to the coordinates of the centroid, n refers to the total number of vertices on the surface of the convex hull and C represents the calculated centroid of the convex hull. As explained, after calculating the centroid of all the convex hulls present in frame one and the frame subsequent to it, the Euclidean distance between the points is computed. Below Euclidean distance is found for two centroid points $C_1$and C_2,

$$d = \sqrt{(C_{2x} - C_{1x})^2 + (C_{2y} - C_{1y})^2 + (C_{2z} - C_{1z})^2} \quad (3.5)$$

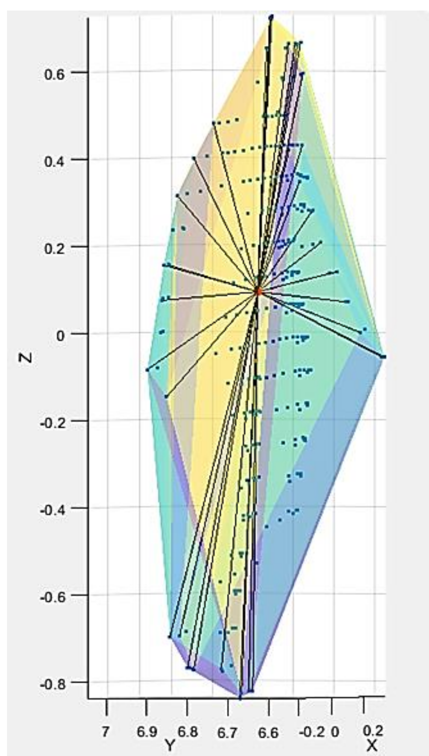


Figure 3.10 Finding the centroid of a convex hull.

The tracking method proposed in this thesis was not compared to other tracking methods as 3D tracking is not the focus of this thesis, regardless this methodology suited this scenario. The results of this methodology are attached in the next section.

3.7 Elasticity & Spatial Data

Elasticity and Spatial data are the two features that are observed while tracking the cluster of interest for t amount of time. These two are considered as the features to predict a class.

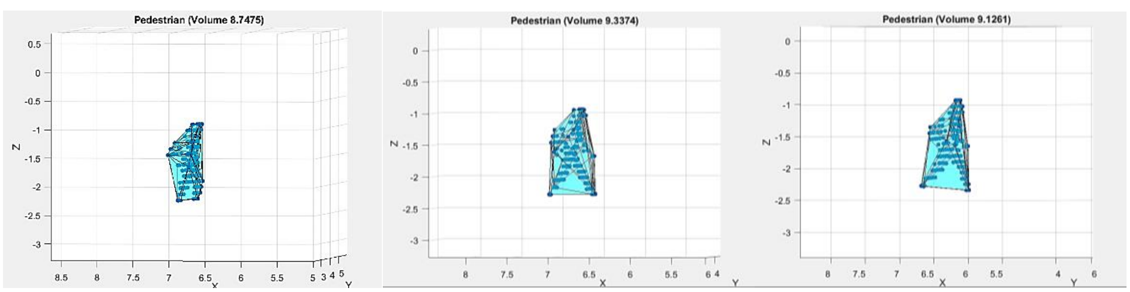


Figure 3.11 Visualizing Elasticity

The figure above shows a pedestrian walking in successive frames. As the pedestrian moves, their elasticity value starts out at 8.7475, increases to 9.3374, and drops to 9.1261. Elasticity can be mathematically represented as,

$S_i(t)$ is the set of all the points for an actor i at time t.

$C(S_i(t)\ )$ is defined as the convex hull function which generates a convex hull for a set of points S_i and $F(.)$ is defined as the function to calculate the volume of the convex hull.

$$f(C(S(t))) = Volume\ of\ all\ points\ creating\ the\ convex\ hull\ at\ time\ t \quad (3.6)$$

$$V(1) = f(C(S_i(1))) \quad (3.7)$$

$$V(2) = f(C(S_i(2))) \quad (3.8)$$

$$V(3) = f(C(S_i(3))) \quad (3.9)$$

$$V(t) = f(C(S_i(n))) \quad (3.10)$$

$$E = \{V(1), V(2), V(3), \dots, V(t)\} \quad (3.11)$$

The final spatial values are represented in a set (E).

The figure below shows the accumulation of the point clouds as time t increases. This results in a steady increase in volume depending on their pace as the pedestrian walks: 8.745, 11.0764 & 12.506.

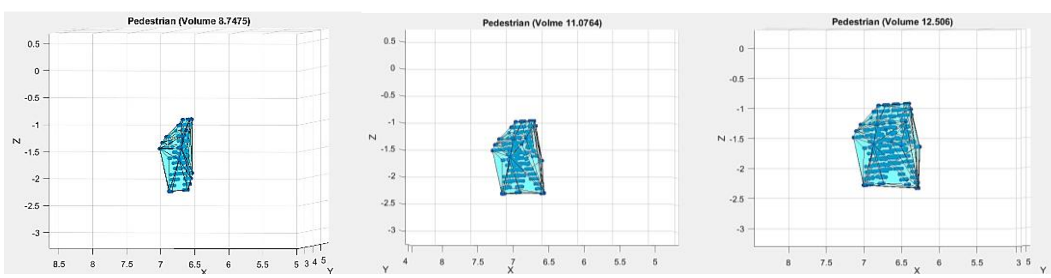


Figure 3.12 Visualizing Spatial Data

$$K(1) = f(C(\{S_i(1)\})) \quad (3.12)$$

$$K(2) = f(C(\{S_i(1) \cup S_i(2)\})) \quad (3.13)$$

$$K(3) = f(C(\{S_i(1) \cup S_i(2) \cup S_i(3)\})) \quad (3.14)$$

$$K(t) = f(C(\{S_i(1) \cup s(2) \cup S_i(3) \ \cup S_i(n)\})) \quad (3.15)$$

$$J = \{K(1), K(2), K(3), \ldots, K(t)\} \quad (3.16)$$

The final spatial values are represented in a set (J). Before making any predictions, the classifier algorithm creates its own set (M) where the values of the set E and J are interwoven.

$$M = \{E(1), K(1), E(2), K(2), E(3), K(3), \ldots, E(t), K(t)\} \quad (3.17)$$

The classifier algorithm learns to find the temporal dependency between the values presented in the set M.

3.8 Classifier

The classifier block is the last addition to the architecture. This part is responsible for reviewing the recorded elasticity & spatial data and identifying the temporal dependency between the recorded values. Once the pattern is noticed the algorithm proceeds to classify the class of the identified cluster. This thesis explores three such algorithms that can explore temporal dependence patterns: Random Forest [131], XG Boost [133], and Gradient Boost [135]. All these three algorithms are explained further in the subsections below.

3.8.1 Random Forest

Introduced by Leo Breiman in 2001, the random forest [131] is part of the ensemble learning models. Here, different models of the same algorithm come together to make a prediction. This makes the algorithm more resistant to overfitting, predicts better even with missing values, and makes parallelization possible, thus lesser training times. The algorithm works by assembling n number of decision trees, to make sure that each decision tree has its own perspective random feature selection is utilized. This also makes sure that the algorithm is trained from a diverse dataset. This is then followed by bagging, resulting in the creation of a separate subset of data for each decision tree thus increasing variability and making the model more robust. All the decision trees then proceed to cast individual votes to make a prediction, the final prediction is based on the mode across all the trees [24] [29]. The concept is well explained through the flow chart attached below.

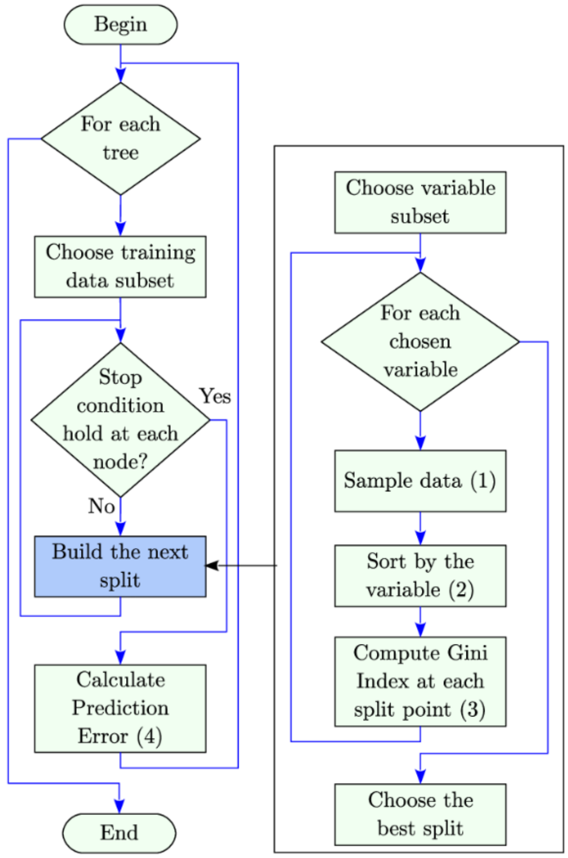


Figure 3.13 Random Forest flowchart. [144]

3.8.2 XG Boost

Extreme Gradient Boosting, widely known as XG Boost, is a supervised machine learning algorithm based on boosting. Like random forest, this algorithm also uses decision trees and is a part of the ensemble learning family. But unlike random forest, each new tree is built to reduce the residual error of the tree built previously. A regularization function is included to avoid overfitting, early stoppage is also supported. Since it requires building trees by learning the mistakes of the previous tree the process cannot be parallelized, making it computationally slower than random forest. However, this also allows it to perform better with datasets that have missing values as each new tree learns to predict the missing values [133] [25]. The figure below shows the working of XG Boost. You can see the residual value being passed down subsequently, this allows the newer trees to learn and perform better than the previous ones.

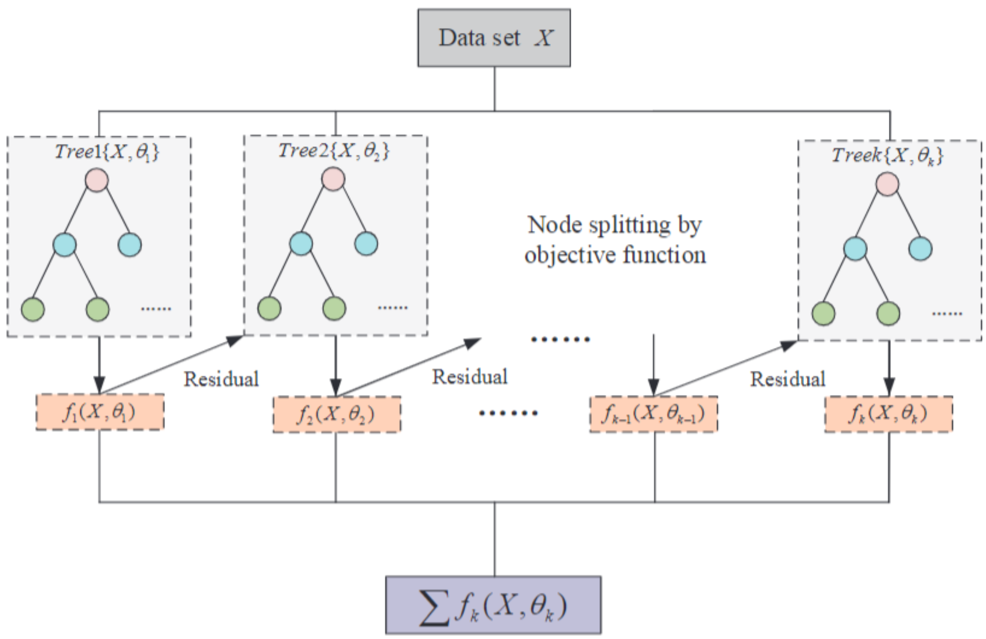


Figure 3.14 XG Boost flowchart. [133]

The algorithm can be mathematically expressed as,

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \quad (3.18)$$

With $\hat{y}_i$ being the predicted value, $f_k(x_i)$ represents the function of input in k-th decision tree, K marks the total number of decision trees and *F* is the set of all the possible values.

3.8.3 Gradient Boost

Gradient Boost is a toned-down version of XG Boost. They both belong to the same family of algorithms, but Gradient Boost does not come with regularization terms like L1 (Lasso) and L2 (Ridge) or early stopping techniques to reduce overfitting [30].

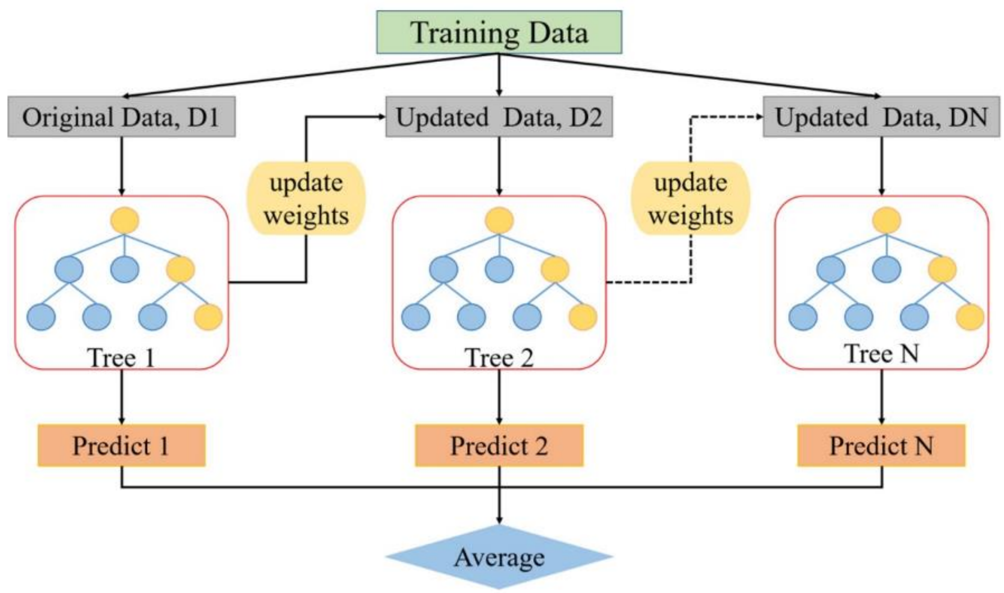


Figure 3.15 Gradient Boost flowchart. [135]

The flowchart attached above explains the algorithm. As you can see, the working is very similar to the gradient boost algorithm.

Chapter 4 Case Study

4.1 Introduction

With the methodology explained in the previous chapter, we will explore and discuss the results obtained and the issues faced while working with real dataset.

4.2 Data Collection

Figure 4.1 Data collection

Before going into the results, it is crucial to understand how the dataset was collected. The data collection team (Edison Li, Saswat Priyadarshi Nayak & Xuanpeng Zaho) used Ouster OS1-128 to collect the point cloud data. The LiDAR sensor was set on a tripod in the afternoon hours in front of the Bourns College of Engineering. The recorded dataset contains rich pedestrian and micromobility interactions. The frame rate was set at 10 Hz. Using the methodology discussed in the previous chapter, 202 sets of elasticity and spatial of pedestrians and e-scooter users were collected. Each set has eight values of elasticity and spatial data corresponding to a single actor. In other words, each actor had 16 values associated with them, thus the machine learning algorithms used had 16 features to differentiate between a class. Since the dataset was captured at 10 HZ, collecting 8 pairs of values accounted for 800 milliseconds. Thus, the algorithm takes 1/8 of a second to differentiate between a class. This thesis shows the results of the models trained with 8 pairs of elasticity and spatial data, depending on the dataset available this number can be decreased or increased. Below are the results of the algorithms used. Figure 4.2 shows three pedestrians walking towards the LiDAR and two pedestrians walking away from the sensor. For better visibility two pedestrians approaching the LiDAR are highlighted in red.

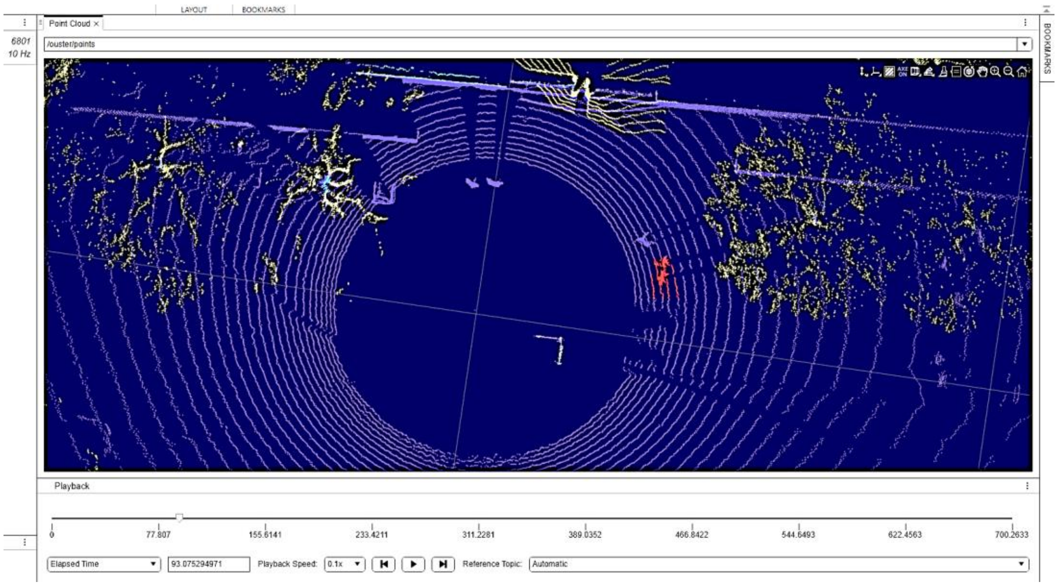


Figure 4.2 Visualizing the dataset

4.3 Visualizing Plane Elimination and Clustering Algorithms

It is imperative that the algorithm eliminates the ground plane before processing further. Figure 4.3 shows a pedestrian segmented with the ground plane; it is easy to observe that if the ground plane is included most of the elasticity features exhibited by the class will be left unnoticed as the convex hull envelops a broader area. This thesis uses RANSAC to identify planes, the algorithm's robust nature allows for quick ground plane elimination without disturbing the actors. The green dots represented in Figure 4.4 are the segmented plane in a point cloud scene.

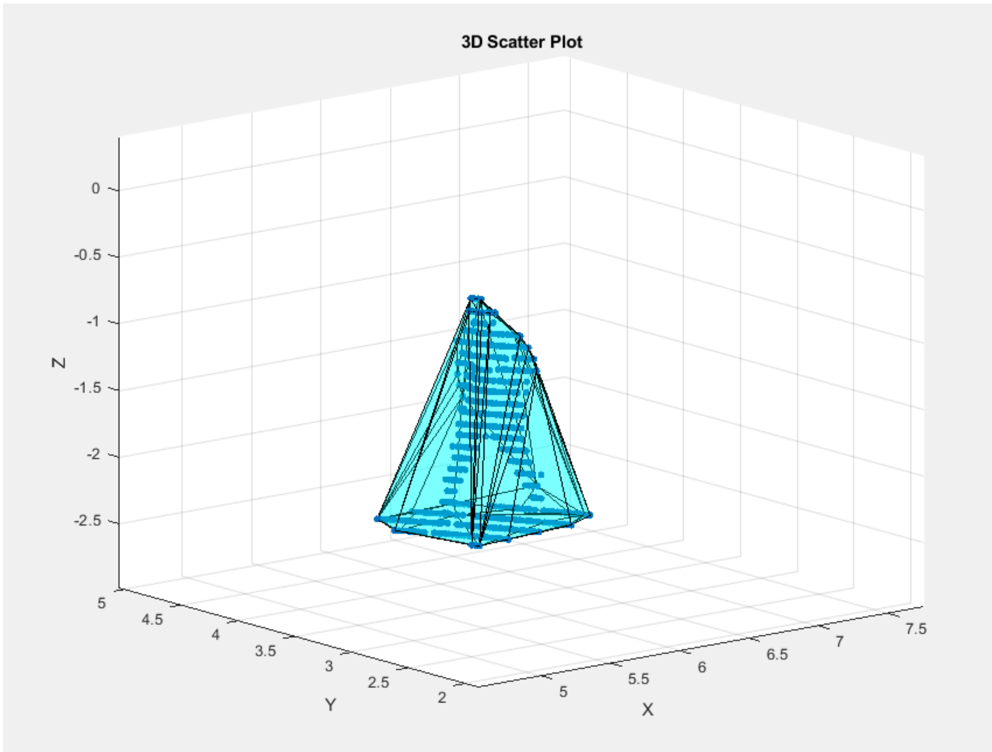


Figure 4.3 Pedestrian with ground plane

As mentioned previously, this thesis uses DBSCAN to cluster the points. After filtering out the ground plane the point clouds are sent for clustering. Clustering allows us to filter out any stray points that might affect the volume calculation in the next step. It also allows for isolating multiple actors in a scene. Figure 4.5 shows DBSCAN clustering two pedestrians into two respective clusters and eliminating the stray points in the bottom left corner.

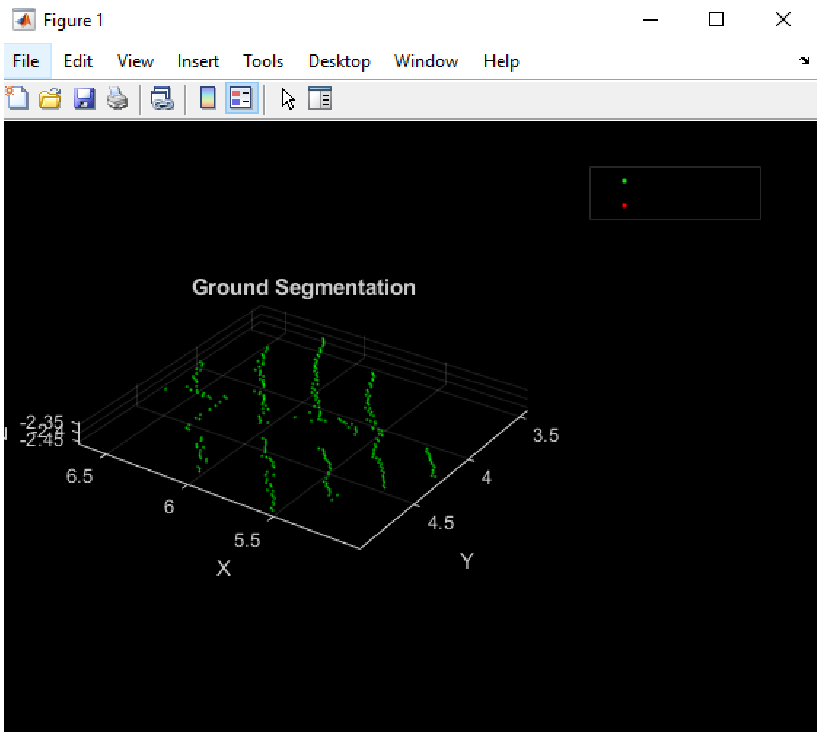


Figure 4.4 Segmented ground plane.

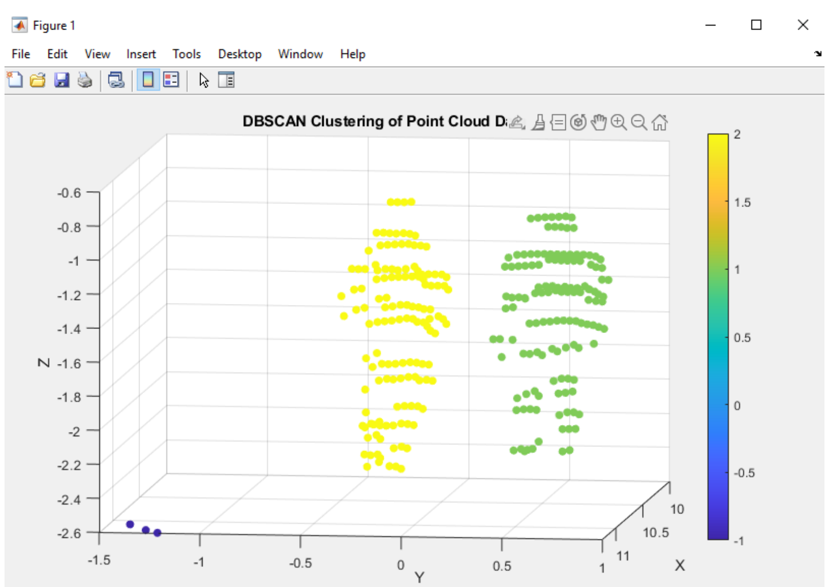


Figure 4.5 DBSCAN clustering

4.4 Visualizing Convex Hull Construction and Tracking Algorithms

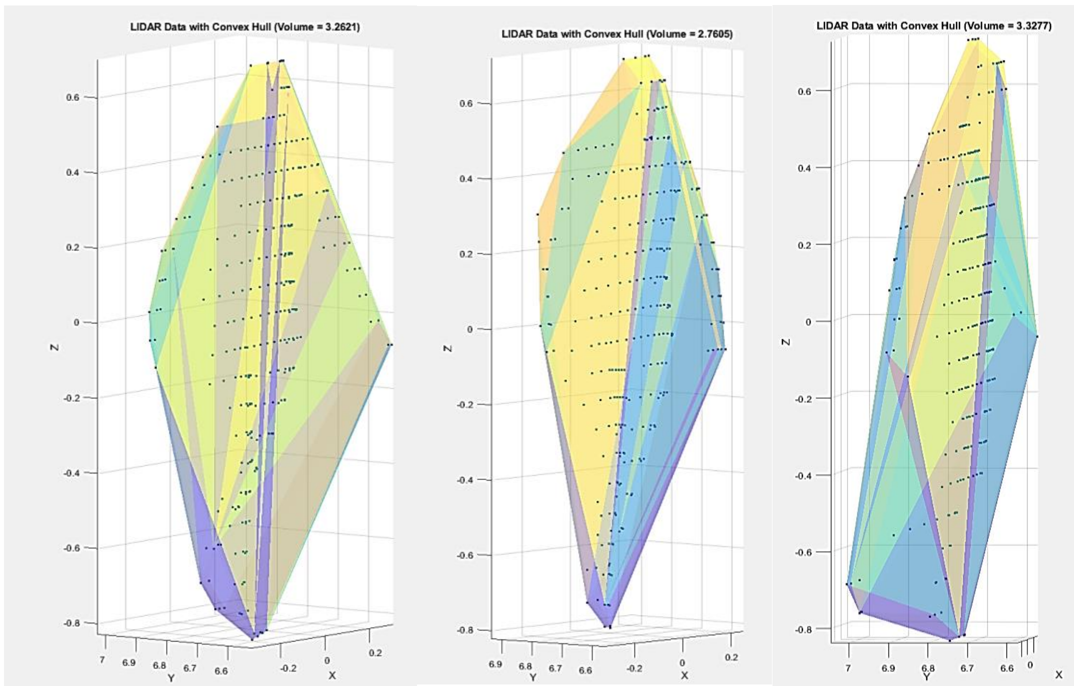


Figure 4.6 Visualizing change in pose and volume of a pedestrian walking.

Convex hull is used to find the volume of the cluster, Figure 4.6 shows a pedestrian wrapped in a convex hull in three successive frames. You can notice the change in volume as the pedestrian expands and contracts their limbs to generate motion, thus exhibiting elasticity as explained earlier. Figure 4.7 shows the tracking algorithm in action, the green point shown in the figure represents the calculated centroid and the red shade represents the convex hull wrapping the point clouds. The two clusters represented belong to a single pedestrian in consecutive frames, the taskbar below pictures read “Euclidean distance between centroids: 0.24478.” This distance will dramatically be larger for any other convex hull apart from its own in consecutive frames.

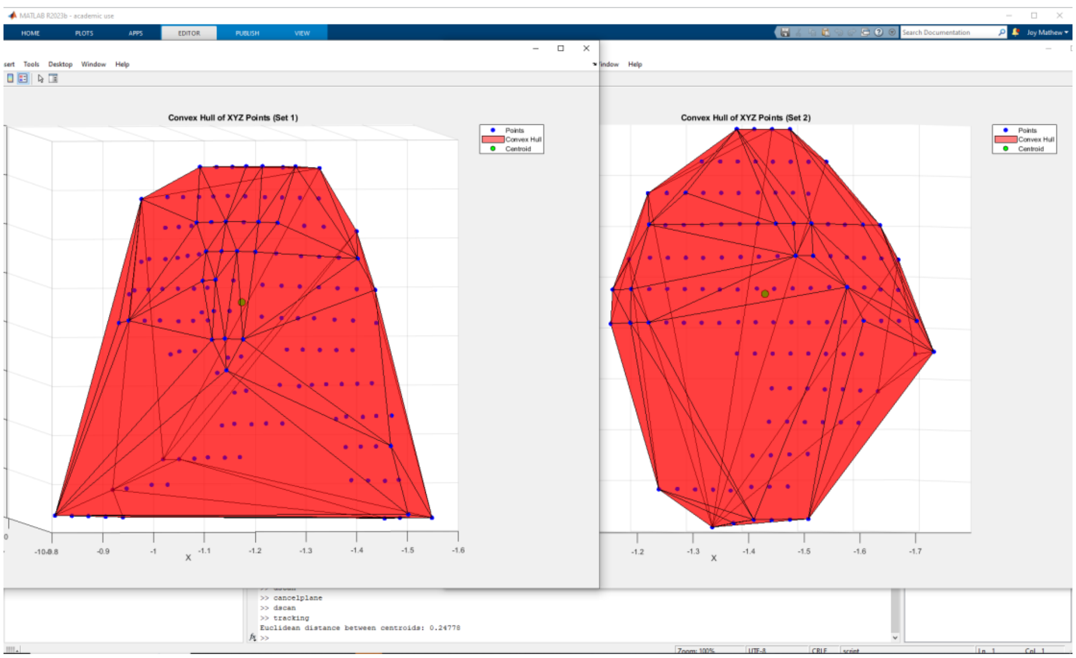


Figure 4.7 Finding Euclidean distance between two clusters in subsequent frames.

4.5 Visualizing Elasticity and Spatial Data

In this subsection, the elasticity and spatial data are visualized by charting them on the graphs presented below. Figure 4.8 shows the elasticity graph of an e-scooter user. But although micromobility users do not necessarily move their body elasticity is still observed. This is partly due to the noise as it disturbs the cluster's volume. Below attached is a graph (fig. 4.8) plotted between the time and volume of an e-scooter rider approaching the LiDAR sensor. You will be able to observe the reduction in volume of the convex hull of the rider.

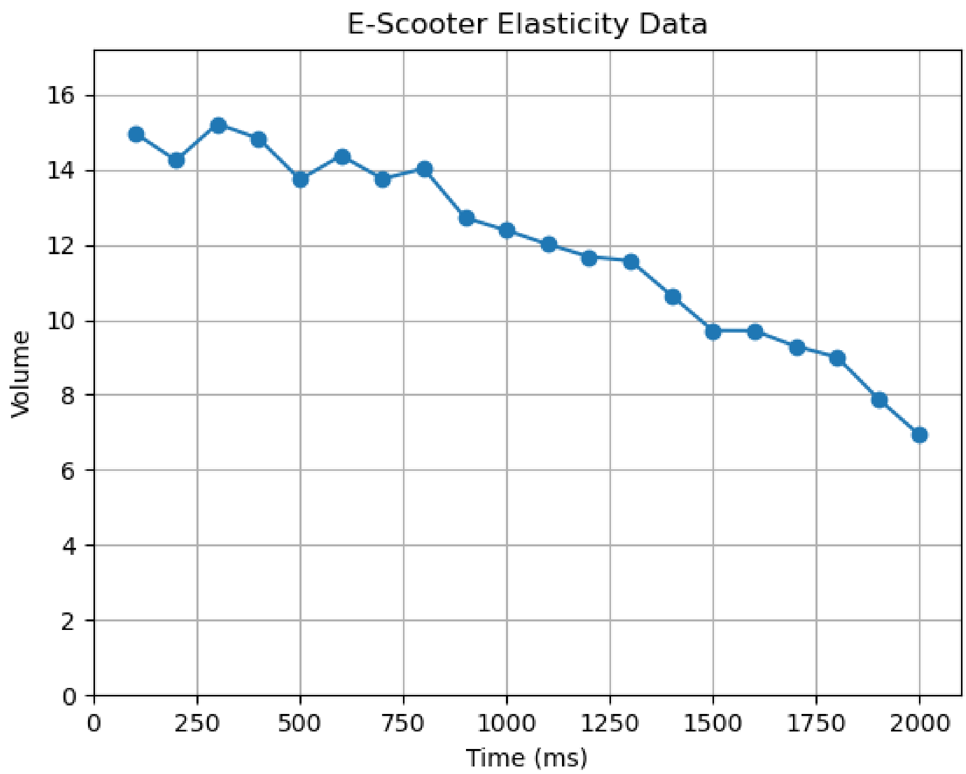


Figure 4.8 E-scooter approaching the LiDAR.

Figure 4.9 attached below shows the graph of a rider moving away from the LiDAR sensor. Notice the increase in volume of the convex hull as time moves forward. It can be observed that the slope of the graph is not even, this is due to the noises encountered by the clustering algorithm.

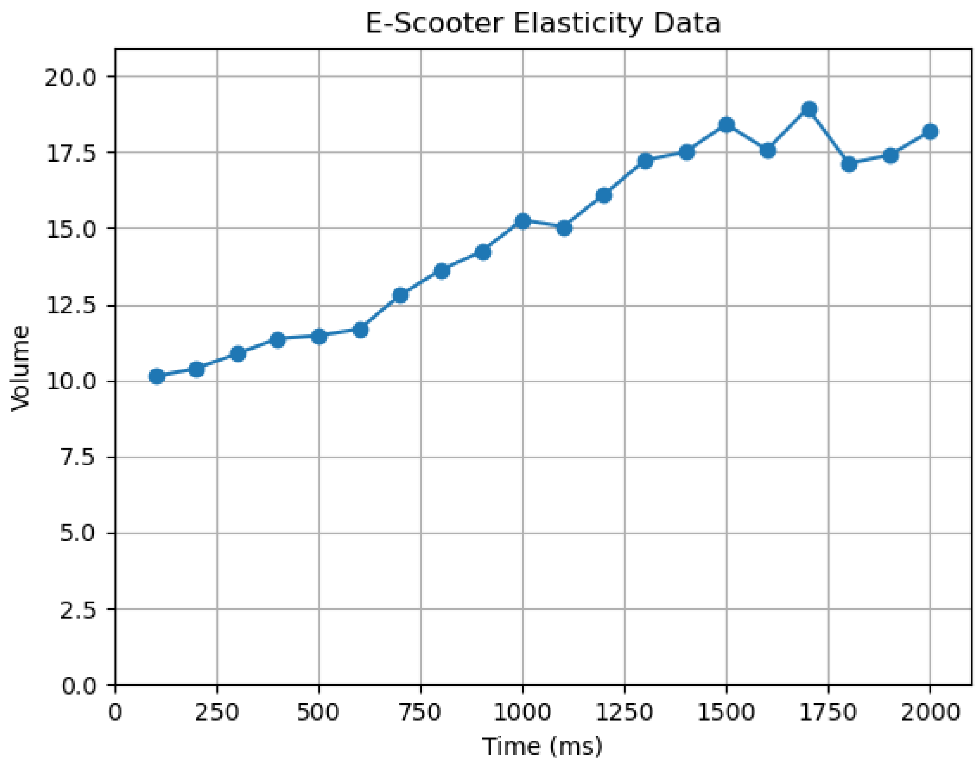


Figure 4.9 E-scooter departing from the LiDAR.

The same phenomenon is observed in pedestrian class also, Figure 4.10 shows the change in volume of a pedestrian approaching the LiDAR. Figure 4.11 shows the change in volume of a pedestrian departing from the LiDAR sensor. Notice how the pedestrian graphs have a lot of peaks. This continuous rise and fall of the peaks represent the change in the elasticity of a walking pedestrian. Since the micromobility riders exhibit little to no movement this is not observed in their graphs.

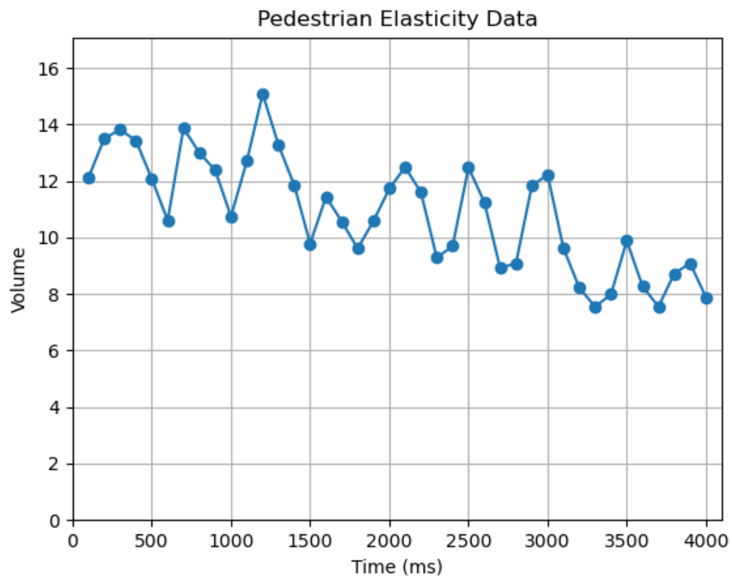


Figure 4.10 Pedestrian approaching the LiDAR.

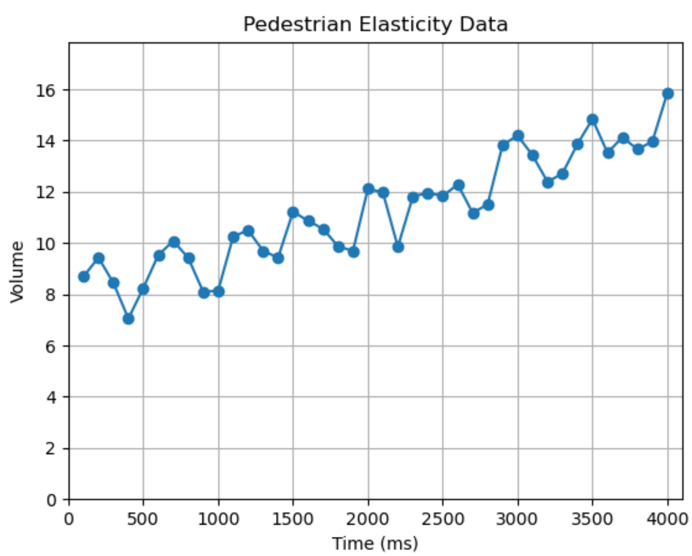


Figure 4.11 Pedestrian departing from the LiDAR.

In some extremely noisy cases, the graph plotted between time and volume tends to exhibit more unwanted peaks which can result in false positives. Figure 4.12 shows an e-scooter user departing from the LiDAR. The abnormal peaks observed can be due to noise clustered by DBSCAN as part of the main cluster. The steep fall of the points indicates that the cluster has suffered from partial occlusion.

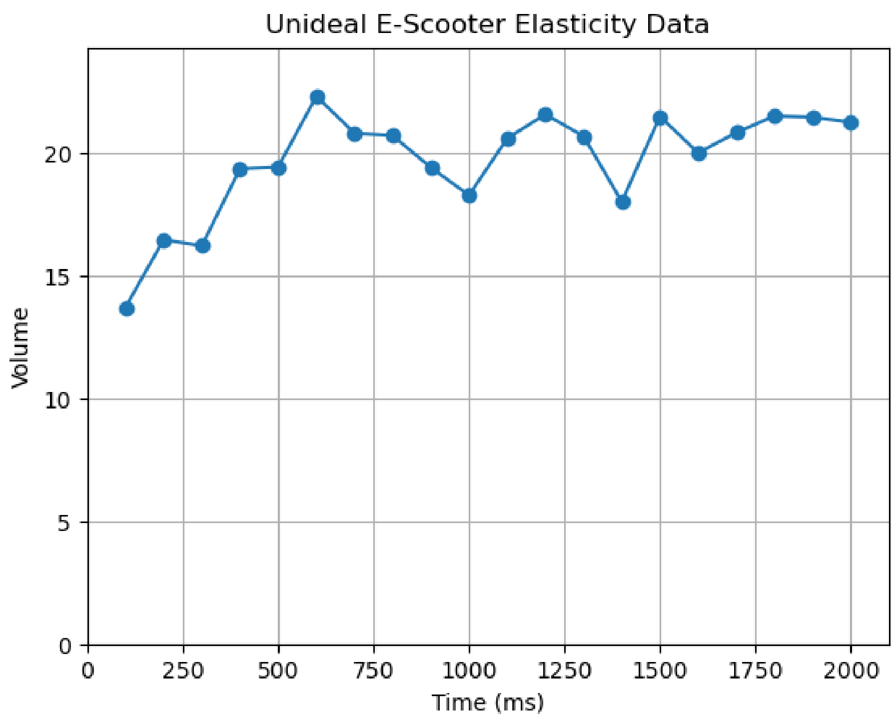


Figure 4.12 Unideal e-scooter data (departing from the LiDAR).

Figure 4.13 shows a case of noisy pedestrian data. Some points in the point cloud could have disappeared thus causing the volume to steeply fall (possibility of partial occlusion). The first tall peak might have been caused due to clustering algorithm allowing stray points in the cluster. Unideal sets of data are not rare, training an algorithm with only elasticity data feature is bound to perform poorly. To avoid this scenario, the algorithm considers a second feature (spatial data) before classifying.

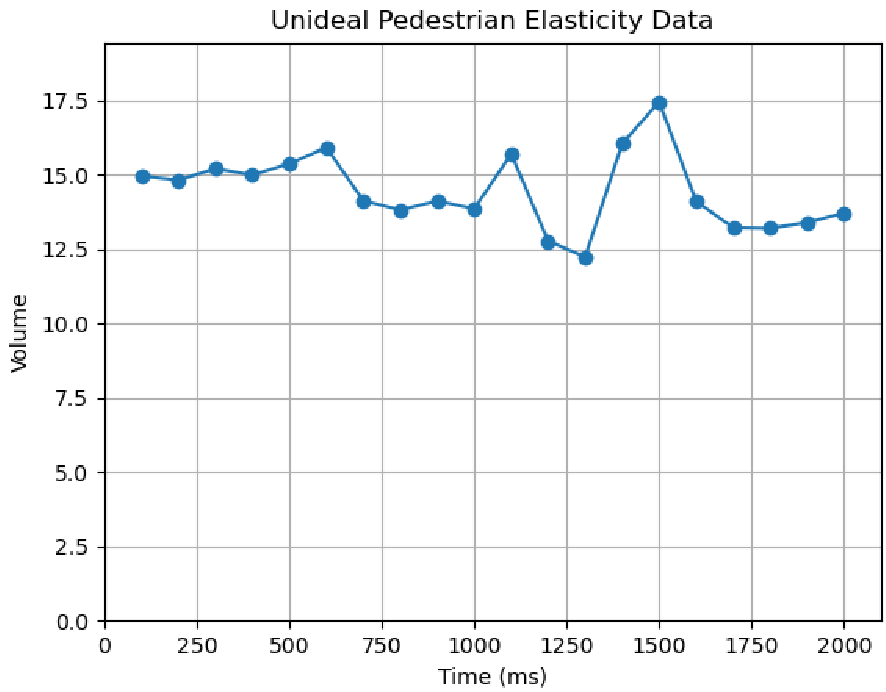


Figure 4.13 Unideal pedestrian data (approaching LiDAR).

As mentioned previously, spatial data is the accumulation of point cloud of a specific cluster over a period of *t* time. Figure 4.14 shows the increase in volume of an e-scooter rider in 2s. In Figure 4.15 a pedestrian's increase in volume is shown for 2s. The algorithm learns to differentiate between these two curves by noticing the pattern of jumps in volume. Since pedestrians move slower compared to e-scooters their change in volume is not very significant. As a comparison, the pedestrian's volume starts at 14 and ends at 64 by the end of 2s. For the same time frame, the e-scooter starts at 19 and ends at 178. Comparatively, this factor is less likely to be affected by noise.

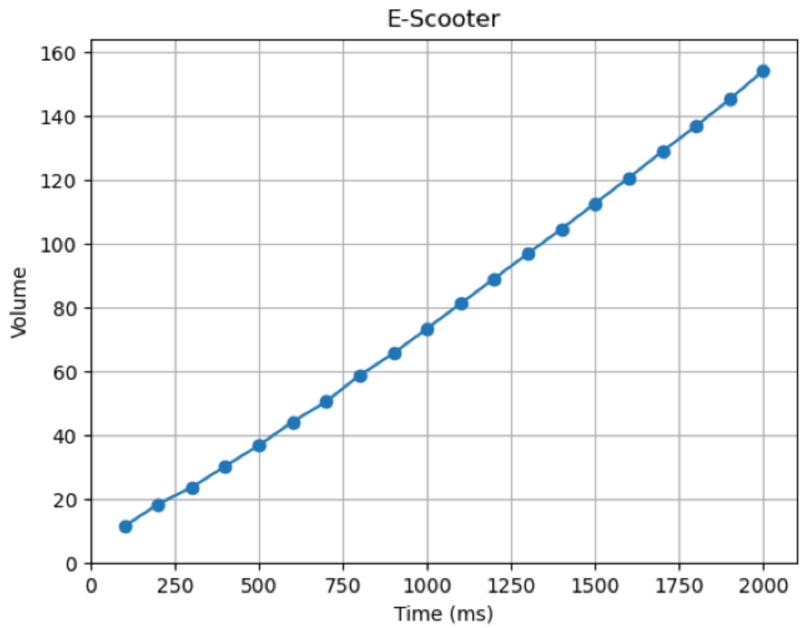


Figure 4.14 E-Scooter Spatial Data

But if the rider decides to drive their scooter to match the pace of an adjacent pedestrian, then the model will not be able to tell the difference. Also, if an actor is too close to the LiDAR sensor physical movements may not be well observed, this will result in slight volume changes that only result in small peaks. These small peaks can technically be present in e-scooter elasticity graph due to noise. Since point clouds can get partially or fully occluded in some cases utilizing two distinct types of features to classify improves the chances of true positives. Thus, elasticity data and spatial is considered before classification.

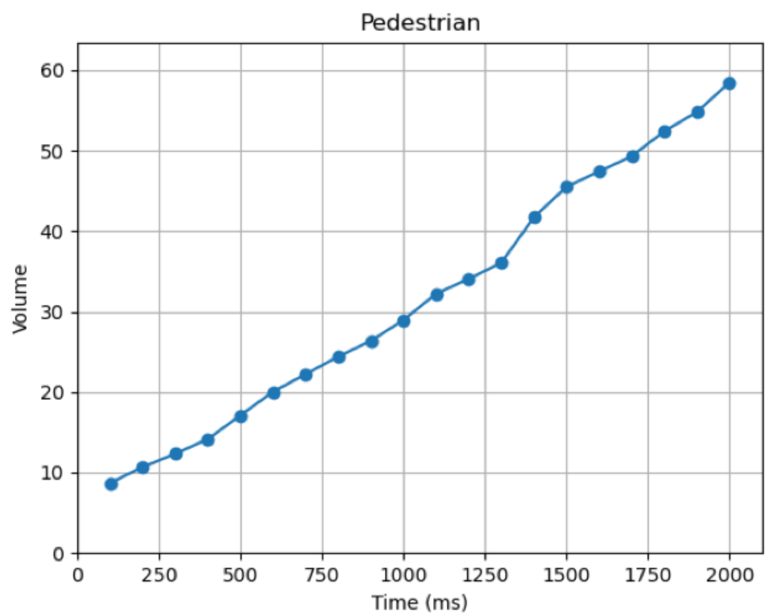


Figure 4.15 Pedestrian Spatial Data

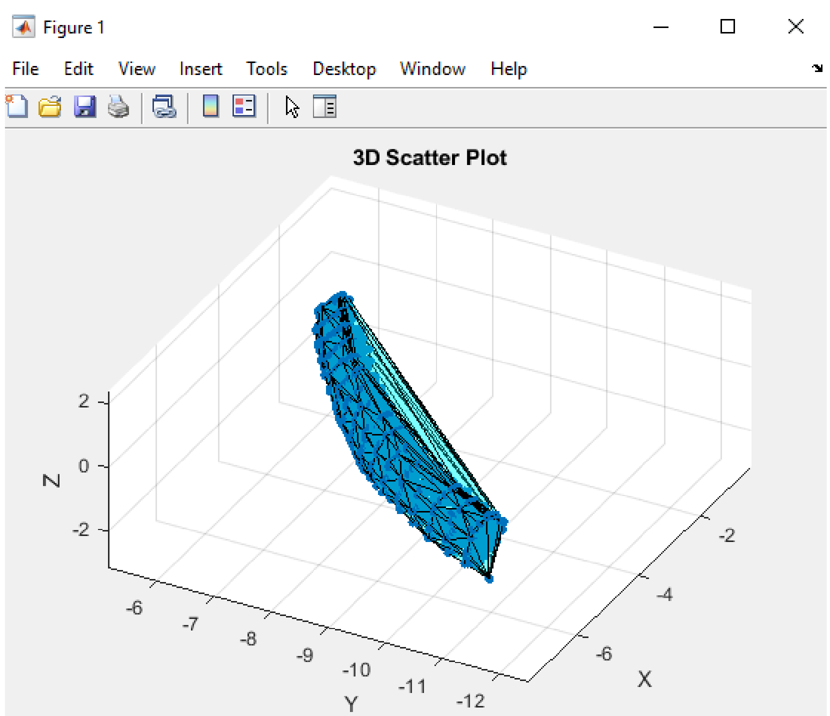


Figure 4.16 E-Scooter

Figure 4.16 shows the spatial data of an e-scooter user for T= 2s. As the actor moves in the field their clusters accumulate with respect to their previous cluster at T-1. Figure 4.17 shows the spatial data of a pedestrian for T= 3s.

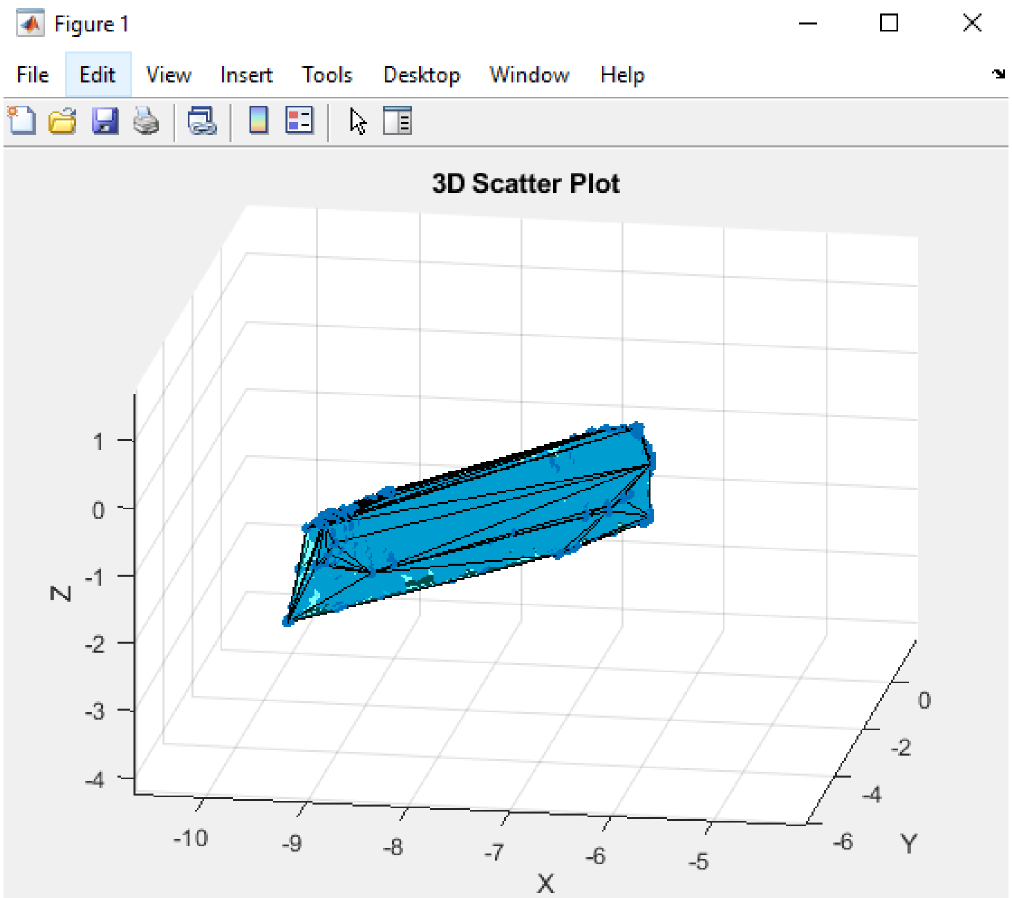


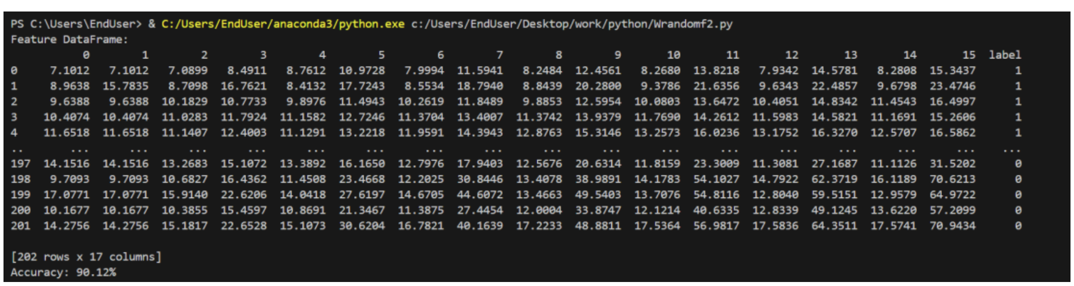
Figure 4.17 Pedestrian

4.6 Result Analysis

Three algorithms namely Random Forest, XG Boost and Gradient Boost are trained with the same dataset and their results are analyzed by plotting a confusion matrix. The actual picture of the terminal is also attached, it shows the features used for training and also depicts the accuracy of the model.

4.6.1 Random Forest Results

Trained with 16 features (eight sets of elasticity data & eight sets of spatial data). Pedestrian and scooter classes each had 101 sets of data, bringing the total to 202 sets as you can see in Figure 4.18. N estimator = 81000 and random state = 43.



```
PS C:\Users\EndUser> & C:/Users/EndUser/anaconda3/python.exe c:/Users/EndUser/Desktop/work/python/Wrandomf2.py
Feature DataFrame:
           0        1        2        3        4        5        6        7        8        9       10       11       12       13       14       15  label
0     7.1012   7.1012   7.0899   8.4911   8.7612  10.9728   7.9994  11.5941   8.2484  12.4561   8.2680  13.8218   7.9342  14.5781   8.2808  15.3437      1
1     8.9638  15.7835   8.7098  16.7621   8.4132  17.7243   8.5534  18.7940   8.8439  20.2800   9.3786  21.6356   9.6343  22.4857   9.6798  23.4746      1
2     9.6388   9.6388  10.1829  10.7733   9.8976  11.4943  10.2619  11.8489   9.8853  12.5954  10.0803  13.6472  10.4051  14.8342  11.4543  16.4997      1
3    10.4074  10.4074  11.0283  11.7924  11.1582  12.7246  11.3704  13.4007  11.3742  13.9379  11.7690  14.2612  11.5983  14.5821  11.1691  15.2606      1
4    11.6518  11.6518  11.1407  12.4003  11.1291  13.2218  11.9591  14.3943  12.8763  15.3146  13.2573  16.0236  13.1752  16.3270  12.5707  16.5862      1
..       ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...    ...
197  14.1516  14.1516  13.2683  15.1072  13.3892  16.1650  12.7976  17.9403  12.5676  20.6314  11.8159  23.3009  11.3081  27.1687  11.1126  31.5202      0
198   9.7093   9.7093  10.6827  16.4362  11.4508  23.4668  12.2025  30.8446  13.4078  38.9891  14.1783  54.1027  14.7922  62.3719  16.1189  70.6213      0
199  17.0771  17.0771  15.9140  22.6206  14.0418  27.6197  14.6705  44.6072  13.4663  49.5403  13.7076  54.8116  12.8040  59.5151  12.9579  64.9722      0
200  10.1677  10.1677  10.3855  15.4597  10.8691  21.3467  11.3875  27.4454  12.0004  33.8747  12.1214  40.6335  12.8339  49.1245  13.6220  57.2099      0
201  14.2756  14.2756  15.1817  22.6528  15.1073  30.6204  16.7821  40.1639  17.2233  48.8811  17.5364  56.9817  17.5836  64.3511  17.5741  70.9434      0

[202 rows x 17 columns]
Accuracy: 90.12%
```

Figure 4.18 Random Forest

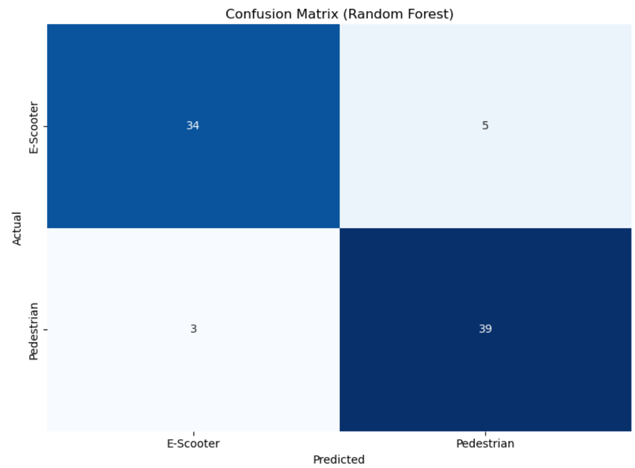


Figure 4.19 Confusion Matrix of Random Forest

The confusion matrix for Random Forest is presented in Figure 4.19. The figure shows the result of the test data. Out of the original sets (202) that were recorded, 40% of the data (81) was used as test sets for the algorithm.

4.6.2 XG Boost Results

Trained with 16 features. Pedestrian and scooter classes each had 101 sets of data, bringing the total to 202 sets as you can see in Figure 4.20. N estimator = 81000 and random state = 43.

```
PS C:\Users\EndUser> & C:/Users/EndUser/anaconda3/python.exe c:/Users/EndUser/Desktop/work/python/xg.py
Feature DataFrame:
           0        1        2        3        4        5        6        7        8        9       10       11       12       13       14       15  label
0     7.1012   7.1012   7.0899   8.4911   8.7612  10.9728   7.9994  11.5941   8.2484  12.4561   8.2680  13.8218   7.9342  14.5781   8.2808  15.3437      1
1     8.9638  15.7835   8.7098  16.7621   8.4132  17.7243   8.5534  18.7940   8.8439  20.2800   9.3786  21.6356   9.6343  22.4857   9.6798  23.4746      1
2     9.6388   9.6388  10.1829  10.7733   9.8976  11.4943  10.2619  11.8489   9.8853  12.5954  10.0803  13.6472  10.4051  14.8342  11.4543  16.4997      1
3    10.4074  10.4074  11.0283  11.7924  11.1582  12.7246  11.3704  13.4007  11.3742  13.9379  11.7690  14.2612  11.5983  14.5821  11.1691  15.2606      1
4    11.6518  11.6518  11.1407  12.4003  11.1291  13.2218  11.9591  14.3943  12.8763  15.3146  13.2573  16.0236  13.1752  16.3270  12.5707  16.5862      1
..       ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...    ...
197  14.1516  14.1516  13.2683  15.1072  13.3892  16.1650  12.7976  17.9403  12.5676  20.6314  11.8159  23.3009  11.3081  27.1687  11.1126  31.5202      0
198   9.7093   9.7093  10.6827  16.4362  11.4508  23.4668  12.2025  30.8446  13.4078  38.9891  14.1783  54.1027  14.7922  62.3719  16.1189  70.6213      0
199  17.0771  17.0771  15.9140  22.6206  14.0418  27.6197  14.6705  44.6072  13.4663  49.5403  13.7076  54.8116  12.8040  59.5151  12.9579  64.9722      0
200  10.1677  10.1677  10.3855  15.4597  10.8691  21.3467  11.3875  27.4454  12.0004  33.8747  12.1214  40.6335  12.8339  49.1245  13.6220  57.2099      0
201  14.2756  14.2756  15.1817  22.6528  15.1073  30.6204  16.7821  40.1639  17.2233  48.8811  17.5364  56.9817  17.5836  64.3511  17.5741  70.9434      0

[202 rows x 17 columns]
Accuracy: 90.12%
```

Figure 4.20 XG Boost

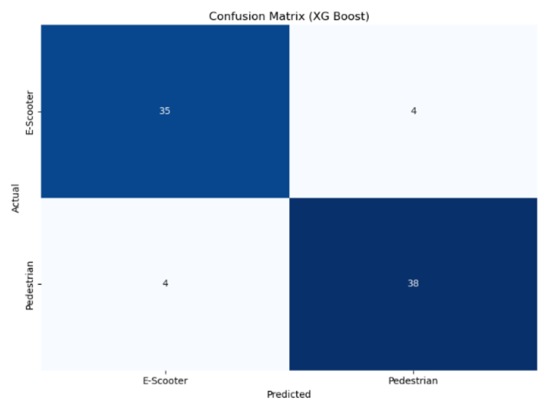


Figure 4.21 Confusion Matrix of XG Boost

Figure 4.20 presents the confusion matrix of the Gradient Boost algorithm. Out of the 81 sets of data four sets were misclassified as e-scooters and four as pedestrians.

4.6.3 Gradient Boost Results

```
PS C:\Users\EndUser> & C:/Users/EndUser/anaconda3/python.exe "c:/Users/EndUser/Desktop/work/python/gradient b.py"
Feature DataFrame:
           0        1        2        3        4        5        6        7        8        9       10       11       12       13       14       15  label
0     7.1012   7.1012   7.0899   8.4911   8.7612  10.9728   7.9994  11.5941   8.2484  12.4561   8.2680  13.8218   7.9342  14.5781   8.2808  15.3437      1
1     8.9638  15.7835   8.7098  16.7621   8.4132  17.7243   8.5534  18.7940   8.8439  20.2800   9.3786  21.6356   9.6343  22.4857   9.6798  23.4746      1
2     9.6388   9.6388  10.1829  10.7733   9.8976  11.4943  10.2619  11.8489   9.8853  12.5954  10.0803  13.6472  10.4051  14.8342  11.4543  16.4997      1
3    10.4074  10.4074  11.0283  11.7924  11.1582  12.7246  11.3704  13.4007  11.3742  13.9379  11.7690  14.2612  11.5983  14.5821  11.1691  15.2606      1
4    11.6518  11.6518  11.1407  12.4003  11.1291  13.2218  11.9591  14.3943  12.8763  15.3146  13.2573  16.0236  13.1752  16.3270  12.5707  16.5862      1
..       ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...    ...
197  14.1516  14.1516  13.2683  15.1072  13.3892  16.1650  12.7976  17.9403  12.5676  20.6314  11.8159  23.3009  11.3081  27.1687  11.1126  31.5202      0
198   9.7093   9.7093  10.6827  16.4362  11.4508  23.4668  12.2025  30.8446  13.4078  38.9891  14.1783  54.1027  14.7922  62.3719  16.1189  70.6213      0
199  17.0771  17.0771  15.9140  22.6206  14.0418  27.6197  14.6705  44.6072  13.4663  49.5403  13.7076  54.8116  12.8040  59.5151  12.9579  64.9722      0
200  10.1677  10.1677  10.3855  15.4597  10.8691  21.3467  11.3875  27.4454  12.0004  33.8747  12.1214  40.6335  12.8339  49.1245  13.6220  57.2099      0
201  14.2756  14.2756  15.1817  22.6528  15.1073  30.6204  16.7821  40.1639  17.2233  48.8811  17.5364  56.9817  17.5836  64.3511  17.5741  70.9434      0

[202 rows x 17 columns]
Accuracy: 86.42%
```

Figure 4.22 Gradient Boost

Trained with 16 features. Pedestrian and scooter classes each had 101 sets of data, bringing the total to 202 sets as you can see in the picture. N estimator = 9,000 and random state = 43. Since XG Boost only uses 9000 N estimators it delivers faster results than the other two algorithms. After the N estimator is set to 9,000 the accuracy does not change. It takes Random Forest 81,000 N estimators to give 90.17% accuracy, but XG Boost does it with just 9,000.

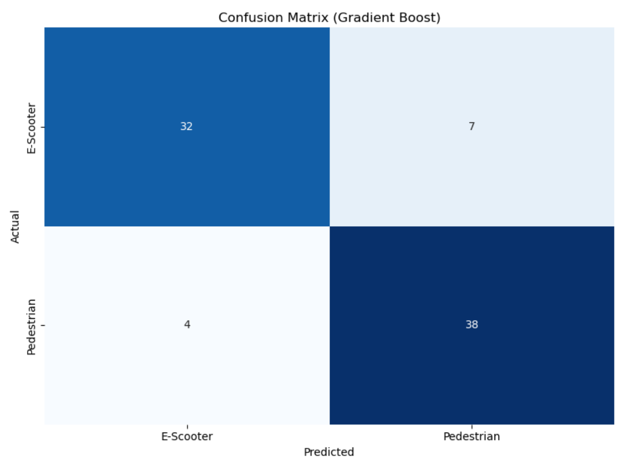


Figure 4.23 Confusion Matrix of Gradient Boost

Figure 4.23 presents the confusion matrix of the Gradient Boost algorithm. Out of the 81 sets of data four sets were misclassified as e-scooters and seven as pedestrian.

4.7 Performance Comparison

The Table shows the accuracy and the n estimators of all the algorithms used. XG Boost and Random Forest had similar accuracy. But since XG Boost only required 9,000 n estimators, it performed the fastest among the three.

Algorithm	Accuracy	N Estimators
Random Forest	90.12%	81,000
XG Boost	90.12%	9,000
Gradient Boost	86.42%	81,000

Table 4.1 Results

Chapter 5 Conclusions and Future Work

5.1 Conclusions

As the public perception of e-scooters increases, more commuters are willing to adapt. Unfortunately, since the existing roads are not designed to accommodate such users, we are observing increased accidents and casualties. This thesis has proposed a novel approach to identify e-scooters from point cloud data recorded with roadside LiDAR sensors. City planners can use this algorithm to understand the frequency of e-scooter users in different time frames. This data will then allow for the design of specialized road strips in important regions for safer commuting of e-scooters. Out of the three algorithms tested XG Boost had the highest precision of 90.17% with the least number of n estimators.

5.2 Future Work

- The proposed algorithm can be made more efficient by using sensor fusion methods to better cluster points to avoid unwanted elasticity readings.
- The algorithm can also be paired with point cloud perception algorithms like PointPillars to find the region of interest and then proceed to track and classify if the actor is a pedestrian or an e-scooter user.

References

[1] World Health Organization. 2023. Global status report on road safety 2023. World Health Organization, Geneva, Switzerland. ix, 81 p. pages.

[2] Simiao Chen, Michael Kuhn, Klaus Prettner, and David E Bloom. 2019. The global macroeconomic burden of road injuries: estimates and projections for 166 countries. The Lancet Planetary Health 3, 9 (2019), e390–e398. https://doi.org/10.1016/S2542-5196(19)30170-6

[3] Renato M. Silva, Gregório F. Azevedo, Matheus V. V. Berto, Jean R. Rocha, Eduardo C. Fidelis, Matheus V. Nogueira, Pedro H. Lisboa, Tiago A. Almeida. 2024. Vulnerable Road Users Detection and Safety Enhancement: A Comprehensive Survey.

[4] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, “Pv-rcnn: Point-voxel feature set abstraction for 3d object detection,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 10 529–10 538.

[5] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, “Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection,” International Journal of Computer Vision, vol. 131.

[6] Maosheng Ye, Shuangjie Xu, Tongyi Cao, “HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection”; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 1631-1640.

[7] Jordan S. K. Hu, Tianshu Kuai, Steven L. Waslander “Point Density-Aware Voxels for LiDAR 3D Object Detection”; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 8469-8478.

[8] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1903–1911.

[9] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "Bevdet: Highperformance multi-camera 3d object detection in bird-eye-view," arXiv preprint arXiv:2112.11790, 2021.

[10] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang & Oscar Beijbom ; PointPillars: Fast Encoders for Object Detection from Point Clouds.

[11] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In CVPR, 2017.

[12] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. In IROS, 2018.

[13] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In CVPR, 2018.

[14] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. CoRR abs/1612.00593 (2016).

[15] Li, Y., Bu, R., Sun, M., Chen, B.: Pointcnn (2018).

[16] Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds (2018).

[17] A Geiger, P Lenz & R Urtasun ; Vision meets robotics: The KITTI dataset, 2013.

[18] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, Dragomir Anguelov; Scalability in Perception for Autonomous Driving: Waymo Open Dataset.

[19] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, Oscar Beijbom; nuScenes: A Multimodal Dataset for Autonomous Driving. Computer Vision and Pattern Recognition (CVPR), 2020.

[20] W. Maddern, G. Pascoe, C. Linegar and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset", The International Journal of Robotics Research (IJRR), 2016.

[21] Michael Meyer and Georg Kuschk; “Automotive RADAR dataset for deep learning based 3D object detection”. 2019 16th European radar conference.

[22] Bijelic, Mario and Gruber, Tobias and Mannan, Fahim and Kraus, Florian and Ritter, Werner and Dietmayer, Klaus and Heide, Felix; “Seeing Through Fog Without Seeing Fog:Deep Multimodal Sensor Fusion in Unseen Adverse Weather”. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2022.

[23] Kiat-Choong Chen, Ian Hsieh, Cao An Wang; “A Genetic Algorithm for Minimum Tetrahedralization of a Convex Polyhedron”. Halifax, Nova Scotia, August 11–13, 2003.

[24] https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

[25] https://www.datacamp.com/tutorial/xgboost-in-python

[26] Vicente Milanés, Jorge Villagrá, Jorge Godoy, Javier Simó, Joshué Pérez, and Enrique Onieva; An Intelligent V2I-Based Traffic Management System. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 13, NO. 1, MARCH 2012.

[27] https://www.ghsa.org/resources/news-releases/TRB-Escooter-Safety Report22

[28] https://appinventiv.com/blog/escooter-trends-and-statistics/

[29] https://www.datacamp.com/tutorial/random-forests-classifier-python

[30] https://www.datacamp.com/tutorial/guide-to-the-gradient-boosting-algorithm

[31] Guldenring, J., Wietfeld, C.: Scalability analysis of context-aware multi-RAT Car-to-Cloud communication. In: IEEE 92nd Vehicular Technology Conference (VTC2) on Proceedings, Victoria, Canada, pp. 1–6. IEEE (2020).

[32] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, James Hays; "Argoverse: 3D Tracking and Forecasting with Rich Maps". Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[33] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan and Peter Carr and James Hays. “Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting”. NeurIPS Datasets and Benchmarks 2021.

[34] Giseop Kim; Yeong Sang Park; Younghun Cho; Jinyong Jeong; Ayoung Kim. “MulRan: Multimodal Range Dataset for Urban Place Recognition”. 2020 IEEE International Conference on Robotics and Automation.

[35] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, Andrew Wallace; “RADIATE: A Radar Dataset for Automotive Perception in Bad Weather”. 2021 IEEE International Conference on Robotics and Automation.

[36] Akhil Kurup and Jeremy Bos; “DSOR: A Scalable Statistical Filter for Removing Falling Snow from LiDAR Point Clouds in Severe Winter Weather”. Computer Vision and Pattern Recognition, 2021.

[37] Mao, Jiageng and Niu, Minzhe and Jiang, Chenhan and Liang, Xiaodan and Li, Yamin and Ye, Chaoqiang and Zhang, Wei and Li, Zhenguo and Yu, Jie and Xu, Chunjing and others; “One Million Scenes for Autonomous Driving: ONCE Dataset”. 2021, https://arxiv.org/abs/2106.11037.

[38] Jabłoński, Joanna Iwaniec, and Wojciech Zabierowski; “Comparison of Pedestrian Detectors for LiDAR Sensor Trained on Custom Synthetic, Real and Mixed Datasets”. https://doi.org/10.3390/s22187014.

[39] Runsheng Xu, Yi Guo, Xu Han, Xin Xia, Hao Xiang, and Jiaqi Ma. 2021. OpenCDA: An Open Cooperative Driving Automation Framework Integrated with Co-Simulation. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, IEEE, Indianapolis, IN, USA, 1155–1162.

[40] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO – Simulation of Urban MObility: An Overview. In Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind, Barcelona, Spain, 55–60.

[41] Alexey Dosovitskiy and German Ros and Felipe Codevilla and Antonio Lopez and Vladlen Koltun; "CARLA}: {An} Open Urban Driving Simulator". Proceedings of the 1st Annual Conference on Robot Learning, pg. 1 – 16, 2017.

[42] Mina Alibeigi, William Ljungbergh, Adam Tonderski, Georg Hess, Adam Lilja, Carl Lindstrom, Daria Motorniuk, Junsheng Fu, Jenny Widahl, and Christoffer Petersson. 2023. Zenseact Open Dataset: A large-scale and diverse multimodal dataset for autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision. IEEE, Montreal, Canada, 20121–20131.

[43] Novel Certad, Enrico del Re, Helena Korndörfer, Gregory Schröder, Walter Morales-Alvarez, Sebastian Tschernuth, Delgermaa Gankhuyag, Luigi del Re, and Cristina Olaverri-Monreal; "2024. Interaction of Autonomous and Manually Controlled Vehicles Multi scenario Vehicle Interaction Dataset". IEEE Intelligent Transportation Systems Magazine (2024).

[44] Runsheng Xu, Xin Xia, Jinlong Li, Hanzhao Li, Shuo Zhang, Zhengzhong Tu, Zonglin Meng, Hao Xiang, Xiaoyu Dong, Rui Song, Hongkai Yu, Bolei Zhou, Jiaqi Ma; "V2V4Real: A Real-world Large-scale Dataset for Vehicle-to-Vehicle Cooperative Perception". CVPR 2023.

[45] Jia Huang, Alvika Gautam, Junghun Choi, Srikanth Saripalli; "WiDEVIEW: An UltraWideBand and Vision Dataset for Deciphering Pedestrian-Vehicle Interactions". arXiv:2309.16057 [cs.RO].

[46] Andras Palffy, Ewoud Pool, Srimannarayana Baratam, Julian F. P. Kooij, Dariu M. Gavrila; "Multi-Class Road User Detection With 3+1D Radar in the View-of-Delft Dataset". IEEE ROBOTICS AND AUTOMATION LETTERS 2022.

[47] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, Huijing Zhao; "SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances". IEEE Intelligent Vehicles Symposium (2020).

[48] Huanan Wang, Xinyu Zhang, Jun Li, Zhiwei Li, Lei Yang, Shuyue Pan, Yongqiang Deng. "IPS300+: a Challenging Multimodal Dataset for Intersection Perception System". 2021 arXiv:2106.02781 [cs.CV].

[49] Deng Yongqiang, Wang Dengjiang, Cao Gang, Ma Bing, Guan Xijia, Wang Yajun, Liu Jianchao, Fang Yanming, Li Juanjuan; "BAAI-VANJEE Roadside Dataset: Towards the Connected Automated Vehicle Highway technologies in Challenging Environments of China". 2021 arXiv:2105.14370 [cs.CV].

[50] M. Hetzel, H. Reichert, G. Reitberger, K. Doll, E. Fuchs, and B. Sick "The IMPTC Dataset: An Infrastructural Multi-Person Trajectory and Context Dataset", IV 2023, Anchorage USA.

[51] Tao WU, Jun Hu, Lei Ye, Kai Ding; A Pedestrian Detection Algorithm Based on Score Fusion for Multi-LiDAR Systems. Sensor Fusion For Autonomous Vehicles, 2021.

[52] Jain, A.; Nandakumar, K.; Ross, A. Score normalization in multimodal biometric systems. Pattern Recognit. 2005, 38, 2270–2285.

[53] Kevin Peterson; Jason Ziglar; Paul E. Rybski; Fast feature detection and stochastic parameter estimation of road shape using multiple LIDAR, IEE 2008.

[54] Arturo L. Rankin, Andres Huertas, Larry H. Matthies; Night-time negative obstacle detection for off-road autonomous navigation, 2007.

[55] Jacoby Larson, Mohan Trivedi; Lidar based off-road negative obstacle detection and analysis, 2011.

[56] Erke Shang, Xiangjing An, Tao Wu, Qiping Yuan, Hangen He; LiDAR Based Negative Obstacle Detection for Field Autonomous Land Vehicles, 2005.

[57] Christoph Mertz, Luis E. Navarro-Serment, Robert MacLachlan, Paul Rybski, Aaron Steinfeld, Arne Suppe, Christopher Urmson, Nicolas Vandale, Martial Herbert, Chuck Thorpe, David Duggins, Jay Gowdy; Moving object detection with laser scanners, 2012.

[58] Tao WU, Jun Hu, Lei Ye, Kai Ding; A Pedestrian Detection Algorithm Based on Score Fusion for Multi-LiDAR Systems. Sensor Fusion For Autonomous Vehicles, 2021.

[59] Arun A. Ross, Anil K. Jain, Karthik Nandakumar. Handbook of Multibiometrics, 2006.

[60] https://requestum.com/blog/ai-in-transportation-and-logistics

[62] R. Abduljabbar, H. Dia, S. Liyanage, S.A. Bagloee; Applications of Artificial intelligence in transport: an overview, Sustainability 11 (189) (2019).

[63] Juncong Fei, Wenbo Chen, Philipp Heidenreich, Sascha Wirges, and Christoph Stiller; SemanticVoxels: Sequential Fusion for 3D Pedestrian Detection using LiDAR Point Cloud and Semantic Segmentation.

[64] Maite Szarvas, Utsushi Sakait and Jun Ogata; Real-time Pedestrian Detection Using LIDAR and Convolutional Neural Networks, 2006.

[65] W. Wang, D. Tran, and M. Feiszli, "What Makes Training Multi-Modal Networks Hard?" arXiv preprint arXiv:1905.12681, 2019.

[66] Sourabh Vora, Alex H. Lang, Bassam Helou & Oscar Beijbom ; PointPainting: Sequential Fusion for 3D Object Detection. 2020 CVPR Access.

[67] www.github.com/eduardohenriquearnold/CODD

[68] www.zod.zenseact.com/frames/

[69] www.ouster.com

[70] Ji Dong Choi and Min Young Kim; "A sensor fusion system with thermal infrared camera and LiDAR for autonomous vehicles and deep learning based object detection". ICT Express Volume 9, Issue 2, April 2023.

[71] Wael Farag; "Real-time lidar and radar fusion for road-objects detection and tracking". International Journal of Computational Science and Engineering Vol. 24, No. 5, 2021.

[72] Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, Zaiqing Nie; “DAIR-V2X: A Large-Scale Dataset for Vehicle-Infrastructure Cooperative 3D Object Detection”. arXiv:2204.05575 [cs.CV]. 2022.

[73] Hao Xiang, Zhaoliang Zheng, Xin Xia, Runsheng Xu, Letian Gao, Zewei Zhou, Xu Han, Xinkai Ji, Mingxi Li, Zonglin Meng, Li Jin, Mingyue Lei, Zhaoyang Ma, Zihang He, Haoxuan Ma, Yunshuang Yuan, Yingqian Zhao, Jiaqi Ma; “V2X-Real: a Largs-Scale Dataset for Vehicle-to-Everything Cooperative Perception”. arXiv:2403.16034 [cs.CV], 2024.

[74] Arnold, Eduardo and Mozaffari, Sajjad and Dianati, Mehrdad; “Fast and Robust Registration of Partially Overlapping Point Clouds”. IEEE Robotics and Automation Letters, 2021.

[75] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, Jiaqi Ma; “OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication”. In 2022 International Conference on Robotics and Automation (ICRA). IEEE, IEEE, Philadelphia, PA, USA, 2583–2589.

[76] Kumar Apurv, Renran Tian, and Rini Sherony; “Detection of E-scooter Riders in Naturalistic Scenes”. arXiv:2111.14060 [cs.CV]. 2021.

[77] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[78] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll ́ ar, C. L. Zitnick, Microsoft coco: common objects in context, in: European Conference on Computer Vision, Springer, 2014.

[79] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[80] H. Nguyen, M. Nguyen, Q. Sun, Electric scooter and its rider detection framework based on deep learning for supporting scooter-related injury emergency services, in: International Symposium on Geometry and Vision, Springer, 2021.

[81] Shane Gilroy a,b, , Darragh Mullins b , Edward Jones b , Ashkan Parsi b , Martin Glavin b; "E-Scooter Rider detection and classification in dense urban environments". Results in Engineering Volume 16, December 2022.

[82] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[83] X. Zhou, D. Wang, P. Krahenbühl, "Objects as Points", arXiv preprint arXiv: 1904.07850.

[84] Dong Chen, Arman Hosseini, Arik Smith, Amir Farzin Nikkhah, Arsalan Heydarian, Omid Shoghli, Bradford Campbell; "Performance Evaluation of Real-Time Object Detection for Electric Scooters", arXiv:2405.03039 [cs.CV], 2024.

[85] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430, 2021.

[86] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flowguided feature aggregation for video object detection. In Proceedings of the IEEE international conference on computer vision, pages 408–417, 2017.

[87] Khalil Sabri, Célia Djilali, Guillaume-Alexandre Bilodeau, Nicolas Saunier, Wassim Bouachir; "Detection of Micromobility Vehicles in Urban Traffic Videos". arXiv:2402.18503 [cs.CV], 2024.

[88] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, Francisco Herrera; "Deep Learning in Video Multi-Object Tracking: A Survey". arXiv:1907.12740 [cs.CV].

[89] Bo Wu & Ram Nevatia; "Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors". International Journal of Computer Vision, 2007.

[90] Jiaying Wu, J. A. N. van Aardt, Joseph McGlinchy, and Gregory P. Asner. 2012. A Robust Signal Preprocessing Chain for Small-Footprint Waveform LiDAR. IEEE Transactions on Geoscience and Remote Sensing 50, 8 (2012).

[91] H. Li, J. Chang, F. Xu, Z. Liu, Z. Yang, L. Zhang, S. Zhang, R. Mao, X. Dou, and B. Liu. 2019. Efficient Lidar Signal Denoising Algorithm Using Variational Mode Decomposition Combined with a Whale Optimization Algorithm. Remote Sensing 11, 2 (2019).

[92] Jiaying Wu, J. A. N. van Aardt, Joseph McGlinchy, and Gregory P. Asner. 2012. A Robust Signal Preprocessing Chain for Small-Footprint Waveform LiDAR. IEEE Transactions on Geoscience and Remote Sensing 50, 8 (2012).

[93] Yao Duan, Chuanchuan Yang, Hao Chen, Weizhen Yan, and Hongbin Li. 2021. Low-complexity point cloud denoising for LiDAR by PCA-based dimension reduction. Optics Communications 482 (2021).

[94] X. Xie, L. Bai, and X. Huang. 2022. Real-Time LiDAR Point Cloud Semantic Segmentation for Autonomous Driving. Electronics 11, 1 (2022), doi.org/10.3390/electronics11010011.

[95] G. Zhou, X. Zhou, J. Chen, G. Jia, and Q. Zhu. 2022. LiDAR Echo Gaussian Decomposition Algorithm for FPGA Implementation. Sensors 22, 12 (2022).

[96] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu; “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). 1996.

[97] Martin A. Fischler, Robert C. Bolles; “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. Communications of the ACMVol. 24, No. 6, 1981.

[98] Lin Li, Yang,Haihong Zhu, Dalin Li, You Li and Lei Tang; “An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells”. 2017, doi.org/10.3390/rs9050433.

[99] Hui Chen, Man Liang, Wanquan Liu, Weina Wang, Peter Xiaoping Liu; “An approach to boundary detection for 3D point clouds based on DBSCAN clustering”. Pattern Recognition Volume 124, April 2022.

[100] Anh Nguyen; Bac Le; “3D point cloud segmentation: A survey”. IEEE Conference on Robotics, Automation and Mechatronics. 2013.

[101] P.J. Besl, R.C. Jain, Segmentation through variable order surface fitting, IEEE Transaction on Pattern Analysis and Machine Intelligence 10, 1988.

[102] J. Chen, B. Chen, Architectural modeling from sparsely scanned range data. Int. J. Comput. Vision 78, 2008.

[103] B. Bhanu, S. Lee, C. Ho, and T. Henderson, Range data processing: Representation of surfaces by edges. In proc.int. Pattern recognition conf, 1896.

[104] R. Schnabel, R. Wahl, R.Klein, Efficient ransac for point cloud shape detection. Comput. Graph. Forum 26(2), 214-226, 2007.

[105] N. Gelfand, L. Guibas, Shape segmentation using local slippage analysis. In Proceedings of Eurographics, pp. 214-223. ACM, New York, 2004.

[106] Ramy Ashraf Zeineldin and Nawal El-Fishawy; "A Survey of RANSAC enhancements for Plane Detection in 3D Point Clouds", Menoufia Journal of Electronic Engineering Research, July 2017.

[107] A. Golovinskiy, T. Funkhouser, Min-cut based segmentation of point clouds, IEEE Workshop on Search in 3D and Video at ICCV, 2009.

[108] ipg-automotive.com/en/products-solutions/software/carmaker/

[109] Leonardo Novicki Neto, Fabio Reway, Yuri Poledna, Maikol Funk Drechsler, Eduardo Parente Ribeiro, Werner Huber and Christian Icking; "TWICE Dataset: Digital Twin of Test Scenarios in a Controlled Environment". IEEE Sensors Journal, 2023.

[110] Fredrik Warg, Sebastien Liandrat, Valentina Donzella, Graham Lee, Pak Hung Chan, Reija Viinanen, Antti Kangasräasiö, Umut Cihan, Heikki Hyyti, Tobias Waldheuer, et al. 2023. ROADVIEW Robust Automated Driving in Extreme Weather:

Deliverable D2. 1: Definition of the complex environment conditions. WP2–Physical system setup, use cases, requirements and standards.

[111] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, Jia Heming; “K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data”. Information Sciences Volume 622, April 2023.

[112] Sakshi Patel; Shivani Sihmar; Aman Jatain; “A study of hierarchical clustering algorithms”. International Conference on Computing for Sustainable Global Development (INDIACom), 2015.

[113] Ulrike von Luxburg; “A Tutorial on Spectral Clustering”, arXiv:0711.0189 [cs.DS], 2007.

[114] James C. Bezdek, Robert Ehrlich, William Full; “FCM: The fuzzy c-means clustering algorithm”. Computers & Geosciences, Volume 10, Issues 2–3, 1984.

[115] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander; “OPTICS: ordering points to identify the clustering structure”. Proceedings of the 1999 ACM SIGMOD international conference on Management of data.

[116] D. Reynolds; “Gaussian Mixture Models”. Encyclopedia of Biometrics 18 October 2018.

[117] Siddharth Madan & Kristin J. Dana; “Modified balanced iterative reducing and clustering using hierarchies (m-BIRCH) for visual clustering”. Pattern Analysis and Applications Volume 19, pages 1023–1040, (2016).

[118] cdn.neuvition.com/technology-blog/clustering-algorithms.html

[119] Jiaming Zhang, Ruiping Liu, Hao Shi, Kailun Yang, Simon Reiß, Kunyu Peng, Haodong Fu, Kaiwei Wang, Rainer Stiefelhagen; “Delivering Arbitrary-Modal Semantic Segmentation”, arXiv:2303.01480 [cs.CV], 2023.

[120] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, Shijian Lu; “Transfer Learning from Synthetic to Real LiDAR Point Cloud for Semantic Segmentation”, arXiv:2107.05399 [cs.CV], 2021.

[121] Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, Chen Feng; “V2X-Sim: Multi-Agent Collaborative Perception Dataset and Benchmark for Autonomous Driving”, arXiv:2202.08449 [cs.CV], 2022.

[122] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, Jiaqi Ma; “V2X-ViT: Vehicle-to-Everything Cooperative Perception with Vision Transformer”, arXiv:2203.10638 [cs.CV], 2022.

[123] Ruiqing Mao, Jingyu Guo, Yukuan Jia, Yuxuan Sun, Sheng Zhou, Zhisheng Niu; “DOLPHINS: Dataset for Collaborative Perception enabled Harmonious and Interconnected Self-driving”, arXiv:2207.07609 [cs.CV], 2022.

[124] https://law.justia.com/codes/california/2009/veh/21220-21235.html

[125] https://codes.findlaw.com/ca/vehicle-code/veh-sect-21235

[126] Xiaoqing Ye, Mao Shu, Hanyu Li, Yifeng Shi, Yingying Li, Guangjie Wang, Xiao Tan, Errui Ding; “Rope3D: TheRoadside Perception Dataset for Autonomous Driving and Monocular 3D Object Detection Task”, arXiv:2203.13608 [cs.CV], 2022.

[127] Ruiyang Hao, Siqi Fan, Yingru Dai, Zhenlin Zhang, Chenxi Li, Yuntian Wang, Haibao Yu, Wenxian Yang, Jirui Yuan, Zaiqing Nie; “RCooper: A Real-world Large-scale Dataset for Roadside Cooperative Perception”, arXiv:2403.10145 [cs.CV], 2024.

[128] Tianqi Wang, Sukmin Kim, Ji Wenxuan, Enze Xie, Chongjian Ge, Junsong Chen, Zhenguo Li, Ping Luo; “DeepAccident: A Motion and Accident Prediction Benchmark for V2X Autonomous Driving”, arXiv:2304.01168v5 [cs.CV], 2023.

[129] Walter Zimmer, Christian Creß, Huu Tung Nguyen & Alois C. Knoll; “TUMTraf Intersection Dataset: All You Need for Urban 3D Camera-LiDAR Roadside Perception”, 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC).

[130] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, Ruigang Yang; “IoU Loss for 2D/3D Object Detection”, arXiv:1908.03851 [cs.CV], 2019.

[131] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).

[132] Kunhare, N., Tiwari, R. & Dhar, J. Particle swarm optimization and feature selection for intrusion detection system. Sādhanā 45, 109 (2020).

[133] Guo, Rui; Zhao, Zhiqian; Wang, Tao; Liu, Guangheng; Zhao, Jingyi; Gao, Dianrong; "Degradation State Recognition of Piston Pump Based on ICEEMDAN and XGBoost", Applied Sciences (2076-3417), 2020, Vol 10, Issue 18, p6593.

[134] Netzer Moriya; "Form Convex Hull to Concavity: Surface Contraction Around a Point Set", arXiv:2401.14189v2, 06 Mar 2024.

[135] Yu Zhang et al 2024 J. Phys.: Condens. Matter 36 015101.

[136]https://www.kittelson.com/ideas/the-importance-of-the-equity-lens-in-transportation-planning-and-design

[137] Royo, S.; Ballesta-Garcia, M. An Overview of Lidar Imaging Systems for Autonomous Vehicles. Appl. Sci. 2019, Vol. 9, Page 4093 2019, 9, 4093, doi:10.3390/APP9194093.

[138]https://www.forbes.com/sites/enriquedans/2020/09/11/the-incredible-shrinking-lidar/

[139] Nitesh, K., Jana, P.K. Relay Node Placement with Assured Coverage and Connectivity: A Jarvis March Approach. Wireless Pers Commun 98, 1361–1381 (2018).

[140] https://www.cuemath.com/geometry/tetrahedron/

[141] https://people.computing.clemson.edu/~goddard/texts/algor/A1.pdf

[142] Hao Zou, Jinhao Cui, Xin Kong, Chujuan Zhang, Yong Liu, Feng Wen, and Wanlong Li. F-siamese tracker: A frustum-based double siamese network for 3d single object tracking. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, Oct 2020. ISBN 9781728162126. doi: 10.1109/

iros45743.2020.9341120.

[143] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completionfor 3d siamese tracking. In Proceedings of the IEEE/CVF Conference on Computer vision and Pattern Recognition (CVPR). IEEE, June 2019. ISBN 9781728132938.

[144] Kunhare, N., Tiwari, R. & Dhar, J. Particle swarm optimization and feature selection for intrusion detection system. *Sādhanā* 45, 109 (2020).

[145] National Safety Council. “Position/Policy Statement, Vulnerable Road Users.” https://nsc.org/getattachment/d5babee6-582d-4e66-804f-8d06f9b021a4/t-vulnerable-road-users-147

[146] Koonce, B. (2021). ResNet 34. In: Convolutional Neural Networks with Swift for Tensorflow. Apress, Berkeley, CA.