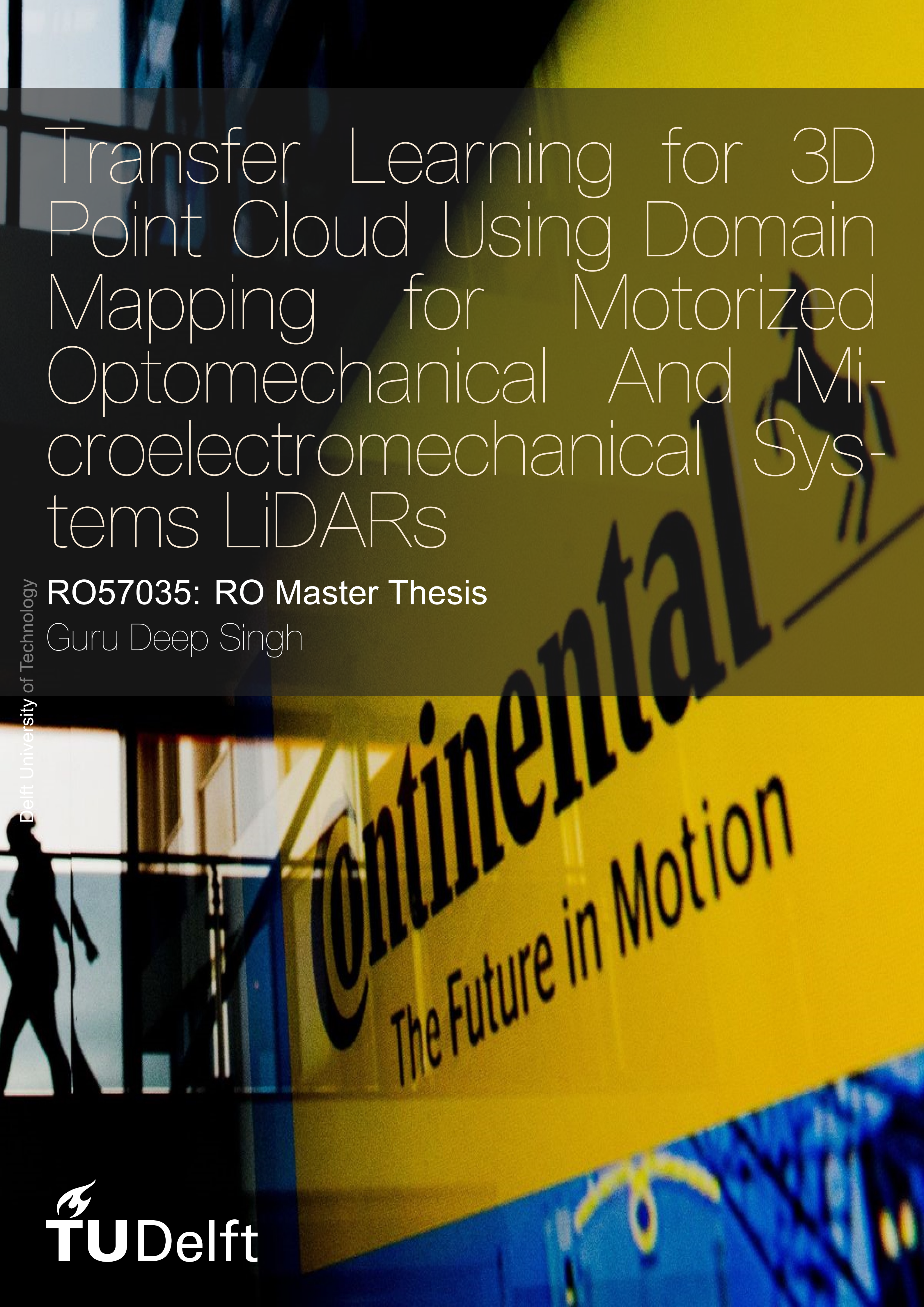


Transfer Learning for 3D Point Cloud Using Domain Mapping for Motorized Optomechanical And Microelectromechanical Systems LiDARs

RO57035: RO Master Thesis

Guru Deep Singh

Delft University of Technology

TU Delft

Transfer Learning for 3D Point Cloud Using Domain Mapping for Motorized Optomechanical And Microelectromechanical Systems LiDARs

by

Guru Deep Singh

to obtain the degree of Master of Science
at the Delft University of Technology

Student number:	5312558	
Project duration:	November 1, 2021 – July 13, 2022	
Faculty:	Faculty of Mechanical, Maritime and Materials Engineering	
Thesis committee:	Dr.Julian Kooij,	TU Delft
	Dr.Pascal Heiter,	Continental AG
	Dr. Holger Caesar,	TU Delft

Cover: Continental (Modified)

Preface

This master thesis is performed in collaboration with Delft University of Technology, The Netherlands, and Continental AG, Germany. This thesis aims to explore how to use the accumulated knowledge of semantic segmentation on Motorized Optomechanical LiDARs on up-and-coming Micro-electromechanical Systems LiDARs using transfer learning techniques.

I enjoyed my stay in Germany during the period on my thesis. I faced multiple challenges during my stay, partially due to the COVID outbreak, ranging from securing a visa and approval from University for the same. However, it was indeed a great learning experience.

I would like to extend my gratitude to my TU Delft supervisor Dr. Julian Kooij and my Continental supervisor Dr. Pascal Heiter without whom this thesis would not have been possible. I would also like to thank Dr. Bihao Wang who guided me with concept development during my initial phase. Moreover, I would like to thank Dr. Ernesto Denicia, and Dr. Sojo James from Continental for their constant guidance and Ms. Karin Van Tongeren, Co-coordinator of MSc Robotics who constantly supported me with my administrative necessities.

I would like to devote my work to my parents, brother, sister, and sister-in-law, who constantly supported me emotionally throughout my journey.

Guru Deep Singh
Delft, September 2022

Abstract

The automotive industry currently has been working on developing various levels of autonomy to assist in different Advanced Driver Assistance Systems (ADAS) with the ultimate aim of moving closer to the realization of an autonomous vehicle. For such ADAS, the industry has been using multiple sensors like Cameras, Radar, LiDAR, etc. LiDAR has been at the forefront of the research as it provides extremely rich and precise information about the surrounding. In research, Motorized Optomechanical LiDAR has been used for almost a decade now. However, it can not be deployed in mass-produced vehicles because of how expensive the sensor is, thus, limiting it to academia without practical use. Therefore, the industry has been working on a different category of LiDAR namely Microelectromechanical Systems (MEMS) LiDAR. Which uses solid-state technology instead of servo motors. Thus, they are compact and much cheaper in cost, making them feasible to deploy in mass-produced vehicles.

There has been a considerable amount of research already done on Motorized Optomechanical LiDAR, therefore, one must take advantage of it to learn about MEMS LiDAR. The domain of Transfer Learning provides an opportunity to learn from the accumulated knowledge of Motorized Optomechanical LiDAR (Source Domain) and use it to reduce the learning time on MEMS LiDAR (Target Domain). Thus, reducing the effort, cost, and time to research MEMS LiDAR technology.

This thesis explores the Domain Mapping strategy of transfer learning for point clouds. A scanning pattern-based domain mapping approach has been described in this thesis to reduce the domain gap between the source and target domain. Moreover, a point cloud densification pipeline that utilizes a depth completion network has been described to further reduce the gap between the two domains. Since semantic segmentation is one of the most actively researched tasks in the industry as it provides the researcher with in-depth information about the environment which could be used in activities like lane detection, parking assist, etc, therefore, this thesis utilizes it as the proxy for evaluating the performance of the domain mapping strategy. It compares a few domain mapping strategies like cropping the Field of View, an adaptation based on scanning pattern, and densification & adaptation to simulate the density of points in the source domain similar to the target domain. This thesis verifies that domain mapping strategy is a suitable solution to learn from Motorized Optomechanical LiDAR and use it to enhance performance on MEMS LiDAR and therefore reduce the data required for research of MEMS.

Contents

List of Figures

List of Tables

1 Introduction

Light Detection and Ranging (LiDAR) is one of the most favored sensors to be deployed in the environment for ADAS to provide exhaustive information [74]. It is based on the principle of Time of Flight (ToF) same as that of radar, however, the information provided by LiDAR point cloud is much more dense and precise than a radar, hence, is used for research and development of Advanced Driver Assistance Systems (ADAS). However, one biggest problem concerning LiDAR is that it is an expensive sensor with a high-end version going up to over 60,000 Euros. Thus, making it suitable for research but not practical to be used for mass-produced vehicles.

The LiDARs which have been used for decades are called Motorized Optomechanical LiDAR (Shown in Figure 1.1). Which consists of a servo rotating an inclined mirror that reflects the laser projected on it (Explained in detail in Section 2.1.2.1). To deploy LiDAR in commercial vehicles which are feasible for mass production, the industry has been involved in research, creating new types of LiDAR which eliminate the servo (most costly component) and rely on solid-state technology to reduce the cost and also the moving components to make LiDAR more robust to vibrations and other disturbances. The up-and-coming of these solid-state technologies is Microelectromechanical Systems (MEMS) LiDAR. Which is also referred to as quasi-solid state LiDAR (Shown in Figure 1.2). MEMS working principle has been described in detail in Section 2.1.2.2.

Figure 1.1: UltraPuck: Motorized Optomechanical LiDAR by Velodyne (Source: Velodyne Official Website)

Since MEMS is relatively new and not much data is available from it yet, therefore it makes it difficult to use with supervised machine learning algorithms. However, a lot of data has already been collected using Motorized Optomechanical LiDAR and is publicly available. Therefore, it only makes sense to try to learn the relevant knowledge from the existing LiDAR and boost the research of MEMS. Every LiDAR has a specific pattern in which it samples the environment also called a scanning pattern, which along with its geometric positioning, etc, determines the data distribution of the samples it collects. Therefore, learning from a category of LiDAR and testing on another would be a challenging task as we need to take into account the difference in data distribution between the two. We should keep in mind that machine learning algorithms have an innate assumption associated with them that, the data distribution of the training and test dataset are the same.

Figure 1.2: Hydra: Micro-electromechanical System LiDAR by Luminar (Source: Luminar Official Website)

Researchers have come up with multiple strategies to learn from a different domain of data and use the gained knowledge to infer on a different domain. This strategy is referred to as Transfer Learning (TL) [86]. The most commonly observed difference between two data distributions is Frequency Feature Bias and Context Bias [86]. The reason for the two is the difference in marginal probability distribution and conditional probability distribution between the two domains respectively. Thus, making the domain gap reduction of two LiDARs a non-trivial task. This brings up the first research question “How can the domain difference between the two categories of LiDAR be reduced?”.

Transfer learning uses different solution strategies with the aim of adapting the source to the target which generally takes place in a latent feature space. This projection of input space to latent space generally causes loss of information, therefore, it is important to know if enough information is preserved after adaptation. This raises the next research question “Does the density of information after adaptation stay the same in the source dataset?”.

Since the scanning pattern determines the sampling done by the LiDAR, it acts as the prime contributor in deciding the data distribution of a LiDAR. Therefore, it is an important aspect to study the reduction of domain difference between the LiDARs. However, the scanning pattern is generally not made public by the manufacturers. Therefore, we need to extract it ourselves. This poses the next research question “How can the scanning pattern of a LiDAR be generated?”.

ADAS applications such as lane assist, parking assist, etc require comprehensive information about the surrounding. Semantic segmentation has been used as one of the most prevalent tasks in the industry as it provides very precise, detailed, and dense information about ambiance by providing point-wise labels and therefore, is the downstream task of this thesis. The performance of semantic segmentation on the target dataset would act as the proxy to the performance of the domain mapping technique used. However, adapting the source dataset to the target dataset may cause semantic information of the source to be lost, but is required for training the supervised segmentation network. Thus, it poses the question “Is the semantic information lost after adaptation?”, “If the semantic information after adaptation is lost, how can it be generated without manual labeling?”.

Semantic segmentation for point clouds have been researched in-depth and there are many established architectures which have proven to be very effective like PointNet [53], PointNet++ [54], SONet [41], PointSIFT [35], RandLA-Net [32], etc. Semantic segmentation networks can be sub-categorized based on how the input is fed to the network, namely, Point-based, Image-based, and Voxel-based. The details of these categories have been mentioned in Section 2.4. Voxel-based categories are more suitable for LiDAR-based transfer learning problems because they help reduce the input size and make it definite. Voxelization helps reduce the computational power required for segmentation and therefore is suitable for ADAS. Moreover, they are suitable for the LiDAR-based transfer learning problems as they reduce the complexity of different input features, details of which have been provided in Section 2.4.3. Hence, for our problem of transfer learning for LiDAR, we deploy a voxel-based network.

In this thesis we perform multiple processing for mapping source to target dataset, ranging from cropping, adapting, densifying & adapting the source dataset. Therefore, to compare the performance of such domain mapping strategies, we train multiple models and compare the best-performing strategy. Finally, providing a suitable domain mapping strategy for our problem.

1.1. Thesis Organization

This thesis has been organized into five chapters, namely, “Introduction”, “Related Work”, “Methodology”, “Experiments” and “Conclusion”.

- **“Chapter 2”** introduces the reader to the associated background knowledge and current work on the topic and the surrounding concepts to allow the reader to better grasp the aspects of the thesis.
- **“Chapter 3”** introduces the reader to the theoretical understanding of the concepts which would be used for experimentation such as adaptation steps, depth completion network, and semantic segmentation network.
- **“Chapter 4”** introduces the readers to multiple experiments conducted during this thesis and their results and analysis.
- **“Chapter 5”** provides readers with the final outcome of the thesis while answering the research questions raised in “Introduction” and also provides points for further research.

2 Related Work

This thesis revolves around transfer learning for point clouds with the downstream task of semantic segmentation, therefore, in this chapter, we introduce the existing fundamentals required to proceed with this thesis. We go into detail about LiDARs by introducing the technical specifications of LiDARs. We also introduce the readers to the categories of LiDARs available in the industry.

Further, we discuss some pertinent datasets used in point cloud research and provide their brief understanding. Since the major focus of this thesis is transfer learning, therefore, we provide a detailed introduction to transfer learning and its categorization. We will also introduce readers to the solution strategies currently developed for transfer learning problems

Finally, we discuss semantic segmentation for point clouds and its categories pertaining to point clouds.

2.1. Light Detection and Ranging (LiDAR)

Light detection and Ranging (LiDAR) is a Time of Flight (ToF) active remote sensing technique which uses electromagnetic (EM) waves to detect a targeted object and determine its distance from the sensor [46]. LiDAR generates high-resolution information about the ambiance and has its uses in multiple fields ranging from archaeology, agriculture, astronomy, geology, robotics, military, and also in automotive. In automotive, the sensor has been deployed in research of Advanced Driver Assistance Systems (ADAS) [58].

2.1.1. Technical Specifications

LiDAR is a complex sensor constituting multiple aspects that affect its performance. These performance governing aspects are generally dependent on some geometric or architectural limitations. However, remains the same for different categories of LiDAR, therefore is important to know their significance for choosing a specific LiDAR for an application.

Axial Resolution
The axial resolution of a LiDAR is a measure of its ability to accurately distinguish between two axially separate target points [58]. It depends on the bandwidth of information that can be carried by the EM waves used in a particular LiDAR. Mathematically, it is the standard deviation of multiple inputs from a target point at a fixed distance.

Scanning Field of View
Field of View (FoV) is the angular range till which a LiDAR can detect target points [58]. FoV is defined separately for horizontal referred to as azimuth and vertical referred to as elevation. Together they determine the complete region in which the LiDAR can detect target.

Transmitted Power
Transmitted power is defined as the energy in the EM waves used for scanning, the transmitted power depends on wavelength, beam diameter, duration of exposure, and pulse width [58]. More energy in EM waves allows them to travel farther and thus, provide a response from a target at a much larger distance from the sensor. However, it is bottlenecked by the safety guidelines related to lasers to prevent harm to humans, commonly referred to as Maximum Permissible Exposure (MPE) regulations.

Frame Rate
The frame rate of a LiDAR is defined as the number of times it can scan the environment within a period [58].

Maximum Operating Range
Maximum Operating Range is the maximum distance a LiDAR can accurately identify a target [58]. It depends on the transmitted power of EM waves and the receiver's sensitivity. However, there are constraints posed on the receiver and transmitted power of laser which limits the operating range of LiDAR.

Scanning Pattern
The scanning pattern of a LiDAR is the method in which it scans the environment [39]. It could be as simple as a uniform raster-based scheme or could be as complex as a mathematical function. Scanning pattern is one of the major factor influencing the point distribution within a frame of point cloud sampled by LiDAR. It depends on the architectural scheme and working principle of the LiDAR. Scanning pattern of a Velodyne VLP-16 can be seen in Figure. 2.1

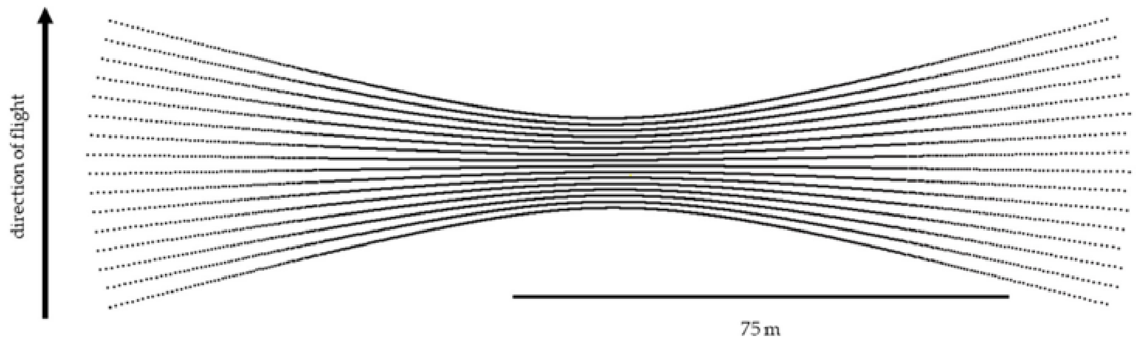


Figure 2.1: [39] Scan Pattern of single pass of velodyne VLP-16

2.1.2. LiDAR Categories

Many LiDARs have been conceptually created, however, only a few of them have been made into prototypes and even fewer have been certified and used for mass data collection. Some of the LiDARs are Motorized Optomechanical LiDAR, Flash LiDAR, Microelectromechanical Systems (MEMS) LiDAR, and Optical Phase Array (OPA) LiDAR. For every LiDAR the working principle of ToF remains the same, however, is very different architecturally. LiDAR's categorization have been shown in Figure 2.2

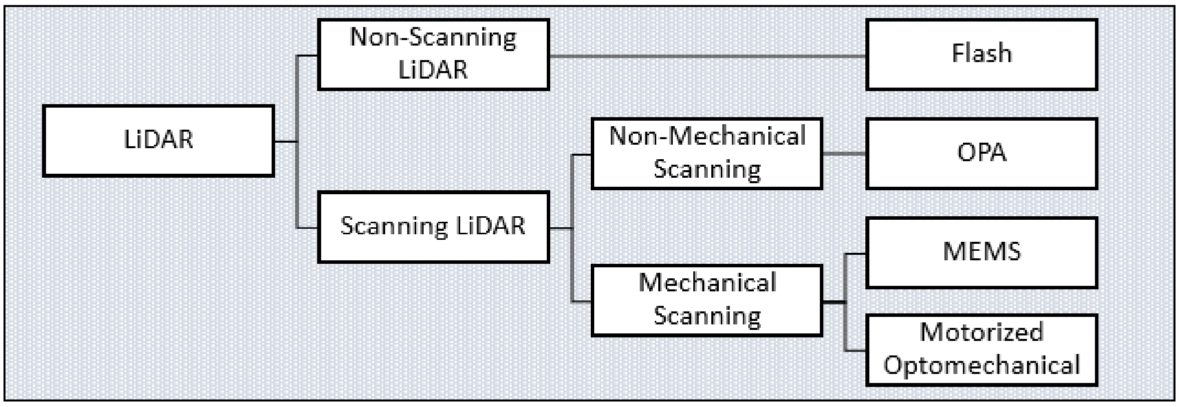


Figure 2.2: Types of LiDAR based on scanning mechanisms

Motorized Optomechanical LiDAR has been used for over a decade now and has proven to be a solution to provide dense information about the surroundings. However, due to their financial overhead, they have not been able to be utilized in mass-produced vehicles, therefore limiting their use in academia. MEMS are a breakthrough for the LiDAR application in automotive because of the usage of solid-state technology allowing to cut prices of LiDAR tremendously. The industry has also been researching Flash LiDAR, whereas OPA is still in the conceptual phase. However, we limit our research in this thesis to Motorized Optomechanical and MEMS LiDAR, which have been explained below.

2.1.2.1. Motorized Optomechanical LiDAR

Motorized Optomechanical LiDAR comes under the umbrella of scanning LiDAR which uses a servo-based rotating mechanism to scan the complete environment. Since the scanning is done while the sensor rotates along the axis of the servo therefore it records complete 360° FoV along azimuth as shown in Figure 2.3. Initially, a nodding mirror mechanism was used to allow the laser to be guided in an elevation direction and therefore scan the environment like a raster, however, it was soon replaced by multiple laser projectors and receiver units stacked vertically which can scan along different elevation angles simultaneously.

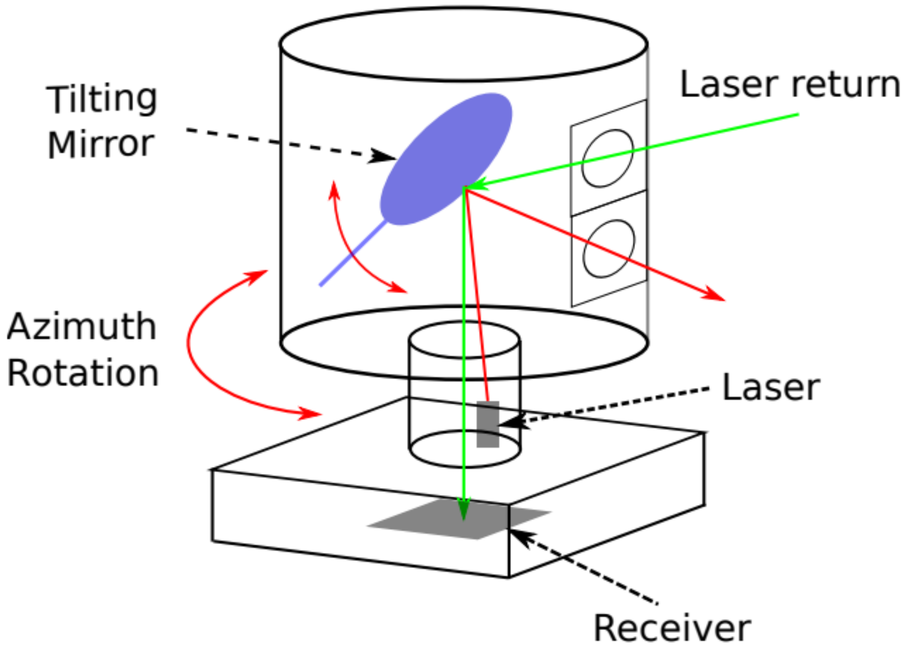


Figure 2.3: [43] Working principle of Motorized Optomechanical LiDAR

These LiDARs are bulky in packaging due to the rotating assembly and are quite expensive because of the mechanical components present. However, they focus power at a particular point of FoV at an instant of time, thus making it safe in a human environment. These sensors allow using high-powered lasers thus increasing the range of LiDAR.

2.1.2.2. Microelectromechanical Systems (MEMS) LiDAR

Microelectromechanical Systems LiDAR are also referred to as quasi-solid state LiDAR because they have a small plate mirror that moves while the remaining system is stationary. The diameter of the mirror ranges from 1mm to 7mm. MEMS uses a cascaded mirror configuration which is responsible for modulating, steering, controlling the phase, etc of the beam. Commonly an array of light transmitters and a cascaded 1D mirror arrangement are used to sweep multiple vertical beams along the horizontal FoV. The transmitted beams are reflected from the target and are collected using a 2D arrangement of Avalanche Photodiodes (APDs). MEMS uses the ToF principle to calculate the range of a target. Since they are devoid of big mechanical-based rotating mechanisms, they are compact and cheaper, however, the alignment of mirrors is a challenging work that requires immense precision. The working of the MEMS can be seen in Figure 2.4

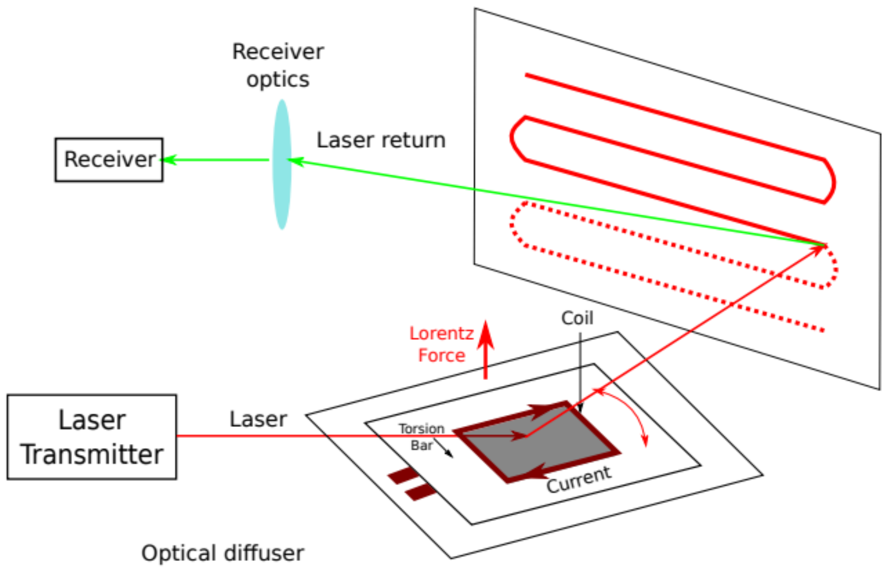


Figure 2.4: [80] Working principle of MEMS LiDAR

2.2. Datasets

Supervised deep learning algorithms need abundant ground-truth labeled data that could be used to train the parameters of a deep neural network [23]. Semantic understanding of point clouds is sought-after research in ADAS applications because it provides high-resolution attributes of the environment. There are plenty of such datasets which are freely made available by the scientific community to be used for research and development of new methodologies.

Point cloud datasets can be categorized on different aspects like Nature of Data, Sequentiality of Data, Type of LiDAR Used, etc. The nature of data category is one of the most prominent ways of choosing a dataset for an application and can be further sub-categorized into RGB-D datasets, static point cloud datasets, synthetic point cloud datasets, and sequential point cloud datasets. However, for our problem statement, we need to pay more attention to the type of LiDAR used to collect the data, mainly on Motorized Optomechanical and MEMS to select the candidate dataset for the source (Motorized Optomechanical) and the target (MEMS).

2.2.1. Motorized Optomechanical Datasets

This section will introduce some of the datasets used in industry which were collected using Motorized Optomechanical LiDAR.

2.2.1.1. SemanticKITTI

SemanticKITTI [4] is a large-scale point cloud dataset that is one of the signature datasets for semantic segmentation. It provides abundant data arranged in 22 sequences. The sequence range from 00 to 21, where the first 11 sequences i.e. 00 to 10 are provided with ground-truth labels to be used for training and validation while the latter 11 sequences i.e. 11 to 21 are used for testing and hence the labels for them are not made public. Researchers are required to submit the inference made on the test sequences online to evaluate the performance of their model. The training sequences contain a total of 23201 frames of point cloud while the testing sequences contain a total of 20351 frames. SemanticKITTI provides an average of 104k points per frame in both training and testing sequences.

SemanticKITTI provides point-wise labels for the Odometry sequences in KITTI [25] suit. Each point of SemanticKITTI is labeled from a set of 28 distinct classes. The 28 classes have a variety of categories ranging from vehicles, pedestrians, road, buildings, and objects. SemanticKITTI provides a different number of annotated points per class which can be seen in Figure 2.5. Moreover, the dataset also provides a voxel-based ground truth labeling per frame for the task of scene completion.

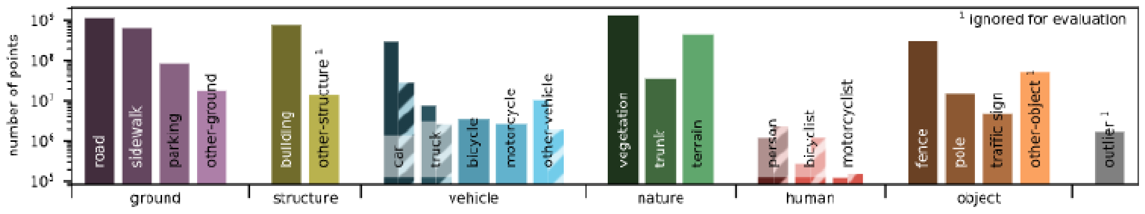


Figure 2.5: [4] Label distribution of points per class in SemanticKITTI dataset. Hatched bars under the vehicle and Human categories represent moving classes, while the solid bars show non-moving classes.

The dataset is collected using a 360^o rotating Motorized Optomechanical Velodyne LiDAR mounted on a vehicle. The sensor setup can be seen in Figure 2.6. The Horizontal and Vertical Field of View (FOV) for the sensor are 360^o and 26.9^o respectively.

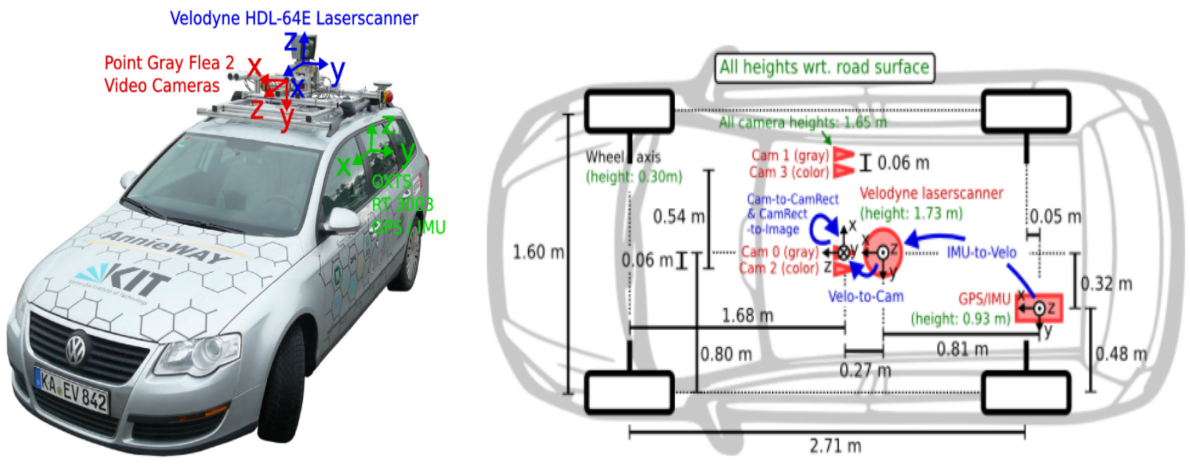

Velodyne HDL-64E Laserscanner
Point Gray Flea 2 Video Cameras
OXTS RT 3003 GPS / IMU
All heights wrt. road surface
All camera heights: 1.65 m
Wheel axis (height: 0.30m)
Cam 1 (gray)
Cam 3 (color)
0.06 m
Velodyne laserscanner (height: 1.73 m)
Cam-to-CamRect & CamRect -to-Image
Cam 0 (gray)
Cam 2 (color)
IMU-to-Velo
Velo-to-Cam
GPS/IMU (height: 0.93 m)
1.60 m
0.54 m
0.06 m
1.68 m
0.80 m
0.27 m
0.81 m
0.05 m
0.32 m
0.48 m
2.71 m

Figure 2.6: [4] Sensor setup for KITTI

2.2.1.2. nuScenes

NuScenes [9] is a large-scale point cloud dataset collected partially in the United States of America and partially in Singapore. It contains 1000 driving scenes each lasting 20 seconds. The driving scenes cover a variety of road scenarios. The dataset provides information on diverse weather conditions, traffic situations, and timings of the day. It is captured using a range of sensors like Camera, LiDAR, Radar, etc to provide multi-modal information.

NuScenes provide approximately 1.4 million camera images and 390k LiDAR point cloud frames. The dataset provides ground truth for the point cloud as point-wise labels as well as 3D bounding boxes. The bounding box ground-truth labels form a set of 23 distinct classes also containing information about the visibility attribute of each instance of the class. On the contrary, point-wise labels additionally contain 9 background classes and 23 foreground classes. However, it should be noted that only 40k point cloud frames are point-wise annotated.

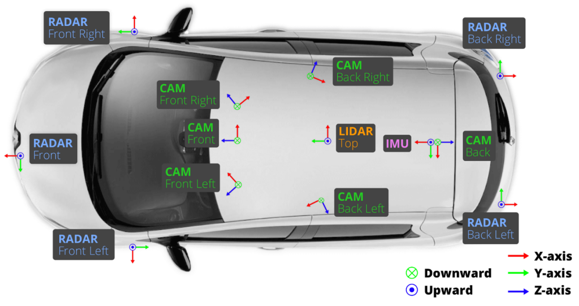


Figure 2.7: [9] Sensor setup for NuScenes

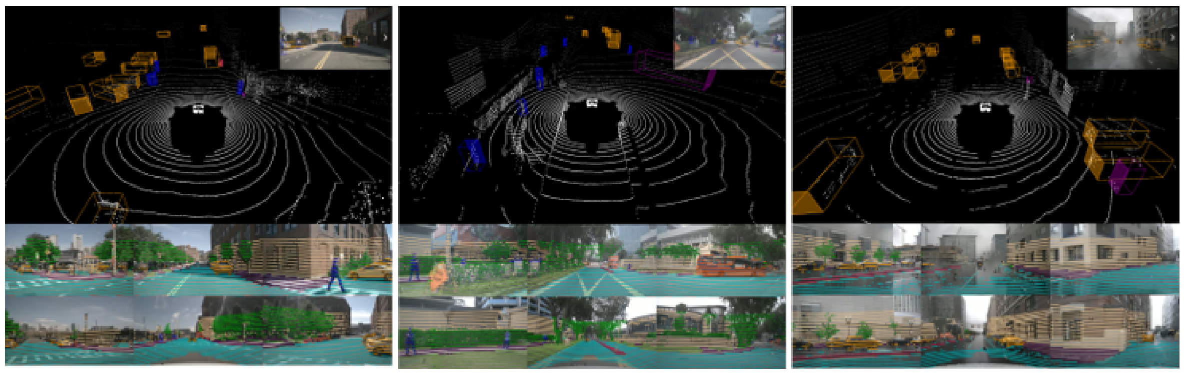


Figure 2.8: [9] An example of point cloud frame from NuScenes

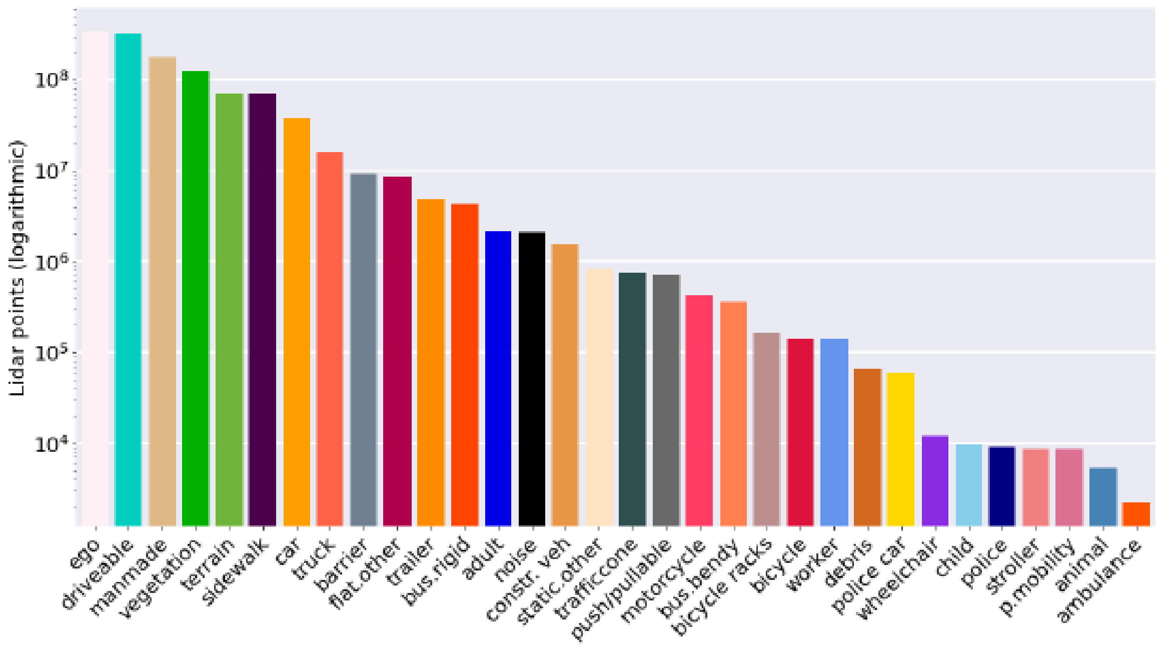


Figure 2.9: [9] Ratio of distribution of points in NuScenes

There is no information available about LiDAR used to collect nuScenes except that it is a 32-channel LiDAR and that the nature of point distribution shows that it is a Motorized Optomechanical LiDAR.

There are many more datasets collected using Motorized Optomechanical LiDAR like SemanticPOSS [51], A2D2 [26], Sydney Urban [18], etc the details for whom have been summarized in Table 2.1. Amongst the mentioned datasets SemanticKITTI is one of the most favored datasets in the domain of semantic segmentation in academia because it originates from the KITTI suit which provides multi-modal ground-truth and thus, allows the use of different data fusion strategies. Therefore, in this thesis, SemanticKITTI will be used as the candidate for Motorized Optomechanical LiDAR (Source Domain).

2.2.2. Microelectromechanical Systems Datasets

This section will introduce some of the datasets used in industry which were collected using MEMS LiDAR.

2.2.2.1. Cirrus

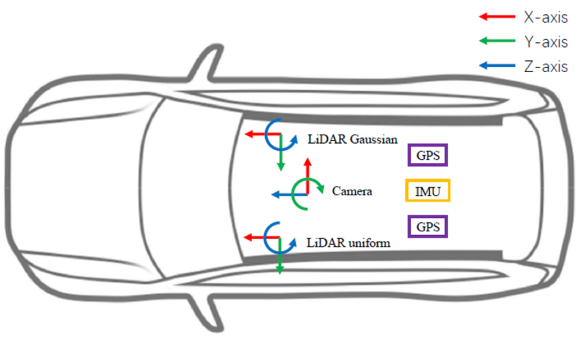


Figure 2.10: [75] Sensor setup for Cirrus

Cirrus [75] is a long-range LiDAR dataset published by Volvo Cars. It provides 7 sequences with the first 4 in the Highway scenario and the latter 3 in the urban scenario. It is collected using cameras and long-range LiDARs which have a 120° horizontal FoV and provides point up to a range of 250m. It is one of the only few long-range point cloud datasets available.

Cirrus is a bi-pattern dataset that provides every frame of point cloud with a uniform scanning pattern and a Gaussian scanning pattern. The Gaussian lies in the elevation direction and provides dense points in front of the vehicle, while a uniform scanning pattern provides a balanced distribution of points throughout the FoV of LiDAR. The Gaussian pattern is more useful in highway scenarios as the region of interest is directly in front of the vehicle and the surroundings can be ignored to an extent while a uniform pattern comes in handy to analyze urban scenarios where the complete FoV is the region of interest. The data collection setup consists of 2 Luminar 120° horizontal-FoV MEMS LIDAR sensors, an RGB camera, GPS, and IMU sensors. The placement of the sensors on the vehicle could be observed in the Figure 2.10.

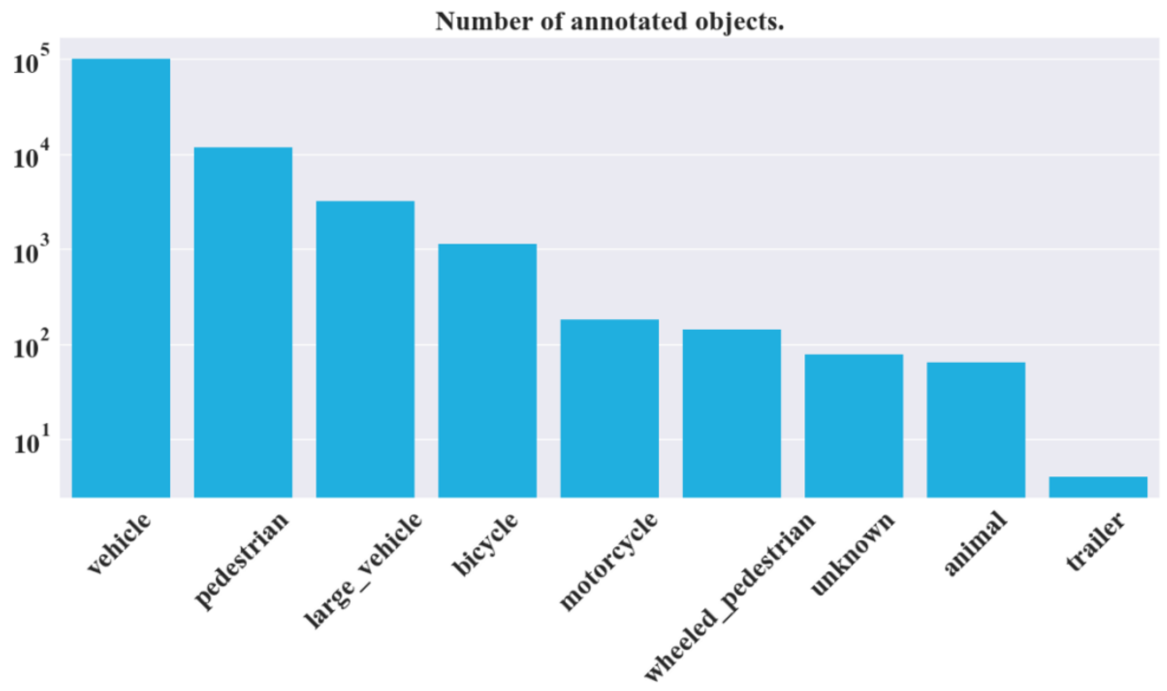


Figure 2.11: [75] Distribution of labels between the classes for Cirrus

Cirrus in 7 sequences provides 6,285 pairs of RGB image, LiDAR point cloud both in uniform and Gaussian configuration. It provides the ground truth of instances as bounding boxes that form a set of 8 distinct classes. The label distribution among these 8 classes can be seen in Figure 2.11. The sample frames of Cirrus can be seen in Figure 2.12

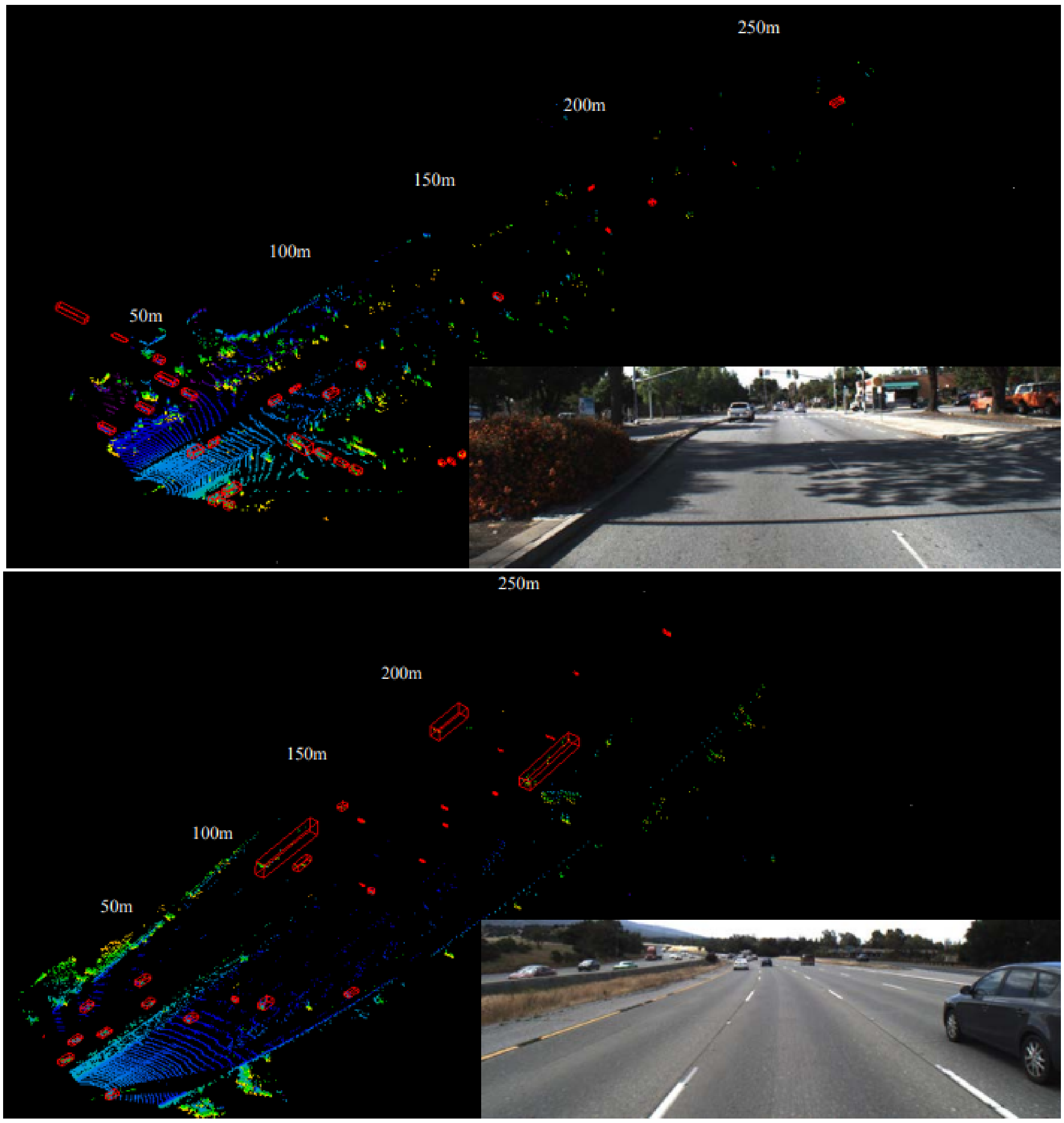


Figure 2.12: [75] An example LiDAR point clouds frame from Cirrus dataset with bounding boxes. Distance is marked in white.

Since MEMS is an up-and-coming technology there are not many publicly available datasets, Cirrus which was published in 2021 is one of a kind dataset available currently. Therefore, Cirrus is the selected candidate for the MEMS dataset (Target Domain), its features have been summarized in Table 2.1.

	[4] SemanticKITTI	[51] SemanticPOSS	[26] A2D2
Organization	Univ. of Bonn	Peking Univ.	Audi
Sensor Model	Velo. HDL-64E	Pandora x 5	Velo. HDL-64
Sensor Type	Optomechanical	Optomechanical	Optomechanical
Horizontal FoV	360°	360°	360°
Vertical FoV	26.9°	-16° to 7°	-15° to 15°
Sensor Range	120m	200m	100m
Power Consumed	60W	30W	8W x 5
Operating Voltage	12V - 32V	9V - 32V	9V - 18V
Laser Wavelength	903nm	905nm	903nm
LiDAR Channels	64	40	16
Horizontal Res.	0.08°	0.2°	0.1°-0.4°
Vertical Res.	0.4°	0.33°	2.0°
Frame Rate	10Hz	10Hz	10Hz
No. Of Scans	43552	2988	41277
No. Of Points	4549M	216M	1238M
No. Of Classes	28	14	38
RGB Images	YES	YES	YES
Annotations	point-wise	point-wise	point-wise
Sequential	Yes	Yes	Yes

	[9] nuScenes	[75] Cirrus	[18] Sydney Urban
Organization	nuTonomy	VolvoCars	ACRF
Sensor Model	Unknown	Luminar Hydra	2 x Velo. VLP-16
Sensor Type	Optomechanical	MEMS	Optomechanical
Horizontal FoV	360°	120°	360°
Vertical FoV	-30° to 10°	30°	26.9°
Sensor Range	70m	250m	120m
Power Consumed	–	55W	–
Operating Voltage	–	9V - 32V	–
Laser Wavelength	–	1550m	903nm
LiDAR Channels	32	N/A	64
Horizontal Res.	–	0.07°	0.08°
Vertical Res.	–	0.03°	0.4°
Frame Rate	13Hz	10Hz	10Hz
No. Of Scans	40K	6258	631
No. Of Points	2780M	–	–
No. Of Classes	23	8	14
RGB Images	YES	YES	NO
Annotations	point-wise	3D Bounding Boxes	point-wise
Sequential	Yes	Yes	No

Table 2.1: Summary of Datasets

2.3. Transfer Learning

Machine learning is a data-centric methodology that needs an enormous amount of data, however, in many domains where data is limited due to difficulty in collecting the data or limited by privacy laws like in the health sector, thus, limiting the research of ML in such domains. Transfer learning (TL) has been under research to boost the research in such domains and solve the problem of insufficient training data. Transfer Learning allows the ML algorithms to learn from a domain where an abundant amount of data is present, referred to as the source domain, and apply the learned knowledge to a new domain where data is limited, referred to as the target domain. The working principle of TL has been visualized in Figure 2.13

In traditional ML an innate assumption is made that the training and test data are independent and identically distributed (i.i.d), however, if the domain of training and test are varied then the performance of the ML algorithm degrades tremendously. Therefore, in our problem of training on Motorized Optomechanical LiDAR and testing on MEMS LiDAR, we need to employ the TL methodology to reduce the domain gap between the two.

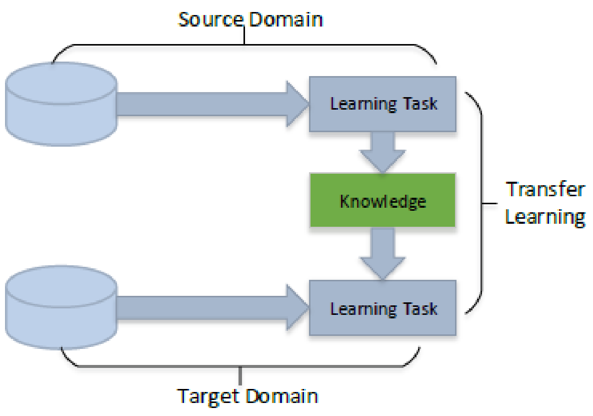


Figure 2.13: [64] Working principle of Transfer Learning

2.3.1. Terminology

In this section, we will describe some fundamental terminologies commonly referred to in TL literature, therefore imperative to know before discussing TL further.

Input Space: Set of all possible inputs. For example: For an image (1 channel) with a resolution of 800 x 600, each pixel's intensity can range from 0-255, thus the input space would be $(800 \times 600)^{256}$. However, only a very few of the instances in input space would make sensible data representing an actual physical entity.

Feature Space ($\mathcal{X}$): Space of features used to describe the dataset.

Domain ($\mathcal{D}$): A domain $\mathcal{D}$ is defined by two parts (Figure 2.14), a feature space $\mathcal{X}$ and a marginal probability distribution P(X), where X = x_1 , . . ., $x_n \in \mathcal{X}$.

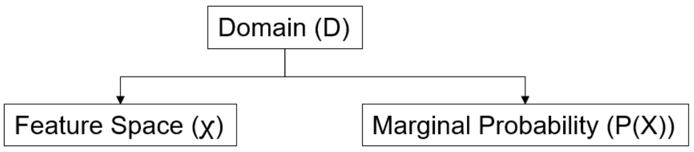


Figure 2.14: Constituents of a domain: X = $x_1, \ldots, x_n \in \mathcal{X}$; Where X is an instance and x_i are the features

Constituents of a task ($\mathcal{T}$): For a given domain $\mathcal{D}$, a task $\mathcal{T}$ is defined by two parts (Figure 2.15), a label space $\mathcal{Y}$, and a predictive function f(·), which is learned from the feature vector and label pairs x_i, y_i where $x_i \in$ X and $y_i \in \mathcal{Y}$.

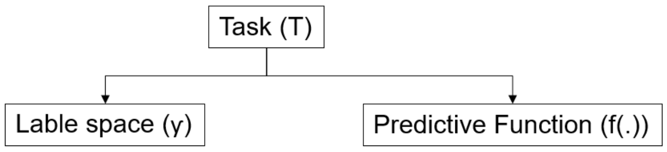


Figure 2.15: Task for a given domain $\mathcal{D}$

2.3.2. Formal Definition of Transfer Learning

After having defined the terminology, we can now proceed to give a formal definition of Transfer Learning with mathematical notations.

Transfer Learning [76]: Given a source domain $\mathcal{D}_S$ with a corresponding source task $\mathcal{T}_S$ and a target domain $\mathcal{D}_T$ with a corresponding task $\mathcal{T}_T$, transfer learning is the process of improving the target predictive function $f_T(\cdot)$ by using the related information from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

2.3.3. Categorization of Transfer Learning

Researchers have categorized TL on different aspects, however, we would focus on the most widely used categorization i.e. problem-based and solution based which can be seen in Figure 2.16

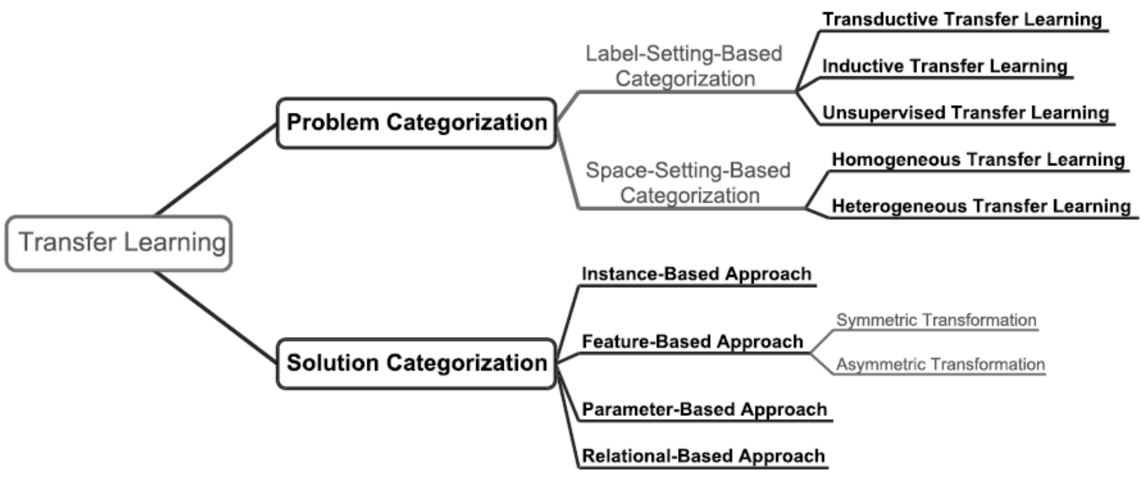

Transfer Learning
Problem Categorization
Label-Setting-Based Categorization
Transductive Transfer Learning
Inductive Transfer Learning
Unsupervised Transfer Learning
Space-Setting-Based Categorization
Homogeneous Transfer Learning
Heterogeneous Transfer Learning
Solution Categorization
Instance-Based Approach
Feature-Based Approach
Symmetric Transformation
Asymmetric Transformation
Parameter-Based Approach
Relational-Based Approach

Figure 2.16: [86] Transfer Learning Categorization

One of the most common categorizations is based on the availability of the labels in the source and target domains. However, in literature, we find inconsistency and ambiguity while defining the category of TL based on the availability of labels, Cook and Feuz [15] coined terms like informed and uninformed TL based on label availability, while in an exhaustive survey by Pan and Yang [50] they described the label based categorization simply as transductive TL if source domain is the sole contributor for the label information, inductive TL if the label information for the target domain instance is available and finally, unsupervised TL if label information for both the source and target domain is unavailable. This leads to ambiguity in information use, moreover, there is also inconsistency in the use of terms like supervised, semi-supervised and unsupervised TL based on the availability of label information like Daume [17] and Chattopadhyay [10] mentions that if the source domain is labeled completely and target label has limited or none labels then it is supervised and semi-supervised TL respectively, while Gong [27] and Blitzed [6] mention that if the source domain is labeled completely and target domain has limited or none labels then it is semi-supervised and unsupervised TL respectively. Therefore, for studying transfer learning it is always advisable to explicitly mention the conditions of label availability.

Another important categorization and very often seen in the literature is based on the consistency between the source's and target's feature space and label space namely Homogeneous TL and Heterogeneous TL.

Homogeneous Transfer Learning: $\mathcal{X}_S = \mathcal{X}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$

Heterogeneous Transfer Learning: $\mathcal{X}_S \neq \mathcal{X}_T$ or/and $\mathcal{Y}_S \neq \mathcal{Y}_T$

Where, $\mathcal{X}_S$ & $\mathcal{X}_T$ are the feature space of source and target domain respectively, and $\mathcal{Y}_S$ & $\mathcal{Y}_T$ are the label space of source and target domain.

The remedy to problems posed by TL has also been categorized in the literature as shown in Figure 2.16. Instance-based transfer learning approaches are mainly based on weighing the instances to cater to the probability discrepancies. Feature-based approaches work by creating new representations from the original features of the data; they can be further divided into two subcategories, that is, asymmetric and symmetric feature-based transfer learning. Asymmetric approaches transform the source features to match the target ones or the target to match the source ones. While symmetric approaches attempt to find a common latent feature space and then transform both the source and the target features into a new feature representation. The parameter-based transfer learning approach works on the use and re-use of the model's parameters generally by sharing them for both the source and target domain. Relational-based transfer learning approaches as the name suggests works on pre-defined relation between the domains i.e. they follow certain rules/guidelines learned from the source.

Transfer Learning for point clouds is especially challenging because of abundant parameters to study like scene, sensor, sensor location, etc. Each point in the cloud frame is a feature and since every frame has a different number of return points, we can conclude that the feature space for point clouds are not the same $(\mathcal{X}_S \neq \mathcal{X}_T)$ thus making it a Heterogeneous TL problem. [56][79] [1]

Heterogeneous TL problems have been solved by the researchers using a feature-based solution strategy, which in literature has been generally referred to as using the "Domain Adaptation (DA)" approach. There are multiple ways of categorizing DA methodologies for point clouds which can be seen in Figure 2.17, First, Input-based which contains directly feeding the 3D point clouds to a network [38][53][54] and converting the 3D point clouds to other representation [1] [16] [84] [77].

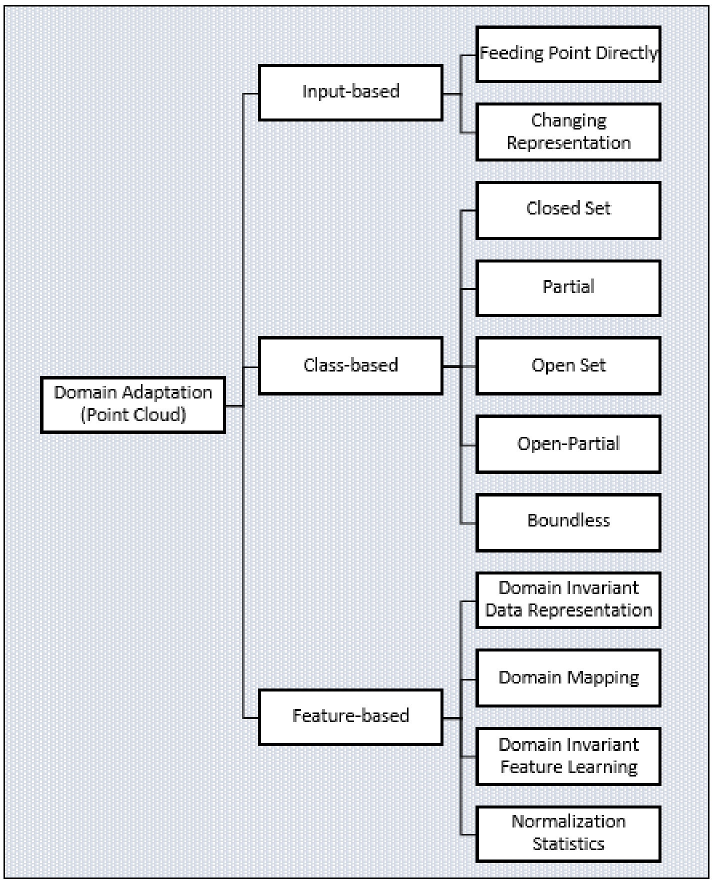


Figure 2.17: Various ways of categorizing Domain Adaptation for point cloud

The second way of categorizing DA for point cloud could be Class-based i.e. based on classes of the source and target domains, and the classes considered during the training. The Class-based DA can be further sub-categorized into Closed Set, Partial, Open Set, Open-Partial, and Boundless which can be seen in Figure 2.18. Most papers in the literature focus on Closed Set DA, where all classes are present in both the source and target domains. Partial DA is where a subset of the classes from the source domain is present in the target domain. Open Set DA is vice versa of the Partial DA. If both source and target set has some common and some unique classes, it is called Open-Partial DA. Boundless DA can be considered as Open Set DA in which all target classes are learned separately.

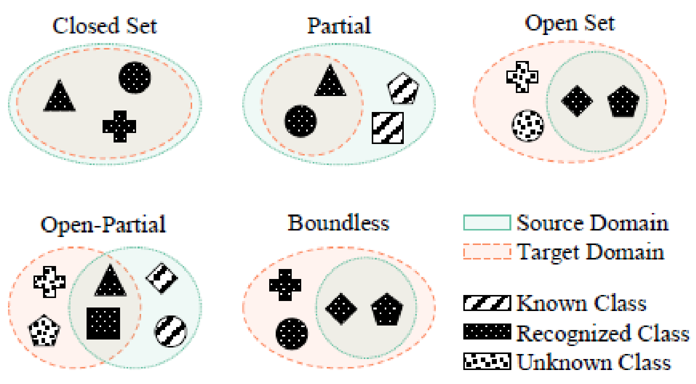


Figure 2.18: [69] Domain Adaptation categorization

Lastly, DA for point cloud could also be categorized as Feature-based. There are many Feature-based strategies followed in literature for DA, for example, Domain Invariant Data Representation, Domain Mapping, Domain Invariant Feature Learning, and Normalization Statistics. Among these strategies Domain Invariant Data Representation and Domain mapping are the most popular and highly employed by researchers.

Domain Invariant Data Representation [70]: It can be defined as a method of developing a hand-crafted approach to move the different domains into a common representation. The pipeline for it can be seen in Figure 2.19

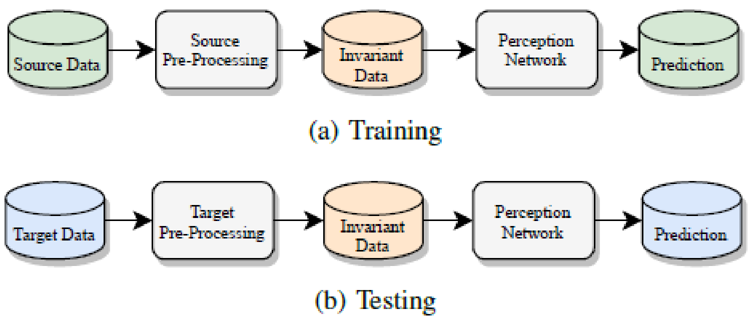


Figure 2.19: [70] Domain Invariant Data Representation

This approach utilizes the information from the source and target domain simultaneously, to come up with this common representation. However, it fails short to perform when we have no or very less information about the target domain. For such scenarios, the Domain Mapping approach comes in handy where very less information from the target domain is required. The methodology used may or may not require the label information of the target domain.

Domain Mapping [70]: In this approach the labeled source data is adapted to look like target data, thus, creating a pseudo-labeled target data. The pipeline for it can be seen in Figure 2.20

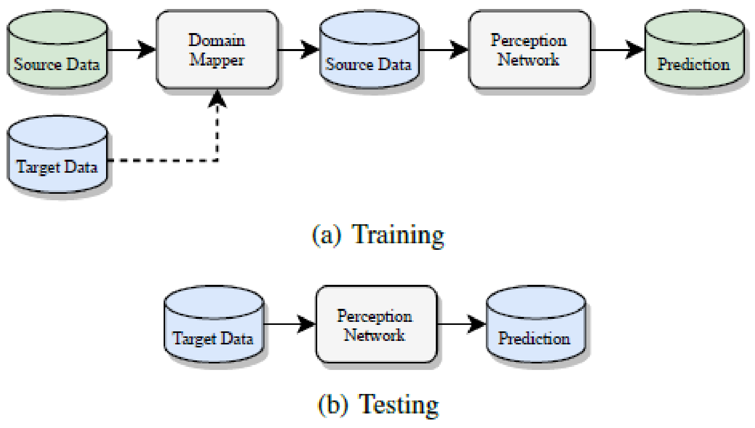


Figure 2.20: [70] Domain Mapping

Since, we chose the space-setting-based categorization of TL which uses feature space as the categorization factor to define our problem as Heterogeneous TL, therefore, to maintain consistency we choose feature-based solution strategy categorization to finalize the solution strategy for our scenario. When we speak about MEMS LiDAR, they are still under development and very few public datasets are available. Thus, not much data has been collected, therefore, Domain Mapping is the best-suited DA approach.

Domain Mapping has been under research for quite some time now but has been focused on images that are done adversarially and at pixel-level in the form of image-to-image translation with conditional GANs [59] [14]. For point clouds, not many adversarial networks are present which can map the domain from source to target in 3D. Moreover, most point cloud domain mapping is done by employing unmodified image GANs, and using top-view LiDAR images to translate the point cloud from one domain to another [60] [61] [62]. The problem with such an approach is the inability to modify the domain in the sensor perspective or the input space directly. This approach works fine for the domain mapping for Motorized Optomechanical LiDAR where the difference in sensor perspective is just the sampling channels, however, when a domain mapping between a Motorized optomechanical and a MEMS LiDAR has to be done, this strategy fails as in this scenario sampling pattern in sensor perspective is very different from each other. Therefore, a sensor perspective-based domain mapping is required.

Some methods employed in literature which adhere to sensor view DA methodologies are catered to solve the cross-sensor adaptation problem but only through up-sampling [71][63] or down-sampling [1] the channels in the elevation direction, since, the sensors used in publications are all Motorized Optomechanical LiDARs, they only differ in the number of elevation channels and not much in the sampling rate along the azimuth. Motorized Optomechanical and MEMS LiDAR differ not only in sampling along the elevation but may vary drastically along the azimuth. For MEMS since the sampling rate may vary across different sections of the sensor perspective depending on the scanning pattern, therefore employing a simple CNN-based interpolating architecture would not be sufficient, therefore, in this thesis, we aim to solve such adaptation problems in sensor perspective along the varying elevation and azimuth simultaneously.

2.4. Semantic Segmentation

Semantic segmentation has been researched for more than a decade. It has been researched extensively in domain of images for which there are abundant public datasets available like PascalVOC2012 [22], BSDS300 [48], BSDS500 [2], etc. The broad classification for semantic segmentation methods could be distinguished into Traditional Methods and Deep Learning Methods.

Semantic segmentation traditionally was treated as a clustering problem in the input feature space of the data. The general methodology uses predefined feature extractors to gather relevant information in the input data, these features are finally labeled using simple classifiers like Support Vector Machine(SVM) [30] or Random Forest (RF) [8]. The performance of these traditional methods quickly degrades with the increasing complexity of input data. Since point clouds pose much more complex decision-making than images, therefore it becomes next to impossible to design such feature extractors. The boom in deep learning provided the much-needed simplification in semantic segmentation of point clouds and therefore, soon deep learning-based methods were popularized.

Deep learning methods use deep neural networks with multiple hidden layers and multi-million learnable parameters to extract relevant features from the point cloud data. However, in most cases, it is not possible to know which feature is being extracted by the network or is more relevant for the task. Therefore, it acts as a black box. Semantic segmentation is at the forefront of the research in point clouds because of its ability to provide exhaustive information about the environment to the user. Researchers have categorized semantic segmentation in the domain of point cloud broadly into three categories, namely, Point-based methods, Image-based methods, and Voxel-based methods. These methods have been described in further detail in the upcoming sections.

2.4.1. Point-Based Methods

Point-based methods are very versatile techniques allowing the network to take an unstructured point cloud having different points per frame as input and output point-wise labels for the cloud. There has been a lot of research done on this methodology, to name a few examples PointNet [53], PointNet++ [54], PointSIFT [35], SO-Net [41], RandLA-Net [32], etc.

PointNet is the prime example of this category. It has been structured to accept unordered point cloud data as input and output semantic labels for each point. Its architecture can be subdivided into three modules. The first module helps to extract the global feature vector with the help of multi-layer perceptrons comprising max-pooling layers. The second module merges the local and global information by feeding the extracted global features to the network. Lastly, the third module is a joint alignment network which helps the architecture become invariant to basic point cloud transformations.

PointNet++ is an updated version of PointNet that aims to extract the local contextual features by applying PointNet recursively on a nested partitioning of the input points. The partitioning is created as neighborhood balls in Euclidean space and is parameterized by its centroid location and scale. The methodology works by processing the data sequentially within three abstraction layers. First, is the Sampling layer which samples the centroid in the Euclidean space using Farthest Point Sampling (FPS) algorithm. These centroid and scale describe the points in the local vicinity which are grouped by the next layer i.e. Grouping layer. Finally, these local environments are sent to the PointNet layer to extract the local features.

In some of the further networks, the concept remains the same as that of PointNet++ however, they employ different local environment generators like PointSIFT [35]. Some network adds to the concept of PointNet++ by adopting different sampling approach. SO-Net [41] initially models a point distribution by making a Self Organizing Map (SOM) of input. It performs a hierarchical based feature extraction of individual points and SOM nodes. Finally, it replaces the input points with single feature vectors. To reduce the computation models like RandLA-Net [32] use random sampling to reduce the computational overhead.

Researchers also proposed convolution-based architectures similar to that used in 2D convolutions in images to make unstructured point clouds be used in a structured fashion like PointCNN [42] which uses an X-Conv 3D convolution. A-CNN [37] proposes angular convolution, and KP-Conv [68], ShellNet [83], Tangent Convolution [66], PVCNN [45] are some of more examples of the convolution based architectures.

2.4.2. Image-Based Methods

Image-based methods change the representation of the 3D point cloud to an image. The images from the point cloud are fed to a neural network like U-Net [57]. The Image-based methods can be further sub-categorized into Multi-view Segmentation and Range Image Segmentation.

Multi-view Segmentation takes images generated by projecting the point cloud into the image plane from multiple perspectives. One of such networks proposed by Lawin et al. [40] considers images generated from a virtual camera rotated along a vertical axis fed to a fully convolutional network-based architecture to learn the information about the point cloud. Another example could be the network suggested by Boulch et al. [7] which divides the point cloud into a grid and generates images along each grid using a virtual camera. In these methods, the position of the virtual camera is particularly important for performance.

Range Image Segmentation strategies work by projecting the point cloud onto a spherical surface giving a 2D projection. These methods are sometimes pre-trained on image segmentation datasets like PascalVOC2012 and are fine-tuned on the point cloud image data. Some of the examples of this method are SqueezeSeg [77] which is an end-to-end deep learning method that outputs point-wise labels. Researchers further developed this method architecturally to improve performance giving Squeeze-Seg V2 [78], and also some other architectures like RangeNet++ [49] which uses spacial constants for retaining local affinities while predictions.

2.4.3. Voxel-Based Methods

Voxel-based methods have also been under research for a long time. It functions by discretizing the space into 3D voxels allowing a fixed-size input to be fed to the segmentation network. This methodology has a variety of pros like providing structure to the unstructured point cloud, reducing the input feature space tremendously, relatively requirement of less computational power in comparison to point-based methods, etc. However, it also has cons like introducing discretization error, the sensitivity of voxel size to capturing contours of geometry, etc. There are variety of Voxel-based segmentation networks like SEGCloud [67], PCSCNet [52], the ones provided by Huang et al. [34] and by Díaz-Medina et al. [19].

Point-based methodology has been under development for years and has provided appreciable results for semantic segmentation for point clouds, however, it is not suitable for our case of transfer learning problem because, since the number of features for each frame is different, thus it will cause the problem to fall under Heterogeneous TL problem which as mentioned in Section 2.3, is a complex problem to solve. Moreover, these networks are computationally heavy and thus their use in ADAS systems is challenging.

Image projection causes loss of 3D information due to discretization. Image-based networks can be used for our problem but voxel-based networks are a better candidate for our case because they retain the 3D information similar to that of point-based networks while making the decision.

The use Voxel-based method is highly dependent on the environment of deployment. In our case of TL, it suits better than other methods since it provides structure to point cloud and converts the feature space into fixed size, thus, making it into a problem of Homogeneous TL. Moreover, the discretization loss has been compensated by deploying an MLP based parallel pipeline for refinement of semantic labels after voxel-based predictions on the point cloud as shown in multiple architecture [52] [85] [56]

2.5. Contribution

In this thesis, we propose a strategy for Domain Adaptation based on a sub-sampling strategy in input feature space i.e. sensor's perspective. The strategy uses a scanning pattern for the target dataset's LiDAR, therefore, we also propose a strategy to extract a plausible scanning pattern for any LiDAR. The proposed strategy can be utilized to reduce the domain gap between LiDAR datasets collected using different categories of LiDARs.

3 Methodology

In this chapter, we explain the domain mapping technique used to map the source dataset i.e. SemanticKITTI to the target dataset i.e Cirrus, and the semantic segmentation approach which was undertaken to generate point-wise labels.

First, we start with adapting the SemanticKITTI using a scanning pattern adaptation methodology mentioned in Section 3.1. The adaptation technique is based on the sampling of the point from the source dataset using the scanning pattern of the target dataset. However, since the scanning pattern is generally a piece of confidential information that is not made public by the manufacturers of the LiDAR. Therefore, we start by generating a plausible scanning pattern of the target dataset using camera pinhole modeling as shown in Section 3.1.1. Once a plausible scanning pattern is created we use it to sample the points from the source data and thus, convert it to a pseudo target dataset.

The sampling strategy used causes a huge data loss only leaving a fraction of the points of the source dataset. We are using a supervised machine learning algorithm to generate semantic labels for the point cloud, which generally are data-hungry algorithms. Thus, this reduction in information is detrimental for training. Therefore, we utilize a point cloud densification pipeline to densify the point cloud of the source dataset and then adapt it to the target dataset using the same adaptation methodology. The pipeline for densification has been described in Section 3.2. The densification methodology is based on a depth completion network that inputs a sparse depth map along with an RGB image and outputs a dense depth map. The source dataset is projected into a depth map which is sparse in nature due to the sensor's resolution which in our case is LiDAR. This sparse image is fed to the depth completion network which outputs a dense depth map for the source dataset. And, finally, this dense depth map is reprojected to the point cloud and adapted to the target dataset.

In this thesis, the downstream task is chosen as semantic segmentation as it provides very rich information about the environment by providing point-wise labels for the point cloud. We choose a state-of-the-art voxel-based network, whose novel feature is cylindrical voxels instead of cubical voxels to account for the increasing sparsity of the point cloud with distance. The cylindrical voxel's size increase radially therefore, as the sparsity of the point cloud increases with distance so does the voxel volume, thus, creating a much more uniform distribution of points per voxel. The details of the model are described in Section 3.3

We train four semantic segmentation models with different data pre-processing done on inputs i.e. Domain Mapping strategy. First, Baseline SemanticKITTI is trained on the unaltered original SemanticKITTI dataset. Second, Cropped SemantickITTI is trained on cropped SemanticKITTI whose field of view is made similar to that of Cirrus (target dataset). Third, Adapted SemanticKITTI is trained on SemanticKITTI that has been adapted to Cirrus using the scanning pattern sampling strategy. Lastly, Densified Adapted SemanticKITTI is trained on SemanticKITTI that has been first densified using the depth completion methodology pipeline and then is also adapted to the Cirrus using the same scanning pattern sampling strategy.

The upcoming sections would go into detail about adaptation, densification, and the semantic segmentation process.

3.1. Adaptation

The adaptation methodology proposed in this thesis which is shown in Figure 3.1 requires knowledge of scanning pattern of the target. Thus, we first start with generation of scanning pattern for Cirrus.

3.1.1. Scanning Pattern Extraction

Cirrus dataset has been collected using two Luminar Hydra sensors one in Uniform configuration and the other in Gaussian configuration. Luminar Hydra is MEMS LiDAR and there is no documentation available for its scanning pattern. Therefore, we try to obtain the scanning pattern from the Cirrus data itself.

We assume a pinhole camera model to project a frame of point cloud in cartesian coordinates (X,Y,Z) to an image plane (U,V) using Equation 3.1. Here the extrinsic transformation matrix has been considered as identity. For the intrinsic matrix parameters, we use the camera calibration matrix provided in the documentation of the SematicKITTI. SemanticKITTI provides different calibration matrix like, P0 and P1 which denote rectified calibration (Cartesian to image) for left and right camera respectively. We use P0 matrix for transforming the point cloud to image. The projected points are then translated from pixel coordinates to radians using the FoV information of the Cirrus.

$$\begin{bmatrix} s.u \\ s.v \\ s \end{bmatrix} = \textbf{K.E} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

$$u = \frac{s.u}{s} \tag{3.2}$$

$$v = \frac{s.v}{s} \tag{3.3}$$

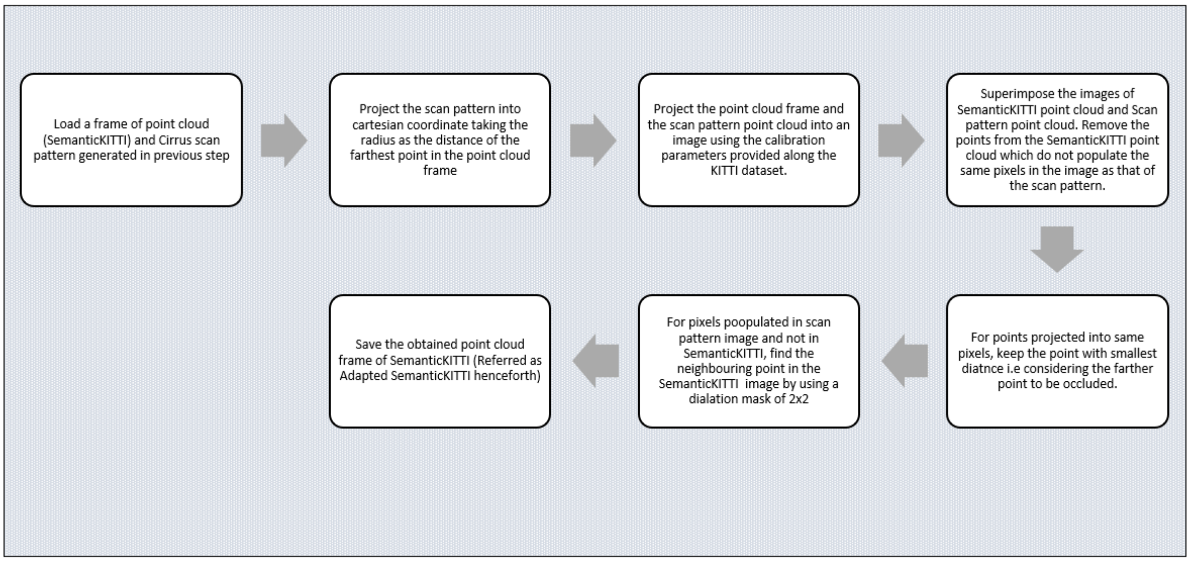
$$\mathbf{K} = \begin{bmatrix} f_x & \rho & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$\mathbf{E} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{3.5}$$

Where, **K** = Intrinsic Transformation matrix ; **E** = Extrinsic Transformation matrix ; s = scaling factor ; u, v = Pixel coordinate ; X,Y,Z = Cartesian coordinate ; f_x, f_y = Focal length in pixel units ; c_x, c_y = Projected optical centers ; ρ = Skewness ; **R** = Rotation matrix ; **t** = Translation vector

3.1.2. SemanticKITTI Adaptation

In section 3.1.1 we generated an estimated scanning pattern for the Luminar Hydra; in this section, we attempt to adapt the data of the SemanticKITTI to that of the Cirrus (Luminar Hydra's scanning pattern). The reason to adapt the SemanticKITTI to Cirrus is so that the data distribution for the SematicKITTI resembles that of the Cirrus.



Load a frame of point cloud (SemanticKITTI) and Cirrus scan pattern generated in previous step
Project the scan pattern into cartesian coordinate taking the radius as the distance of the farthest point in the point cloud frame
Project the point cloud frame and the scan pattern point cloud into an image using the calibration parameters provided along the KITTI dataset.
Superimpose the images of SemanticKITTI point cloud and Scan pattern point cloud. Remove the points from the SemanticKITTI point cloud which do not populate the same pixels in the image as that of the scan pattern.
For points projected into same pixels, keep the point with smallest diatnce i.e considering the farther point to be occluded.
For pixels poopulated in scan pattern image and not in SemanticKITTI, find the neighbouring point in the SemanticKITTI image by using a dialation mask of 2x2
Save the obtained point cloud frame of SemanticKITTI (Referred as Adapted SemanticKITTI henceforth)

Figure 3.1: Adaptation process pipeline

The proposed adaptation based on scanning patterns has multiple steps as shown in the flowchart above. The adaptation takes place for every frame individually. First, the scanning pattern is projected into cartesian space taking the distance of the farthest point of the frame of SemanticKITTI as the radius. Since we have the scanning pattern in the angular coordinates, we can use the azimuth angle, elevation angle, and radius to generate the point cloud. The second step is to project the scanning pattern point cloud and the frame point cloud into an image plane of the left camera of the KITTI setup. KITTI provides the calibration matrix from Velodyne to camera coordinate and camera coordinate to image coordinate. A simple matrix multiplication using the parameters provided in the calibration file projects the Velodyne point cloud to the camera image plane. The image plane is discretized into 1216 x 352 pixels.

There are cases when multiple points populate the same pixels when projected, however, we choose the nearest point (depth-wise in the forward direction) assuming the farther point gets occluded and assuming one response from one ray of a laser. The two images i.e of scanning pattern projection and SemanticKITTI projection are superimposed and a mask is generated from the scanning pattern projection and is used to obtain the points from the SemanticKITTI projection i.e. the points in the pixel are retained if the same pixel is also populated in the scanning pattern projection image.

There can also be cases when the pixels are populated in the scanning pattern projection image but are empty in the SemanticKITTI projection image. In such a case we create a dilation mask retaining the points which populate the neighborhood of the same pixel in the SemanticKITTI projection image. This methodology can also be used to adapt the dataset to any scanning pattern.

3.2. Point Cloud Densification

In this thesis, we use an Image-based depth completion strategy for LiDAR point cloud densification. Depth completion methods take sparse depth maps as inputs and predict dense depth maps as output. There are myriad applications of depth completion in various computer vision applications, autonomous diving applications, and 3D reconstruction for map generation in robotics.

LiDAR, a ranging sensor that generates a highly dense, rich, and informative modality knowledge of the outdoor environment falls short in providing a dense depth image when projected into an image plane. Therefore, a deep learning-based approach has been employed by researchers to generate a dense depth image of the surrounding from a sparse depth image using just sparse depth map information [72][21]. There has been involvement of data fusion strategies using multi-modal information like RGB images to enhance the depth completion networks [33][55][44][12]. The use of multi-modal information especially RGB images has been paving the way for research because it allows extracting different information cues, thus, making the networks much more reliant and robust.

The architecture of the network used for depth completion for point cloud densification is shown in Figure 3.2

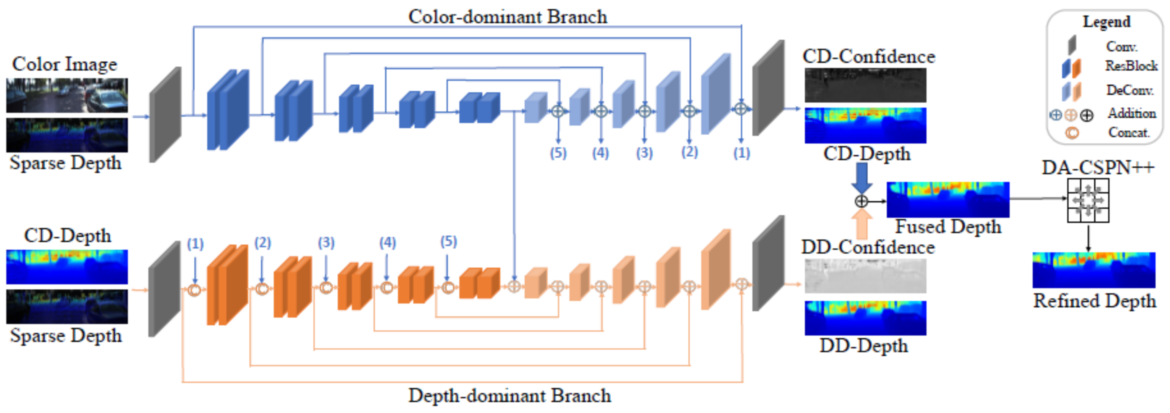


Figure 3.2: [31] Depth completion network architecture

The network shown in Figure 3.2 is structured into two components two-branch backbone and a refinement module. The output depth from the backbone network does not preserve the accurate depth values therefore a refinement module is used which makes the prediction more accurate, efficient, and effective.

3.2.1. Architecture

The network consists of two branches namely the color-dominant branch and the depth-dominant branch. Both of the branches are similar in structure following an encoder-decoder scheme.

3.2.1.1. Backbone Network

The components of the backbone network would be explained in detail in this section.

Color-dominant (CD) Branch

The branch takes a color image and sparse depth map as input and predicts a dense depth map as output. This branch has an encoder-decoder scheme with skip connections. The encoder is structured with a convolution layer and 5 residual blocks [29] and the decoder is structured with 5 deconvolution layers and 1 convolution layer. The convolution layers are accompanied by a batch normalization layer and a ReLU non-linearity function. The branch aims to extract color-dominant cues which allow the dense depth map to remain confident along the boundaries of the objects. However, the predictions could be sensitive to color or texture change.

Depth-dominant (DD) Branch

The architecture of this branch is the same as the CD branch but the features of the decoder of the CD branch are concatenated with the features of the encoder of this branch, however, the input for this branch is the sparse depth map and the dense depth map output of the CD branch. This branch predicts the information cues for dense depth maps which are overall reliable but lacks confidence at the boundary of an object.

Depth Map Fusion

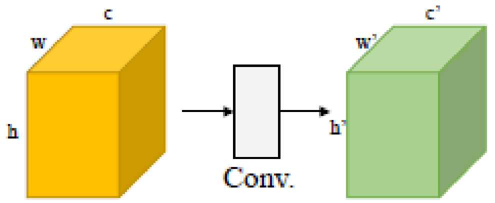
The depth map predicted from the color-dominant branch is reliable at the edges of the objects while the ones from the depth-dominant branch are reliable overall except at edges of objects. Therefore, they both compensate for each other's shortcomings, thereby, creating a much more accurate depth map. The fusion of the two depth maps is done using the confidence map of each branch's prediction as has been mentioned in FusionNet [73]. The mathematical formula used for fusion can be seen in Equation 3.6

$$\hat{D}_f(u,v) = \frac{e^{C_{cd}(u,v)} \cdot \hat{D}_{cd}(u,v) + e^{C_{dd}(u,v)} \cdot \hat{D}_{dd}(u,v)}{e^{C_{cd}(u,v)} + e^{C_{dd}(u,v)}} \tag{3.6}$$

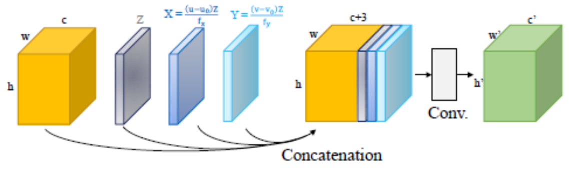
where, (u,v) are the pixel indexes, $\hat{D}_f$ is the fused depth map, $\hat{D}_{cd}$ and $\hat{D}_{dd}$ are the depth map from color-dominant and depth-dominant branch respectively.

Geometric Convolution Layer

3D geometric clues enhance the performance of the depth completion network, therefore, the network introduces geometric convolutions to encode the information of the 3D geometry. The geometric layer differs from traditional convolutions by concatenating the 3D position to the traditional convolution input tensor. The visual representation of the geometric layer can be seen in Figure 3.3.



(a) Traditional convolutional layer



(b) Geometric convolutional layer

Figure 3.3: [31] Comparison of traditional and geometric convolutional layer

The concatenated layers contains (X, Y, Z) which are the position encoding defined in Equation 3.7 - Equation 3.9

$$Z = D \tag{3.7}$$

$$X = \frac{(u-u_0)Z}{f_x} \tag{3.8}$$

$$Y = \frac{(v-v_0)Z}{f_y} \tag{3.9}$$

Where, D is the depth value, (u, v) is the pixel coordinates and u_0, v_0, f_x and f_y are the camera's intrinsic parameters.

3.2.1.2. Refinement Module: Dilated and Accelerated Convolutional Spatial Propagation Network (DA-CSPN++)

The prediction made by the backbone architecture does not maintain the input depth values at corresponding valid pixels. Therefore, a refinement module CSPN++ [13] was utilized to provide depth values at valid pixel indexes. The network utilizes a much more efficient version of CSPN++, by performing some modifications to the traditional CSPN++. It uses a dilation strategy to increase the propagation neighborhood and uses a parallel approach for the propagation of values from neighborhood pixels.

Mathematically, if D^0 is the depth map initially at step zero also referred to as the coarse depth map, then, the refinement done by the module outputs D^t after t iterations which is referred to as refined depth map as can be seen in Equation 3.10.

$$D_i^{t+1} = W_{ii} D_i^0 + \sum_{j \in \mathcal{N}(i)} W_{ji} D_j^t \tag{3.10}$$

Where, D^0 is coarse depth map; D^t is depth map at iteration t; i and j are the pixel indexes; $\mathcal{N}(i)$ is the neighborhood of pixel i and W_{ij} is the affinity between the pixels i and j.

3.2.2. Loss Function

The network uses ℓ_2 loss for training which has been defined mathematically in Equation 3.11.

$$\mathcal{L}(\hat{D}) = \|(\hat{D} - D_{gt}) \odot \mathbb{1}(D_{gt} > 0)\|^2 \tag{3.11}$$

Where, $\hat{D}, D_{gt}$ are the predicted and ground truth depth map respectively, $\mathbb{1}$ is the indicator function, and $\odot$ point-wise product. Since the ground-truth depth map has pixels with depth values and pixels without depth values, however, learning is done only on the pixels with depth values other pixels are ignored.

$$\mathcal{L} = \mathcal{L}(\hat{D}) + \lambda_{cd} \mathcal{L}(\hat{D}_{cd}) + \lambda_{dd} \mathcal{L}(\hat{D}_{dd}) \tag{3.12}$$

To supervise the CD and DD branches individually the network uses the loss function mentioned in Equation 3.12 for initial training. The λ_{cd} and λ_{dd} are the hyperparameters that govern the learning emphasis of the CD and DD branches respectively.

3.2.3. Training

3.2.3.1. Dataset Used

The network is trained of KITTI depth completion [72] dataset provided by KITTI suit [24]. The depth completion dataset provides RGB images and sparse depth maps which are created by projecting the 3D point cloud collected by Velodyne HDL-64E into an image plane. The RGB images provided are of 1216×352 resolution along width and height respectively. The analysis of sparse depth maps shows that it consists of 5% pixels with depth information, while the ground-truth consists of 16% such pixels [72]. The sample frames for KITTI depth with their RGB image, sparse depth projection, and ground truth can be seen in Figure 3.5.

3.2.3.2. Multi-stage Training [31]

A multi-stage approach was used to train the backbone, DA-SCPN++, and the complete model in increments, the sequence of which could be seen in Figure 3.4. The training is commenced with training the backbone network during which the learning rate of 0.001 was used, and a weight decay of $\frac{1}{2}$, $\frac{1}{5}$ and $\frac{1}{10}$ at epochs 10, 15, and 25 respectively. The backbone is trained for an in-total of 30 epochs. During this training, the loss function mentioned in Equation 3.12 is used to guide the learning of the CD and DD branches of the backbone. The hyperparameters λ_{cd} and λ_{dd} were set to 0.2 for initial epochs.

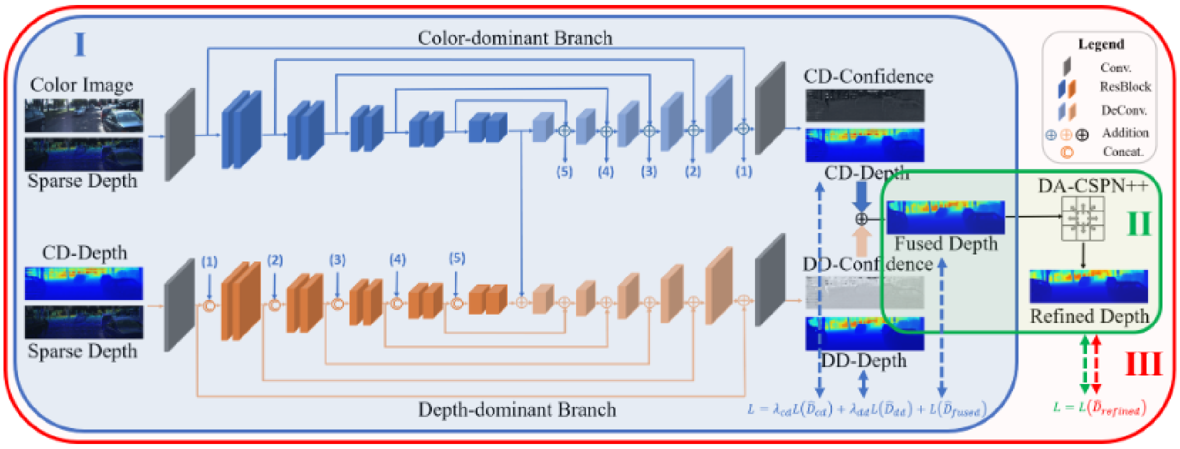


Figure 3.4: [31] Training architecture in multiple stages, first, only backbone is trained, second, only refinement module is trained and lastly, complete architecture is trained together

In the second stage of the training, the refinement module is trained, for which the weights of the backbone are frozen and the optimization of DA-CSPN++ parameters is done for two epochs at a learning rate of 0.001.

Lastly, the third stage of training optimizes the complete network all together. However, the initial learning rates for the backbone and refinement module is set to 0.02 and 0.002 respectively. A weight decay of $\frac{1}{2}$, $\frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{20}$, and $\frac{1}{50}$ was used at 10, 20, 30, 40, and 50 epoch respectively and the training was done for 75 epochs.

The hyperparameters used for all stages of training have been summarised in Table 3.1

Training Stage	**Parameters Optimized**	λ_{cd}	λ_{dd}	**Batch Size**	**Optimizer**
First	Backbone Network	0.2	0.2	6	Adam
Second	Refinement Module	0	0	6	Adam
Third	Backbone + Refinement	0	0	6	Adam

Training Stage	**Learning Rate**	**Epoch**	**Decay Factor:: Epoch Scheme**
First	0.001	30	($\frac{1}{2}$, $\frac{1}{5}$, $\frac{1}{10}$)::(10,15,25)
Second	0.001	2	-
Third	0.002, 0.02	75	($\frac{1}{2}$, $\frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{20}$, $\frac{1}{50}$)::(10,20,30,40,50)

Table 3.1: Hyperparameter summary for depth completion network

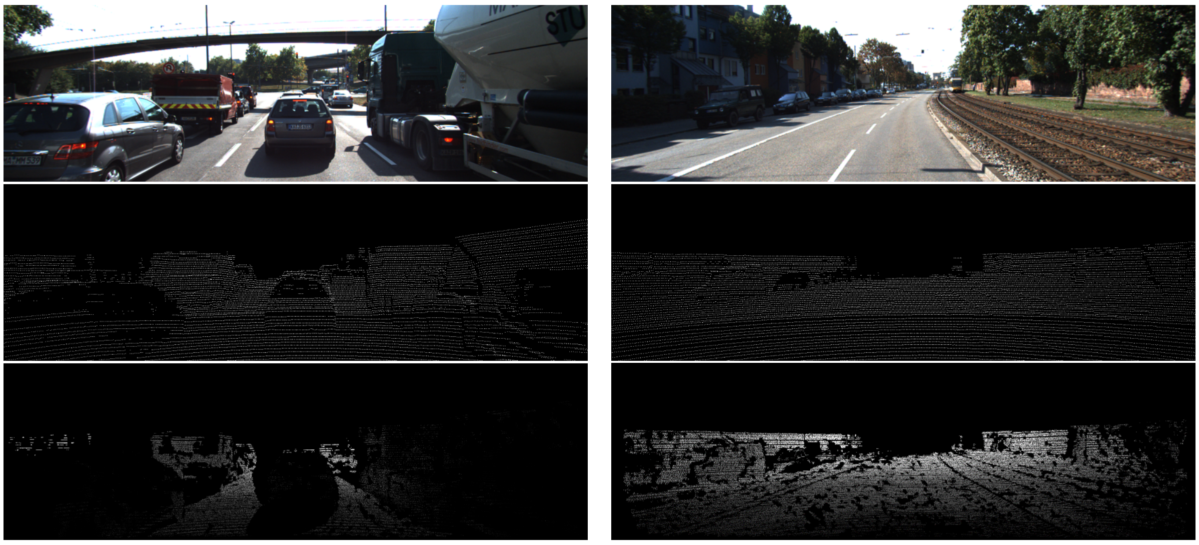


Figure 3.5: Example of KITTI depth dataset, 1) RGB Image 2) Sparse Depth map 3) Ground Truth

3.2.4. Evaluation [31]

The model's performance as mentioned by researchers [31] outperforms all of the existing state-of-the-art depth completion networks based on the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) metric. The model shows an improvement of 6.16mm and 8.28mm in RMSE and MAE over the GuideNet [65] which was the best performing network on KITTI depth previously. The network's runtime was also expedited by using the dilated and accelerated model of CSPN++, thus, making it outperform most state-of-the-art networks.

3.3. Semantic segmentation

This section explains the semantic segmentation procedure. It explains the datasets used in detail, model architecture, loss functions, and label mapping strategy to map source labels to target labels.

3.3.1. Datasets

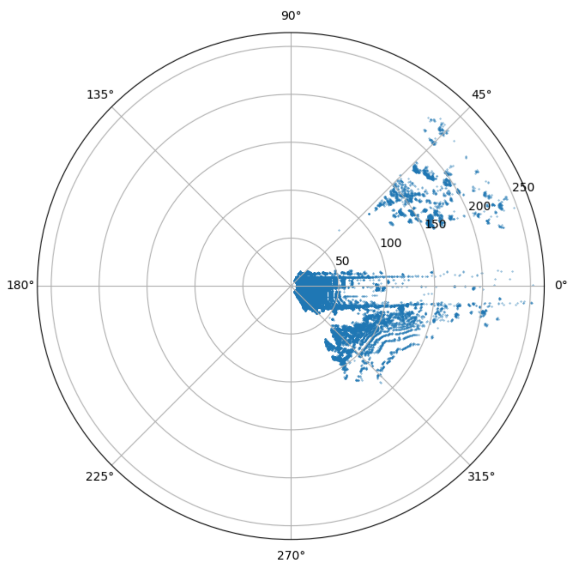
In our scenario, we need an abundant amount of point-wise labeled LiDAR data frames to train the semantic segmentation network. Generation of point-wise annotations is a time-consuming task and requires a significant amount of human effort and monetary support. Therefore, to reduce such effort we aim to find a way to use the data which has already been annotated and use its information to annotate the new data. Many domain mapping strategies have been researched to reduce the distance between the available and target dataset. In the upcoming sections, we will discuss the details of the source dataset and the target dataset.

3.3.1.1. Cirrus: Target Dataset

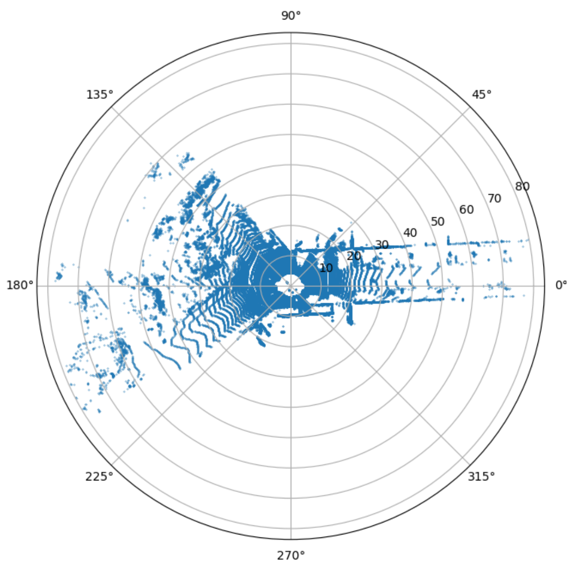
Cirrus [75] is a long-range LiDAR dataset that is chosen as the target domain for our experiments for the downstream task of semantic segmentation. We discussed Cirrus in Section 2.2.2.1, Figure 3.6 shows a bird-eye view polar plot of a single frame of cirrus dataset.

We aim to predict the point-wise labels of the dataset by learning from the source dataset. Cirrus contains sequences that comprise almost 60% of highway scenarios in the south of Germany. It contains seven sequences of which the first four are collected on highways whereas the latter three are collected in an urban scenario. In our experiments, we use Gaussian data frames as they provide denser distribution of points associated with objects in front of the vehicle. We can see the frames of the Cirrus dataset on the highway and urban scenarios in Figure 3.7. Table 3.2 describes the classes of cirrus dataset annotations (Bounding boxes). The visualization of data distribution of cirrus frames along the different axis can be seen in Figure 3.8.

Cirrus provides ground-truth labels as bounding boxes, therefore, we provide the points inside the bounding boxes with the same label as that of the bounding box it lies. After a model predicts semantic labels of points in a frame of cirrus we can only evaluate the points inside the bounding box. Therefore, limiting the evaluation to only a fraction of points of a frame.

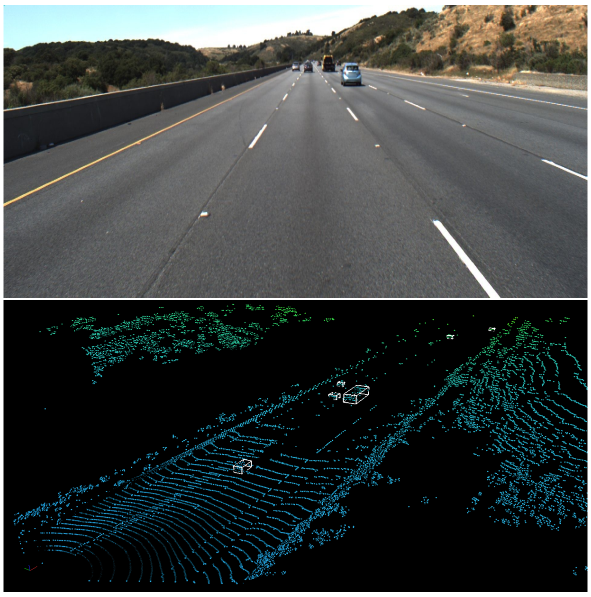


(a) Cirrus

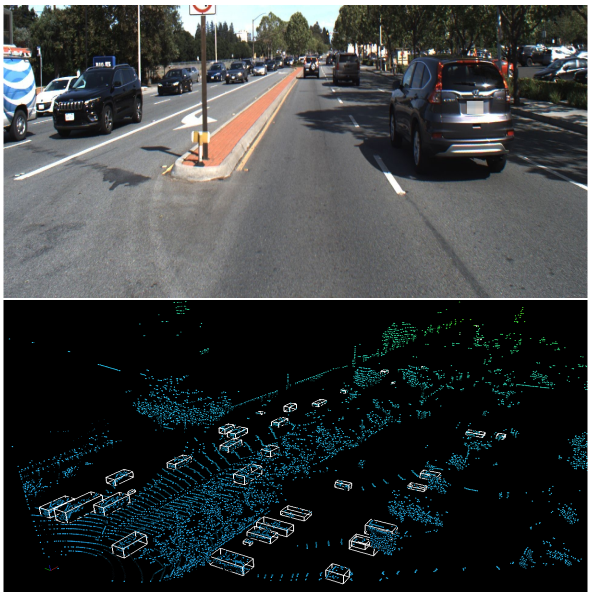


(b) SeamnticKITTI

Figure 3.6: Bird-eye view of Cirrus and SemanticKITTI dataset

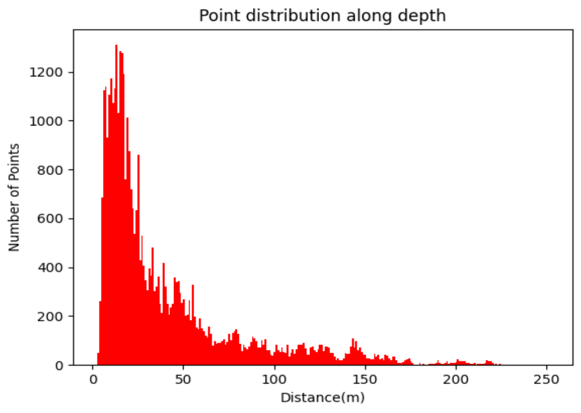


(a) Highway

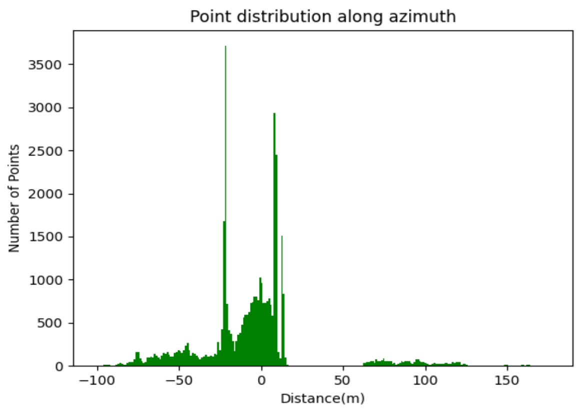


(b) Urban

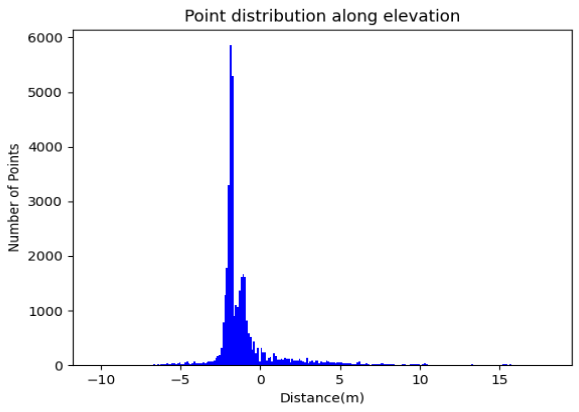
Figure 3.7: Cirrus highway and urban scenario



(a) X-axis: Depth



(b) Y-axis: Azimuth



(c) Z-axis: Elevation

Figure 3.8: Data distribution of Cirrus along different axis

Label Class	Description
vehicle	Consists of different types of cars on the road
large-vehicle	Consists of SUVs, jeeps, trucks, buses and vehicles with continuous body
trailer	Vehicles that has separated front and back. The two parts of the body are connected via movable joint
pedestrian	People on foot
bicycle	Bicycles
animal	All types of animals in the scene
wheeled-pedestrian	Riders on bicycles or motorcycle
motorcycle	Motorcycles and scooters
unknown	Random objects of the road that can be considered as obstacles

Table 3.2: Cirrus label (Bounding Boxes) details

3.3.1.2. SemanticKITTI: Source Dataset

As we discussed in Section 2.2.1.1, SemanticKITTI [4] is a sequential LiDAR dataset with 360^o horizontal FoV. The dataset has been collected on a highway, approximately 30%, and an urban environment. The dataset consists of point-wise annotated frames with 28 classes. Table 3.3 defines the category of labels for semanticKITTI, moreover, other-structures, other-objects, and other-vehicle classes are the fallback classes for their subsequent main category. Labels such as vehicular and pedestrians are further sub-classified into moving and non-moving classes. Points that appear due to reflection and other inconsistencies are labeled as "outliers".

The dataset contains labels for training sequences (00 to 10) which are available under creative common license and hence can be used for our research. It also contains test sequences (11 to 21) for which the labels are not publicly available but the evaluation could be done online via benchmark setup available on CodaLab Competition[1]. Figure 3.9 shows a few frames from different sequences along with their color mapping. We can see the point distribution of SemanticKITTI along the different axes in Figure 3.10.

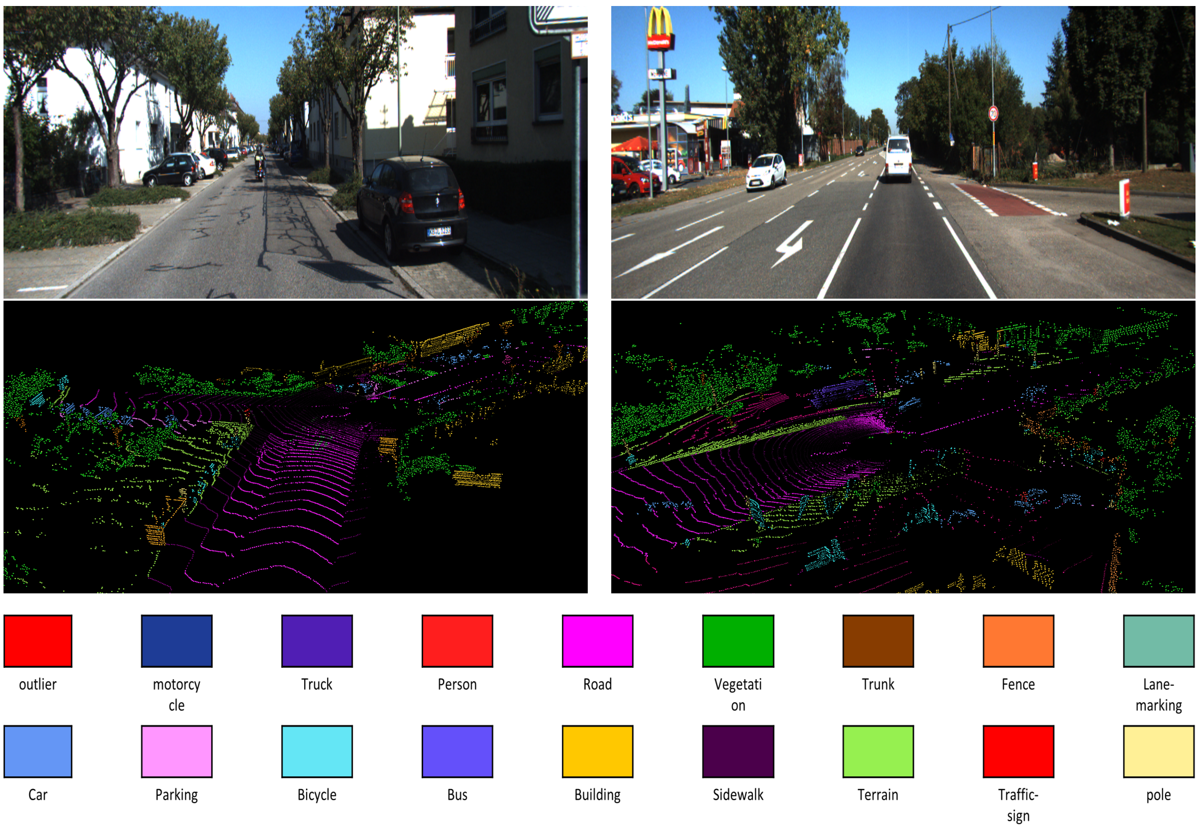
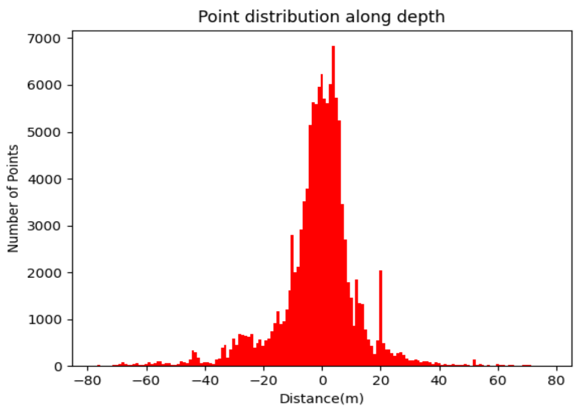
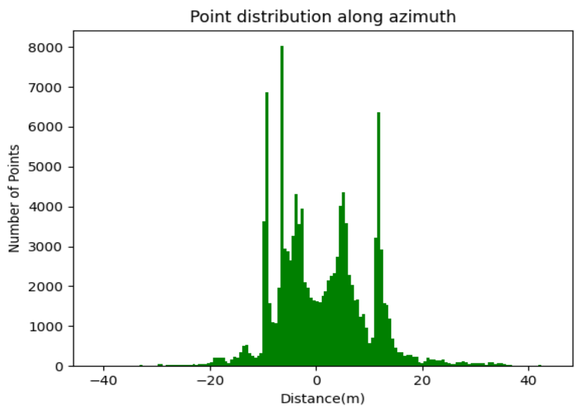


Figure 3.9: Example of SemanticKITTI dataset

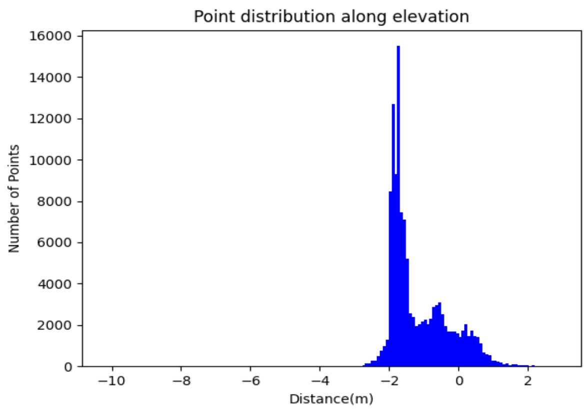
[1]`https://competitions.codalab.org/competitions/20331`



(a) X-axis: Depth



(b) Y-axis: Azimuth



(c) Z-axis: Elevation

Figure 3.10: Data distribution of original SemanticKITTI along different axis

Label Class	Description
road	All the driveable area of a scene
sidewalk	Area that is reserved for pedestrians and bicycles. Cars are not allowed in this area
parking	Road area that is explicitly reserved for parking
other-ground	Includes ground area that is not identified as either of the road, sidewalk or parking. For example, paved path of a gas station
building	Whole building and its components
other-structure	Includes other vertical structures that include tunnel walls, bridge posts etc
car	Cars, SUVs, jeeps and continuous body shape vans
truck	Trucks and vans with separated body
bicycle	Bicycles without the rider
motorcycle	Motorcycle, rider is not included
other-vehicle	Trailers, caravans, fallback types of vehicles or any type of vehicle that is not defined in other categories
vegetation	Bushes, schrubs, tree-tops or any identifiable vegetation
trunk	Tree trunk
terrain	Grass, soil or any other type of horizontal vegetation
person	Humans that are sitting, standing or moving with their own legs
bicyclist	Human driving a bicycle or standing by the bike
motorcyclist	Humans driving a motorcycle or standing near the motorcycle
fence	Separators like barriers, small walls or fences
pole	Traffic sign post or lamp post
traffic sign	Traffic sign excluding its mounting
other-object	Advertising columns
outlier	Points caused by inaccuracies and reflections in the scan. If we don't know the source of the point we also label it as outlier

Table 3.3: SemanticKITTI label details [4]

3.3.2. Model Architecture

In this thesis, we use an end-to-end deep learning model to predict the semantic labels of the input point cloud. The choice of the network selected is critical for the approach undertaken. The various categories of the semantic segmentation network have been described in Section 2.4. For our problem we choose a voxel-based network architecture [85].

The network architecture comprises two major components. First, 3D cylindrical partitioning of the point cloud, and second, a 3D U-net to process the partitioned data. The network architecture can be seen in Figure 3.11. To extract the 3D geometric features of the object from the point cloud the network utilizes an asymmetric 3D convolution in the hidden layers. Instead of using conventional 3D convolution kernels, the network uses an asymmetrical residual block (A). To retain the contextual information of the point cloud, the network has a dimensional-decomposition based context modeling module (DDCM) at the end of the U-net.

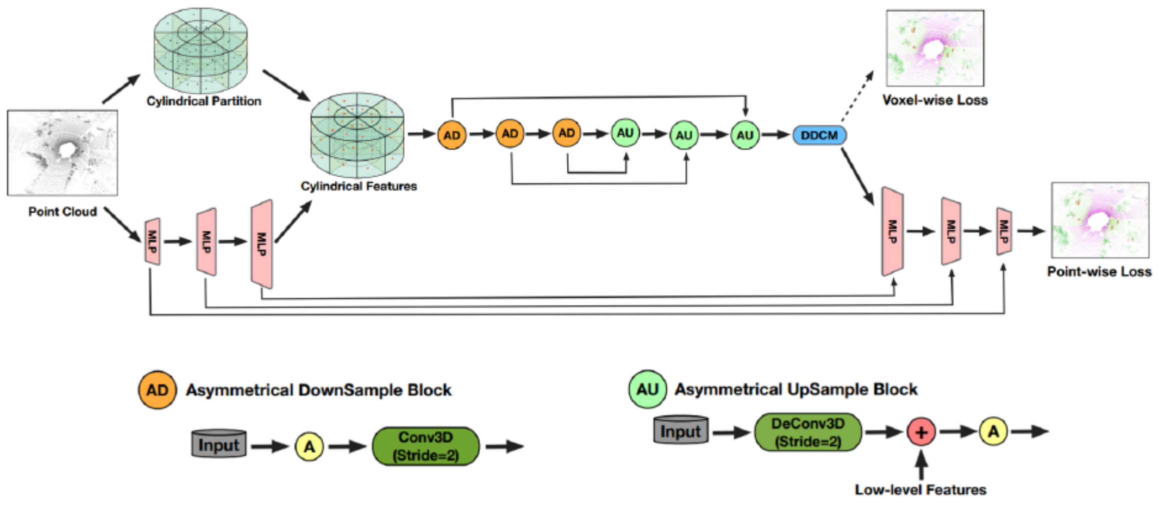


Figure 3.11: [85] Semantic Segmentation network architecture

3.3.2.1. Cylindrical Voxelization

In the traditional voxelization scheme, the voxels are cube-shaped having constant volume. Point clouds generated using LiDAR get sparser with the increase in distance due to the nature of the sampling strategy used in LiDARs. Using cubic voxels throughout the 3D space of the point cloud results in cubic voxels closer to the origin being highly populated with points but the ones farther away from the origin are mostly unoccupied.

The chosen network utilizes a cylindrical voxel scheme, dividing the point cloud 3D space into cylindrical voxels, therefore, having different volumes along the radius of the cylinder. This voxelization is performed in the cylindrical coordinate system, since the volume of the cylindrical voxels increases along the radius and the point distribution of the cloud gets sparser, it results in balanced point distribution per voxel.

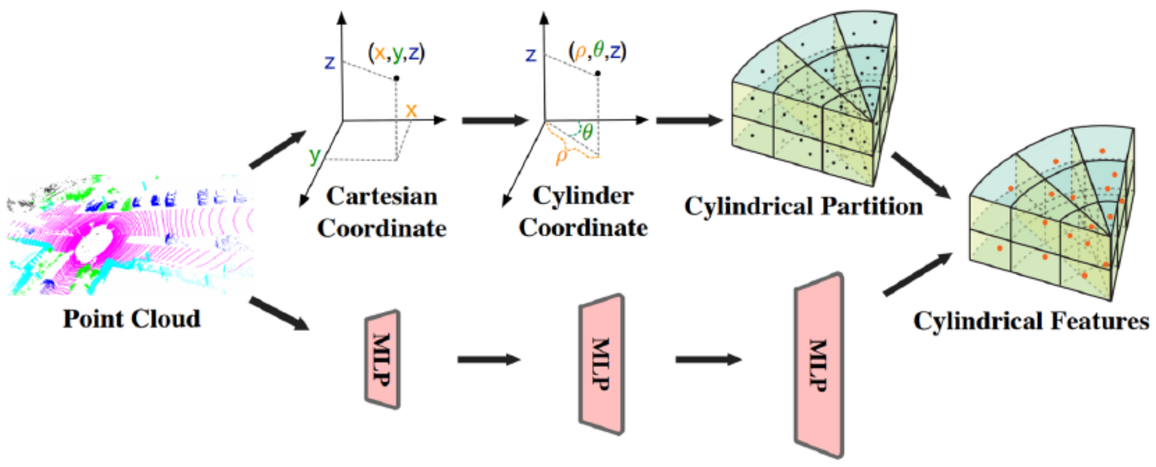


Figure 3.12: [85] Pipeline for 3D cylindrical voxelization

Figure 3.12 shows the pipeline for generation of cylindrical voxel scheme. First, the point cloud is converted from cartesian coordinate system (x, y, z) to cylindrical coordinate system (ρ, θ, z), where ρ represents the radius, θ represents the azimuthal angle and z represents the height of the point. Second, the transformed data is partitioned into a cylindrical grid of a fixed ratio, which is given as a hyper-parameter. As we move radially out, the volume of the voxels increases. The partitioned data is also assigned information extracted by feeding the point cloud to a Multi Layered Perceptron (MLP) based Pointnet [53]. Dimensions of the feature vector (C) extracted from MLP are also provided as a hyper-parameter to the network. Finally, the cylindrical grid is flattened starting from zero degrees, therefore creating a point cloud having a cylindrical representation. The representation generated is $(C \times H \times W \times L)$, where H, W, and L represent the radius, azimuth, and height of the voxel containing the point respectively.

3.3.2.2. Asymmetric Residual Block

The scenario presented in the dataset consists of multiple cubical objects of interest, for example, trucks, cars, busses, trailers, etc. Study [20] showed that features extracted by criss-cross weights of convolution kernels are better than traditional convolution kernels. An asymmetrical residual block is used to strengthen the vertical and horizontal response of the kernel. The block is designed having a $3 \times 1 \times 3$ convolution kernel followed by a $1 \times 3 \times 3$ convolution kernel and also a branch with $1 \times 3 \times 3$ kernel followed by a $3 \times 1 \times 3$ kernel. Finally, the responses of both the branches are added together. The asymmetrical block can be seen in Figure 3.13

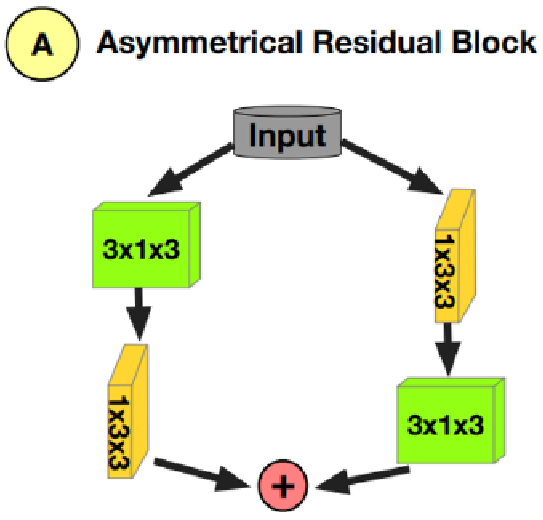


Figure 3.13: [85] Asymmetrical Residual Block

Asymmetrical block shown in Figure 3.13 is used for up-sampling and down-sampling block as shown in architecture in Figure 3.11. For downsampling 3D convolution with stride = 2 is performed on the compression side of the U-net while in the upsampling, asymmetric residual block fuses the low-level feature. The skip layers from the equal level downsampling layer are provided to upsampling block to provide spatial information lost during downsampling. The asymmetric block also helps reduce the computational and memory overhead compared to using a 3D kernel $(3 \times 3 \times 3)$ by reducing the parameters to be trained. It provides the same receptive field but bears 33% less computational cost.

3.3.2.3. Dimension Decomposition Context Modeling

Global contextual information plays a vital role in improving the results in the context of semantic segmentation. Point cloud provides very dense and rich information about its environment which has diverse information. To capture the contextual information of the environment, research [82] showed that a high-rank tensor is required which might be computationally expensive. Therefore, the used architecture utilizes high-rank matrix decomposition theory [11] to reduce the high-rank matrix into three low-rank matrices for capturing the context of height, depth, and width. The three individual contextual information are combined to provide a global context.

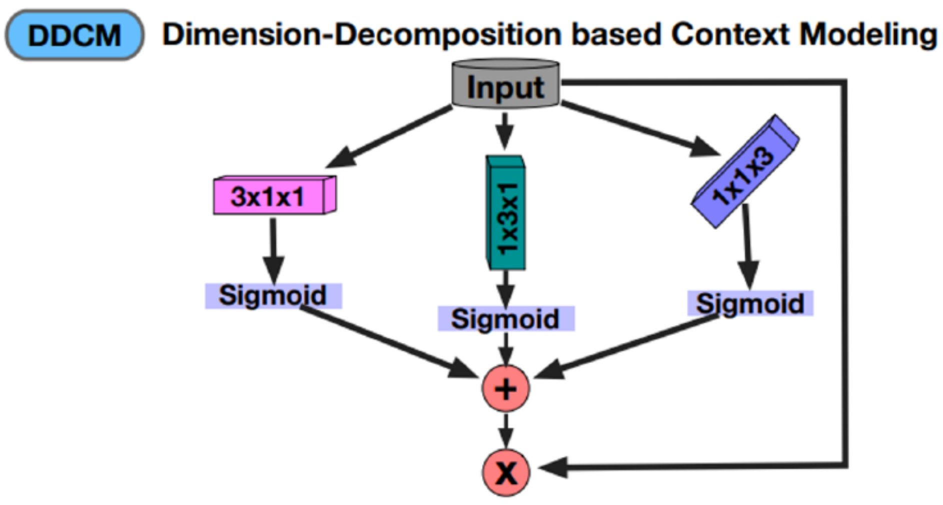


Figure 3.14: [85] Dimensional Decomposition based Context Modeling (DDCM)

The DDCM block used in the architecture can be seen in Figure 3.14. The three kernels of dimensions $3 \times 1 \times 1$, $1 \times 3 \times 1$ and $1 \times 1 \times 3$ followed by a sigmoid which modulates the results are used to encode the information from three different dimensions. The weights of kernels from three dimensions contain co-occurring contextual information from three different views. Finally, all these individual dimensional information are summed up to provide the global context.

3.3.2.4. Point-wise Label Refinement module

Voxel-based semantic segmentation architecture generally suffers from discretization loss i.e. the predictions of the network are on the voxel and not on the points themselves. The prediction made on the voxel is dispersed to all the points within the voxel, which in the case of a voxel consisting of points from multiple classes results in misclassification. Therefore, to solve this problem the architecture utilizes a point-wise refinement module after the DDCM module.

This module works by projecting the voxel-wise feature extracted initially before U-net onto the point-wise data based on the point-voxel mapping. The module provides both of the point features (point-data and voxel-data) before and after 3D convolution to compare them and finally combines this information to output point-wise refined labels.

3.3.3. Loss Function

Machine learning problems are treated as optimization problems. Every optimization problem has an objective function with an aim for it to be maximized. This maximization problem is taken care of by changing the variable parameters and in the case of machine learning these parameters are called weights and biases. However, we generally convert the maximization to minimization problem by changing the objective function to a Loss function. In ML algorithms we aim to adjust the parameters to reduce the error gradient and thereby minimize the loss function or maximize the objective function. The aim of converting the objective function to a loss function is to use gradient descent methodology which is predominantly used for convex optimization problems. There are quite a few loss functions available for semantic segmentation, however, we will focus on the ones being used in the defined architecture.

3.3.3.1. Weighted Cross Entropy Loss

Cross entropy loss measures the error between two distributions having a certain probability distribution. It is a derived and simplified form of Kullback Leibler divergence. Which is a framework of Maximum likelihood, where the aim is to find optimum parameters which tend to reduce the errors between the model's predicted probability distribution and the ground truth distribution provided by the training dataset. In research cross entropy loss has been used for multi-class classification problems, where the output from the model is the probability of an instance belonging to one of the independently distributed classes. In the scenario of semantic segmentation of point clouds, cross entropy loss is applied to all points individually i.e. for each point model predicts its probability vector consisting of the probability of the point belonging to each class. This predicted vector is compared with a ground truth vector which is one-hot encoded to the ground truth class of the point in focus. The point-wise cross entropy is averaged over all the points in a frame of cloud, therefore giving a metric for comparison. Since the point-wise loss is averaged, it implies equal learning for all points in the training data. Therefore, if the dataset is biased which most training datasets are generally, it would result in the model learning being biased towards the dominant class.

To counter the problem of biased dataset, researchers came up with weighing strategy [47]. The weighted cross entropy loss is given by:

$$\mathcal{L}_{WCE} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M} w_k \times y_m^k \times \log(h_\theta(x_m, k)) \tag{3.13}$$

Where, M is number of training samples, K is the number of classes, w_k is the weight for class k, y_m^k is target label for the training sample m and class k, x_m is an instance of training sample and h_θ is the neural network with weights parameterized as θ.

3.3.3.2. Lovasz-Softmax Loss

One of the most commonly used metrics for semantic segmentation is Intersection over union (IoU), also referred to as Jaccard Index. In the domain of semantic segmentation, Lovasz-softmax loss allows a network to optimize mean IoU directly, thus allowing a network to perform better on the training dataset.

For a given ground truth label vector as y^* and predicted label vector as y, the Jaccard index for a class c is in the range [0,1] and is calculated by:

$$\mathcal{J}_c(y^*, y) = \frac{|\{y^* = c\} \cap \{y = c\}|}{|\{y^* = c\} \cup \{y = c\}|} \tag{3.14}$$

From Equation 3.14, the corresponding loss function can be derived as:

$$\Delta_{\mathcal{J}_c}(y^*, y) = 1 - \mathcal{J}_c(y^*, y) \tag{3.15}$$

In the scenario of multi-class classification mean IoU (mIoU) is calculated by averaging the Jaccard index of all the classes in the training dataset.

Lovasz-softmax aims to improve the mIoU evaluation metric by generating convex surrogates to sub-modular loss functions [5]. It attempts to smoothen the discrete loss to allow the Jaccard index to be optimized in a continuous function framework. It provides an extension that is based on sub-modular analysis which maps the misprediction of the model to a set of real numbers. Considering M_c as the mispredicted points for class c, then the loss could be also written as:

$$\Delta_{\mathcal{J}_c} : M_c \in \{0,1\}^p \mapsto \frac{|M_c|}{|\{y^* = c\} \cup M_c|} \tag{3.16}$$

Lovasz extension is a convex closure of submodular set function and is defined as:

$$\overline{\Delta} : \mathbf{m} \in \mathbb{R}^p \mapsto \sum_{i=1}^{p} m_i g_i(\mathbf{m}) \tag{3.17}$$

where,

$$g_i(\mathbf{m}) = \Delta(\{\pi_1, ..., \pi_i\}) - \Delta(\{\pi_1, ..., \pi_{i-1}\}) \tag{3.18}$$

m is the vector of errors and π is the permutation ordering of components of **m** in descending order.

Now, $\overline{\Delta_{\mathcal{J}_c}}$ is the Lovasz extension applied to $\Delta_{\mathcal{J}_c}$, which is also the resulting surrogate loss i.e. Lovasz hinge applied to Jaccard Loss [81]. The resulting output **F** is a composition of piece-wise linear functions.

$$loss(\mathbf{F}(c)) = \overline{\Delta_{\mathcal{J}_c}}(\mathbf{m}(c)) \tag{3.19}$$

Moreover, in case of multiple classes the Lovasz Softmax loss can be defined as:

$$loss(f) = \frac{1}{|C|} \sum_{c \in C} \overline{\Delta_{\mathcal{J}_c}}(\mathbf{m}(c)) \tag{3.20}$$

3.3.3.3. Overall Training Loss Function

The network shown in Figure 3.11 uses a combination of two loss components namely, point-wise loss and voxel-wise loss. The final loss is the combination of these losses. Voxel-wise loss in itself is a combination of two losses. First, Weighted cross-entropy loss, and second, Lovasz-softmax loss. The details of these losses have been described in Section 3.3.3.1 and 3.3.3.2. Lovasz loss helps optimize the intersection-over-union score directly while weighted cross-entropy loss contributes to improving point accuracy. Point-wise loss comprises weighted cross-entropy loss.

The overall loss function is optimized using Adam [36] optimizer and output inferred after point-wise refinement is considered.

$$\mathcal{L}_{complete} = \mathcal{L}_{voxel-wise} + \mathcal{L}_{point-wise} \tag{3.21}$$

where,

$$\mathcal{L}_{voxel-wise} = \mathcal{L}_{WCE} + \mathcal{L}_{Lovasz}$$
$$\mathcal{L}_{point-wise} = \mathcal{L}_{WCE}$$

3.3.4. Label Mapping

Our source and target datasets namely, SemanticKITTI and Cirrus respectively have different label domains. Therefore, to generate semantic labels in the label space of Cirrus from a network trained on the label space of SemanticKITTI, we define a label mapping from source to target domain.

SemanticKITTI contains the labels like vehicles and pedestrians which are further sub-classified into moving and static classes, however, these sub-categorizations are merged into a single class. And classes like other-structure, other-objects, etc were mapped to unlabeled and are ignored while training. The intra SemanticKITTI mapping used can be seen in Table 3.4

The network is trained on the label space of the source (SemanticKITTI) and provides inference on the target (Cirrus) in the label space of SemanticKITTI. To evaluate the Cirrus dataset, we need the inference in the Cirrus label space. Therefore, we use the custom-generated label space mapping from SemanticKITTI to Cirrus to convert the inference on Cirrus from SemanticKITTI label space to Cirrus label space. The label mapping used for cross-dataset label conversion can be seen in Figure 3.15. The mapping was generated to keep the semantic and geometric information between the datasets the same. Moreover, classes in SemanticKITTI which do not have a semantic representation in Cirrus were ignored during evaluation.

Original Class Value	**Original Semantic meaning of class**	**Mapped Semantic meaning of class**
0	Unlabeled	Unlabeled
1	Outlier	Unlabeled
10	Car	Car
11	Bicycle	Bicycle
13	Bus	Other-vehicle
15	Motorcycle	Motorcycle
16	On-rails	Other-vehicle
18	Truck	Truck
20	Other-vehicle	Other-vehicle
30	Person	Person
31	Bicyclist	Bicyclist
32	Motorcyclist	Motorcyclist
40	Road	Road
44	Parking	Parking
48	Sidewalk	Sidewalk
49	Other-ground	Other-ground
50	Building	Building
51	Fence	Fence
52	Other-structure	Unlabeled
60	Lane-marking	Road
70	Vegetation	Vegetation
71	Trunk	Trunk
72	Terrain	Terrain
80	Pole	Pole
81	Traffic-sign	Traffic-sign
99	Other-object	Unlabeled
252	Moving-car	car
253	Moving-bicyclist	bicyclist
254	Moving-person	person
255	Moving-motorcyclist	motorcyclist
256	Moving-on-rails	Other-vehicle
257	Moving-bus	Other-vehicle
258	Moving-truck	truck
259	Moving-other	Other-vehicle

Table 3.4: Intra SemanticKITTI mapping

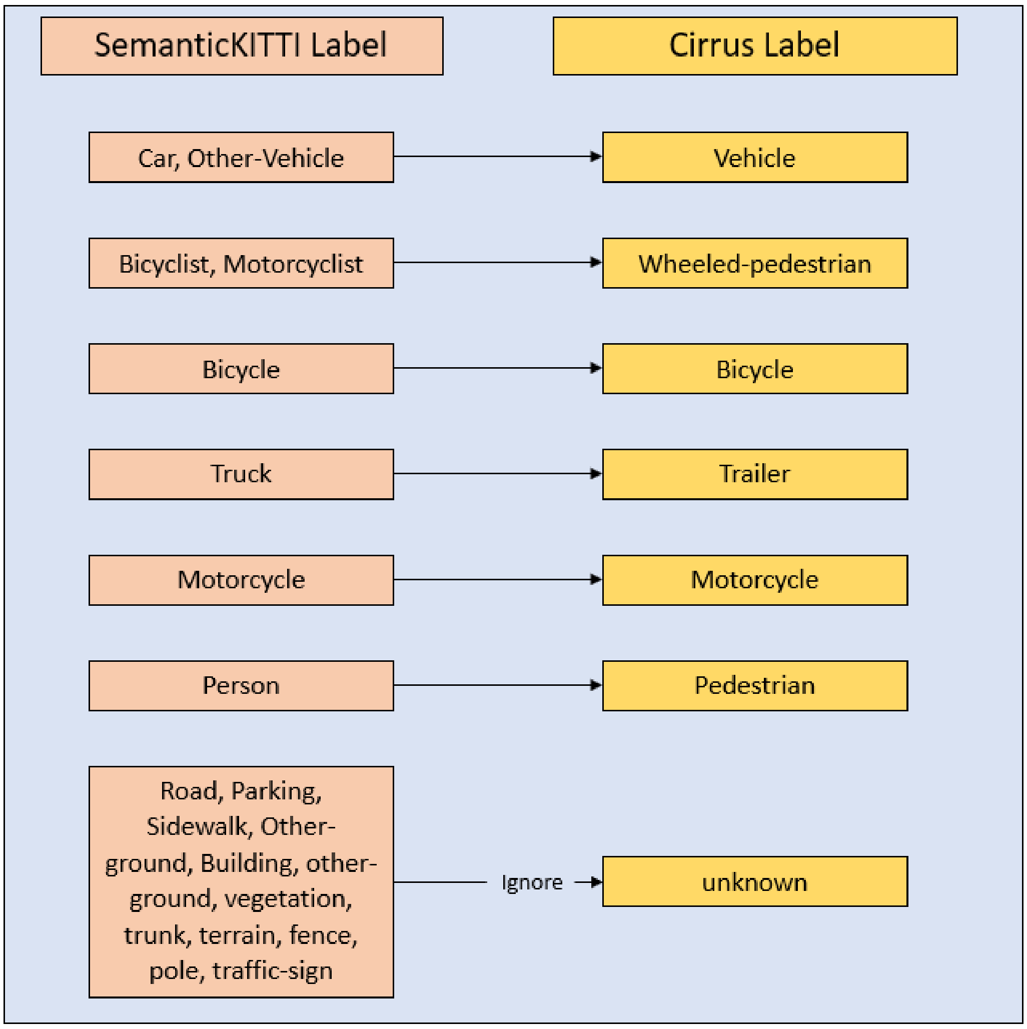


Figure 3.15: SemanticKITTI to Cirrus mapping

4 Experiments

In this chapter, we explain multiple experiments and their results using the methodologies explained in Chapter 3.

We first start by describing the Adaptation, which includes extraction of the scanning pattern of Cirrus dataset. Moreover, the result of the adaptation of SemanticKITTI and compare the data distribution of SemanticKITTI before and after adaptation.

Then, we describe different steps of SemanticKITTI densification which in turn includes multiple sub-steps such as sparse depth map generation, dense depth map generation, etc.

Further, we explain different domain mapping strategies such as Cropping, Adaptation, and Densification & Adaptation. Which are utilized for the final experimentation of semantic segmentation where all the models trained using different domain mapping strategies are evaluated.

4.1. Experiment 1: Adaptation

In this section, we will go through the process of adaptation of SemanticKITTI which in turn includes the generation of the scanning pattern of the Cirrus dataset.

4.1.1. Scanning Pattern Extraction

We can observe Cirrus's bi-pattern scanning pattern after projecting it on the image plane in Figure 4.1 and Figure 4.2. For the Gaussian pattern, the Gaussian distribution is observable along the elevation axis while the sampling rate is constant along the azimuth. Whereas in the case of a uniform pattern the sampling along the elevation and the azimuth is uniform however, the sampling frequency along the azimuth is higher than that along the elevation axis. We limit ourselves to the analysis of the Gaussian pattern only.

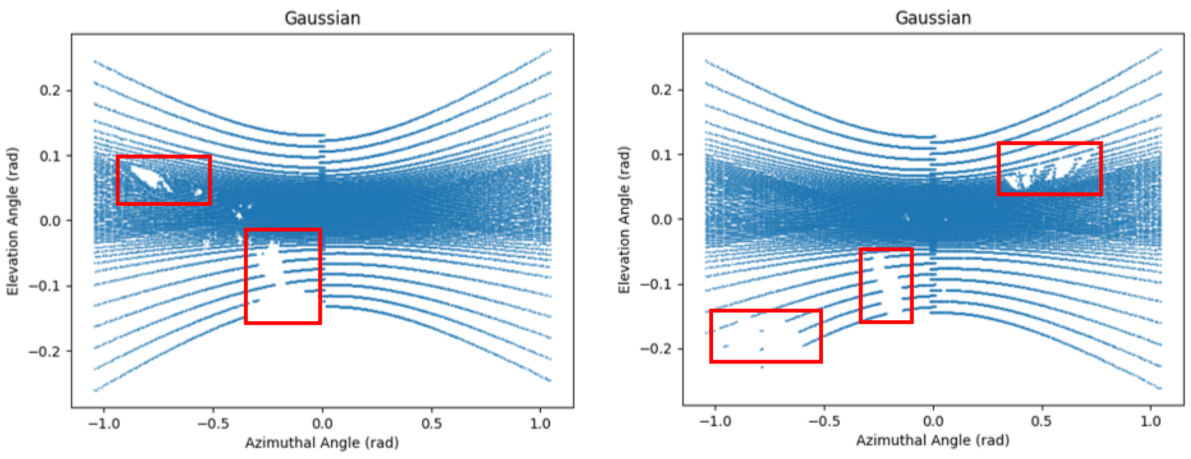


Figure 4.1: Different frames from Gaussian scanning pattern of Cirrus with gaps (Red)

Generating scanning patterns by projecting the data is difficult as it has multiple gaps which depend on the environment, occlusion, etc. It is visible that different frame has different gaps depending on the surrounding as can be seen in Figure 4.1. Therefore, to get a plausible scan pattern 2 Gaussian frames that compensate for each other's gaps were selected and superimposed as shown in Figure 4.3. Finally, Nearest neighbor (N=1) algorithm was used to generate the estimated scan pattern as shown in Figure 4.4 and also to keep the points in the scanning pattern the same as in one frame.

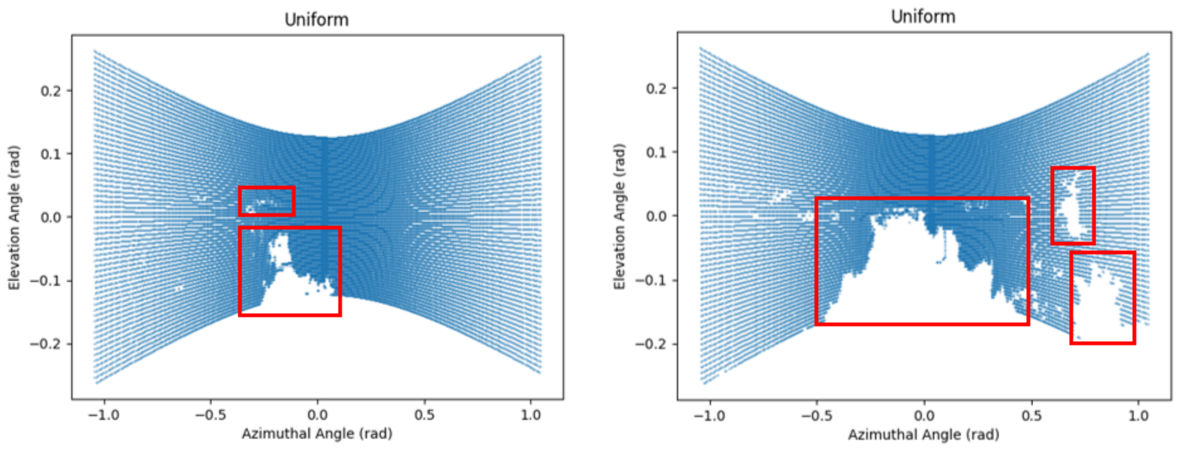


Figure 4.2: Different frames from uniform scanning pattern of Cirrus with gaps (Red)

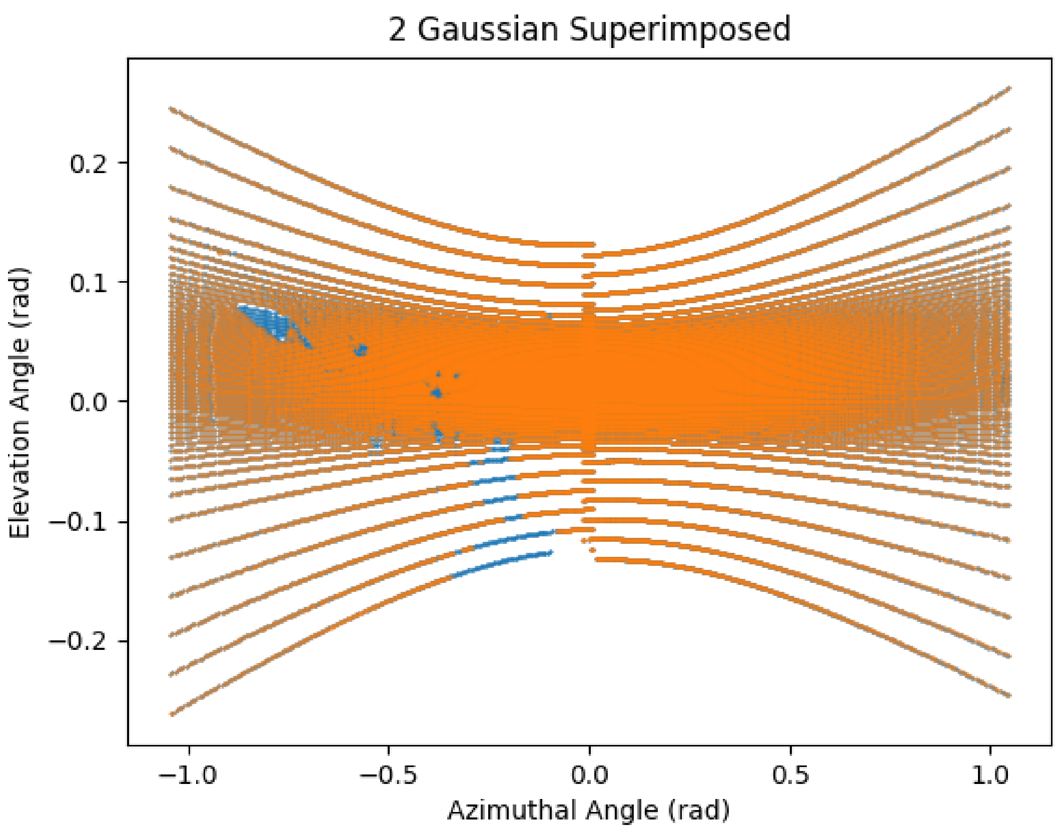


Figure 4.3: Gaussian frames superimposed, Blue: Frame 1, Orange: Frame 2

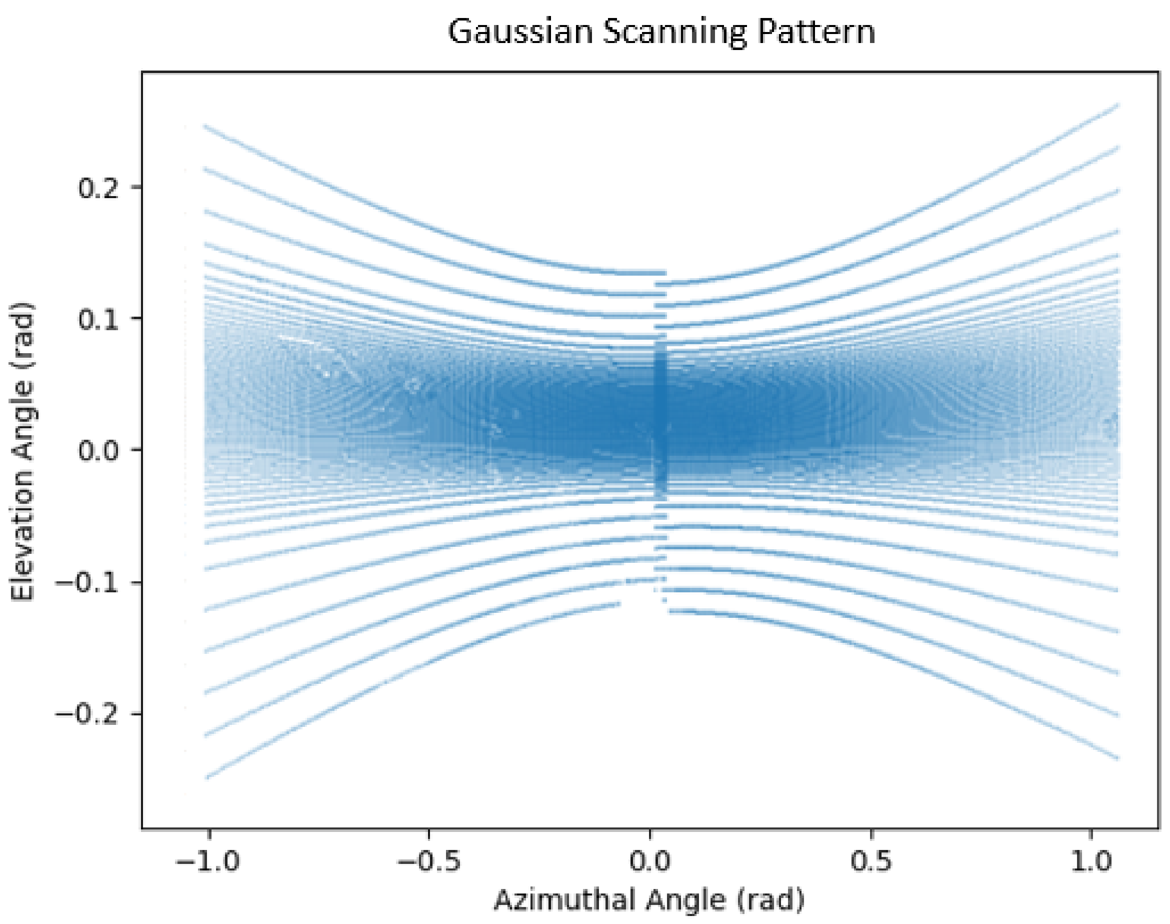


Figure 4.4: Estimated Gaussian scanning pattern

4.1.2. SemanticKITTI Adaptation

The adaption results in an extreme reduction of the point cloud from the original dataset (Approximately 92.1% average compression). The original and the adapted dataset can be seen in Figure 4.5

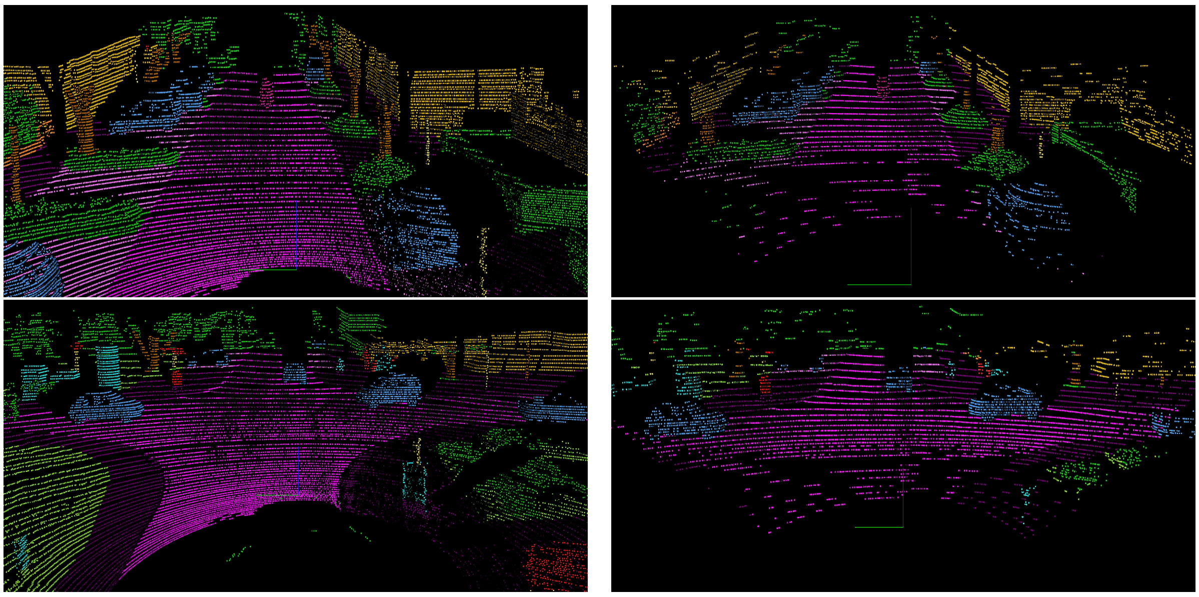
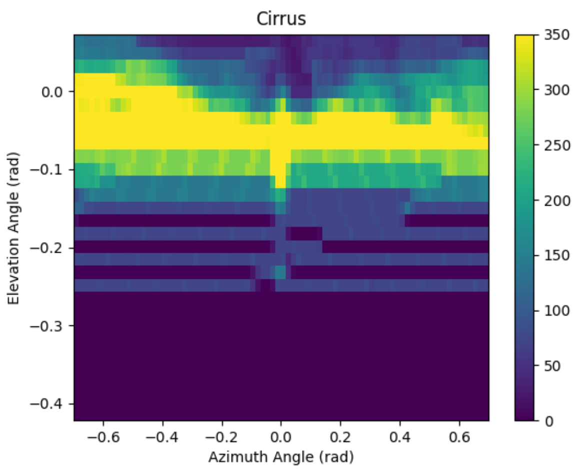


Figure 4.5: Comparison of Original SemanticKITTI and Adapted SemanticKITTI, first column shows frame of original SemanticKITTI while second column shows its respective adapted frame

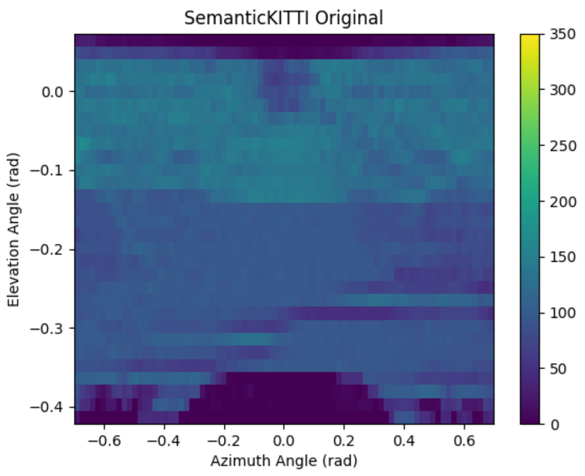
	Original	**Cropped**	**Adapted**
Average points per frame	104k	40k	8.2k
Average compression compared to original	-	61.5%	92.1%
Average compression compared to cropped	-	-	79.5%

Table 4.1: Data compression after adaptation

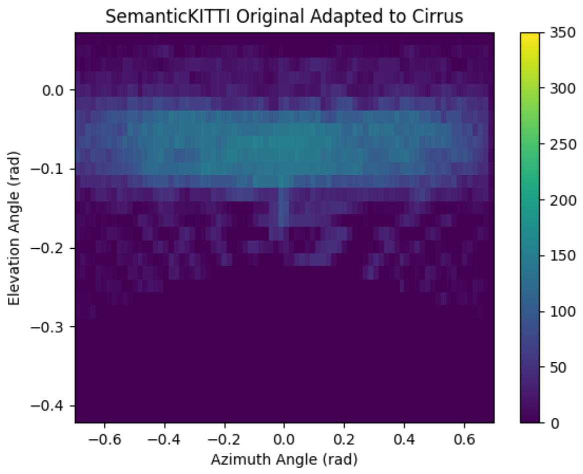
To compare how the adaptation process changed the data distribution for SemanticKITTI we project the point cloud in a plane along azimuth (Horizontal) and elevation (vertical) direction. Since data distribution of a point cloud for a frame could be highly dependent on the environment it is taken, therefore we project multiple frames on a plane and normalize it to obtain a plausible distribution. The data distribution of original SemanticKITTI and Cirrus can be seen in Figure 4.6. For SemanticKITTI the Field of View (FoV) has been limited to 120^o for a fair comparison to Cirrus as it has an FoV of 120^o. This FoV cropping results in a reduction of data available i.e. leads to data compression of about 61.5%.



(a) Cirrus



(b) SemanticKITTI



(c) Adapted SemanticKITTI

Figure 4.6: Data distribution for Cirrus, SemanticKITTI, and adapted SemanticKITTI dataset (Pixel color represents number of points populating per sq. radian angle along azimuth and elevation)

From Figure 4.6 we can see that the distribution of the Cirrus is Gaussian along the elevation whereas for SemanticKITTI it is uniform as expected. Moreover, the density of the points in the Cirrus dataset is more than in SemanticKITTI. We can observe qualitatively from Figure 4.6 that after adaptation the distribution of SemanticKITTI is close to that of Cirrus but is still much less dense. Therefore, we need to use a densification scheme to bring the distribution of SemanticKITTI even closer to Cirrus.

For sanity check of distributions generated for Cirrus and SemanticKITTI, and adapted SemanticKITTI we plot the distribution of Cirrus and SemanticKITTI and adapted SemanticKITTI from their respective different sequences which can be seen in Figure 4.7

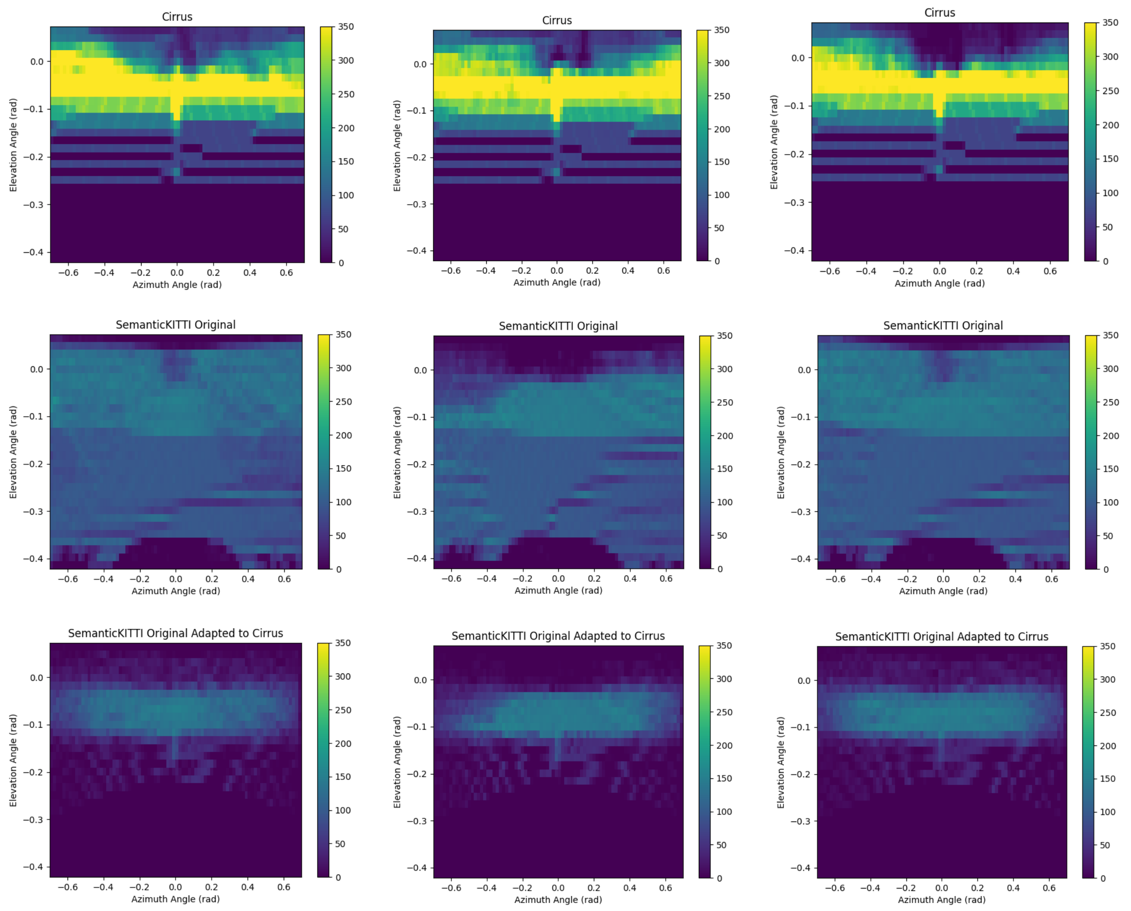


Figure 4.7: Data distribution for Cirrus (First row, Sequence 1, 2 and 3), SemanticKITTI Original (Second row, Sequence 00, 01 and 03) and Adapted SemanticKITTI Original (Third row, Sequence 00, 01 and 03)

4.2. Experiment 2: SemanticKITTI Densification

SemanticKITTI [4] which provides a point-wise label for the odometry dataset from the raw KITTI suit [24] can be adapted to that of the scanning pattern of the Cirrus pattern extracted in Section 4.1.1 however, as shown in Table 4.1 the data available for training after adaptation is highly compressed. And, as shown in Figure 4.6 the density of the points available per square radian of azimuth and elevation angle is much lower than that of Cirrus. Therefore, the upcoming sections explain the densification of the SemanticKITTI point cloud to increase the density of the point and therefore, reduce the domain gap between the adapted SemanticKITTI and Cirrus. Thus, making densified adapted SeamnticKITTI a true proxy for Cirrus, and thereby staying true to the Domain Mapping solution of Transfer Learning. The upcoming sections would describe the sequence of steps used for densifying the SemanticKITTI using the depth completion strategy described in Section 3.2. The densification steps which would be described in upcoming sections have been summarized in Figure 4.8.

Since SemanticKITTI is a subset of the KITTI suit and its densified ground truth sequences are available in the KITTI depth dataset, one could ask a question of using it directly instead of densifying the SemanticKITTI dataset. However, the KITTI depth dataset provides the ground truth for the frames at a much lower frequency than the dataset of SemantiKITTI, therefore, providing the ground truth densified data for much fewer frames. Thus, if one was to use it, the data available for training the semantic segmentation dataset would be reduced by $\approx 58\%$, therefore a densification strategy is used to densify every frame of SemanticKITTI to allow the semantic segmentation network to use a rich and abundant dataset to learn.

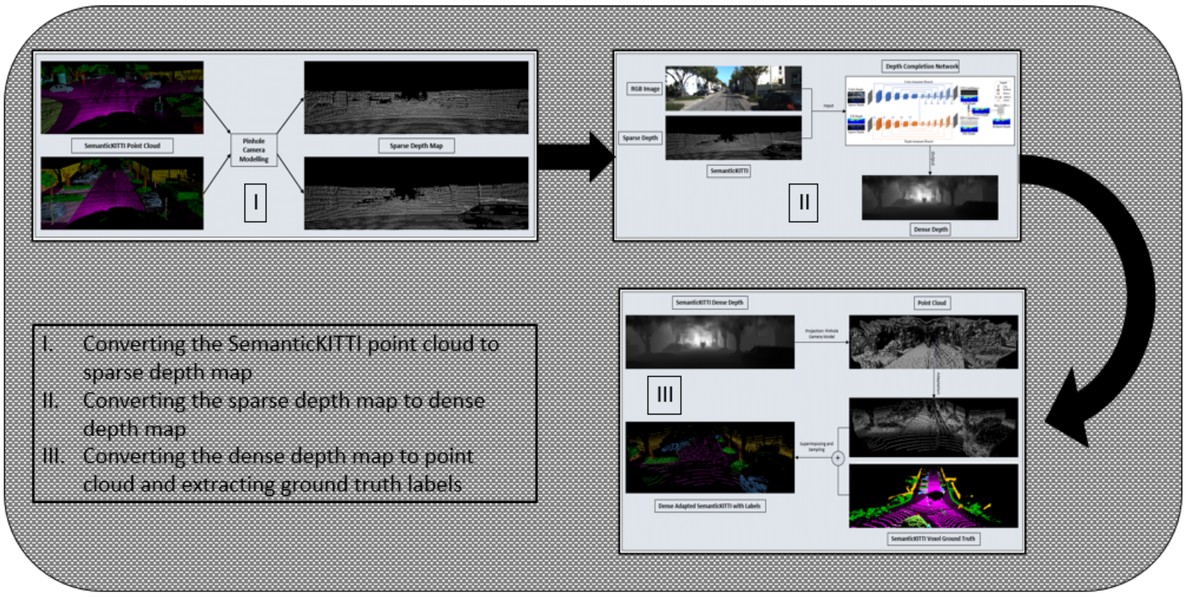

SemanticKITTI Point Cloud
Pinhole Camera Modelling
Sparse Depth Map
I
RGB Image
Sparse Depth
SemanticKITTI
Depth Completion Network
Dense Depth
II
SemanticKITTI Dense Depth
Point Cloud
III
Dense Adapted SemanticKITTI with Labels
SemanticKITTI Voxel Ground Truth
I. Converting the SemanticKITTI point cloud to sparse depth map
II. Converting the sparse depth map to dense depth map
III. Converting the dense depth map to point cloud and extracting ground truth labels

Figure 4.8: Summary of steps followed for SemanticKITTI densification

4.2.1. SemanticKITTI Sparse Depth Map Generation

SemanticKITTI dataset provides the dataset as point clouds in ".bin" files. To use it for densification we first have to convert it to a depth map for it to be used by the depth completion network. Therefore, we first use the pinhole camera methodology to convert the SemanticKITTI point cloud to the depth image. For conversion, we use the calibration matrix provided by the KITTI suit. The point cloud has been provided in Velodyne coordinate frame, therefore we first convert the point cloud from Velodyne to the left camera (KITTI setup consists of left and right camera calibration, for our scenario we use all the calibration matrix pertaining to the left camera) 3D coordinate frame. Then, we convert the point cloud from the camera's 3D coordinate frame to the camera's image coordinate frame i.e. (u, v). The resolution of the image is kept as 1216×352 because of the constraints posed by the depth completion network used in the next step of SematicKITTI densification.

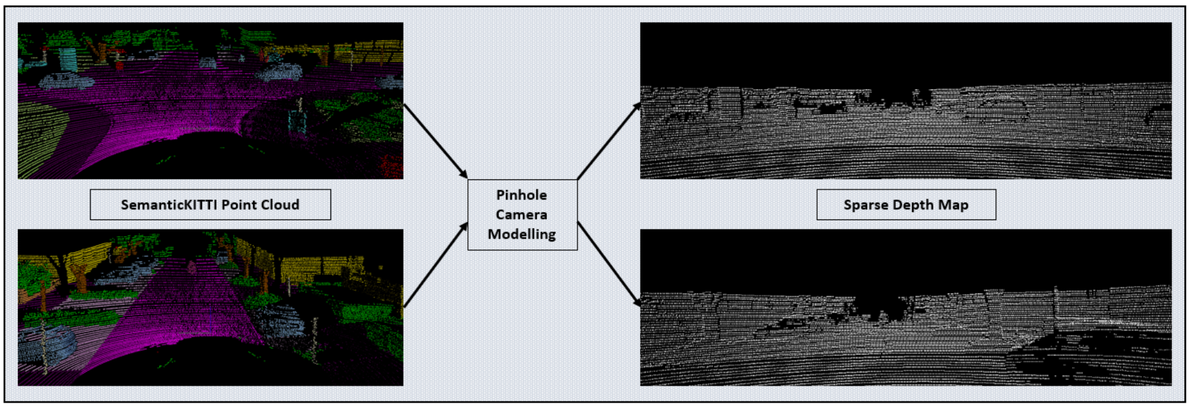


Figure 4.9: Generating sparse depth map from point cloud using pinhole camera modeling

The method is governed by principles of pinhole camera modeling. Thus, uses the Equation 3.1. However, in this case, the extrinsic and intrinsic matrix i.e. **E** and **K** is provided by the KITTI suit. The calibration matrix provided by the KITTI suit has already been rectified therefore, no further processing is required and can be used in a plug-and-play fashion. The sparse depth generated after using the pinhole camera model on SemanticKITTI can be seen in Figure 4.10.

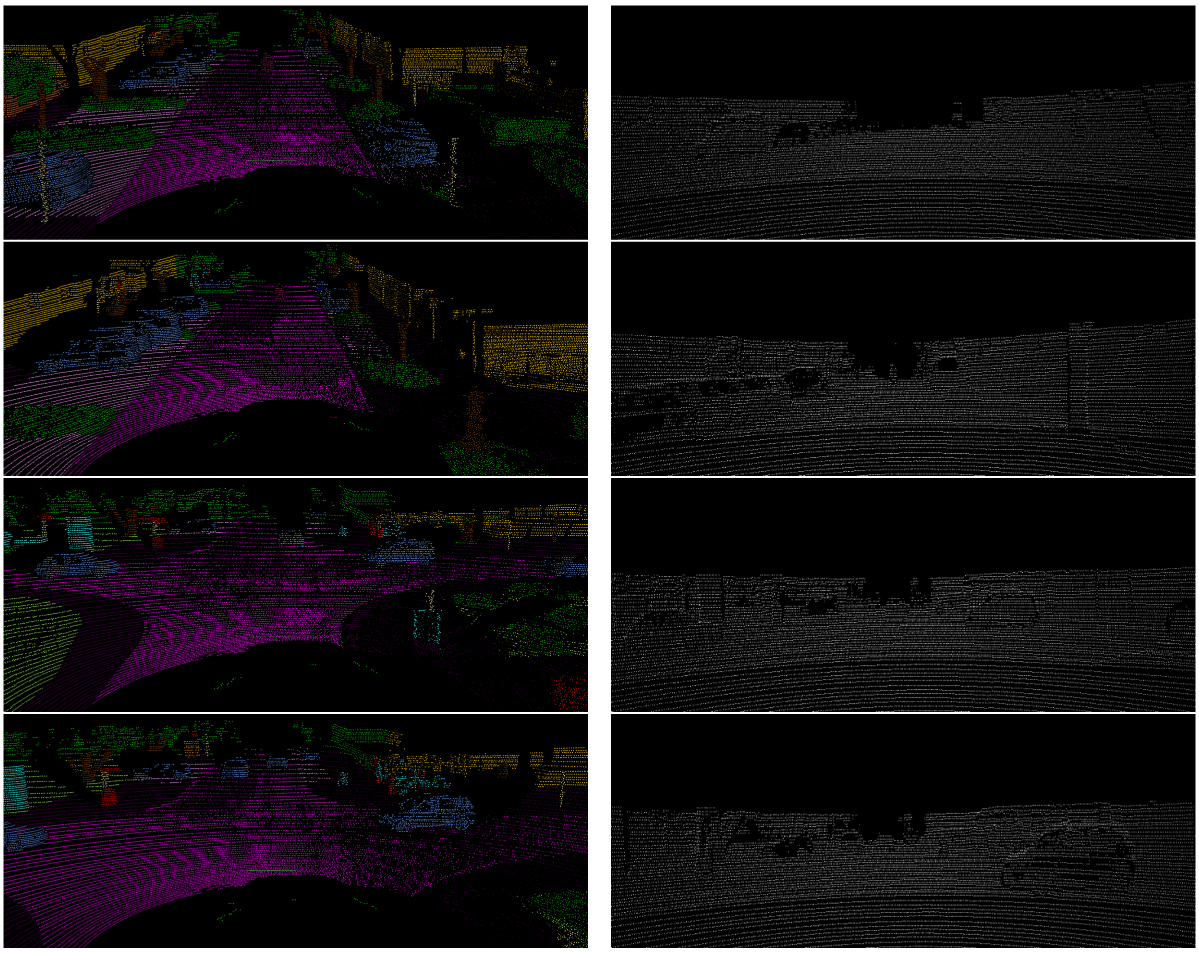


Figure 4.10: Converting SemanticKITTI point cloud into depth map, first column show the SemanticKITTI point cloud and the second column show their corresponding projected depth map

4.2.2. SematicKITTI Dense Depth Map Generation

In Section 4.2.1, we projected the SemanticKITTI point cloud into a depth image, however, the generated depth map is sparse and if adapted to Cirrus results in a very small amount of points to learn. Therefore, we use a depth completion network mentioned in Section 3.2 to densify the sparse depth map to a dense depth map. Trained depth completion architecture takes SemanticKITTI's RGB image and sparse depth map as input and outputs a densified depth map for the input frame. The input sparse depth map and their corresponding output dense depth map can be seen in Figure 4.12.

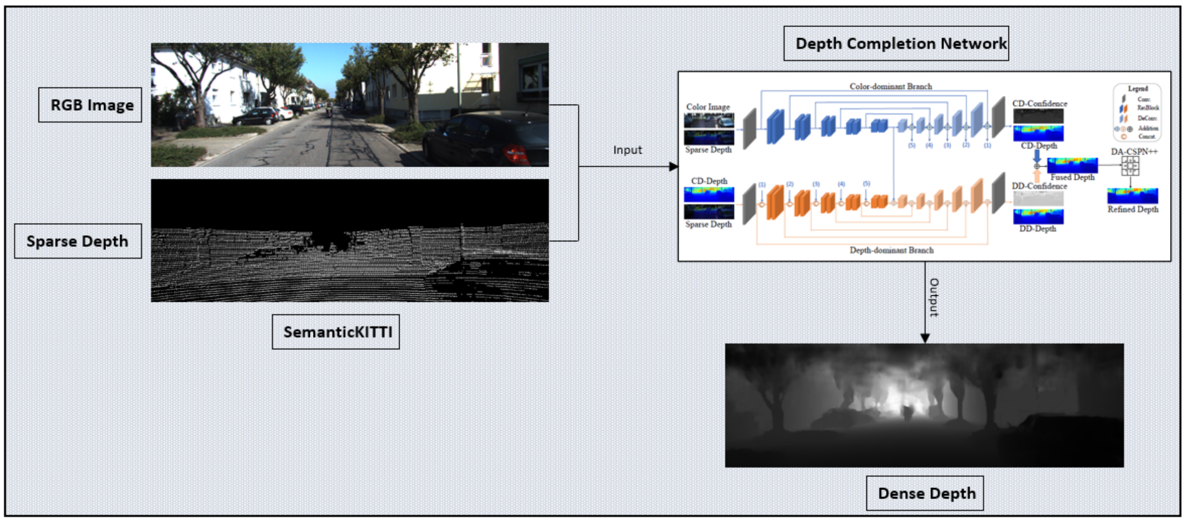


Figure 4.11: Generating dense depth map from sparse depth map using the network in Figure 3.2

The depth completion network used was trained on the KITTI depth completion dataset, however, was tested on SemanticKITTI in our scenario. A question of domain difference while training and testing could be posed, however, SemanticKITTI is a subset of the KITTI suit and is collected using the same sensor setup. Therefore, the network can be used to test on SemanticKITTI as the condition of similar data distribution between training and testing is satisfied.

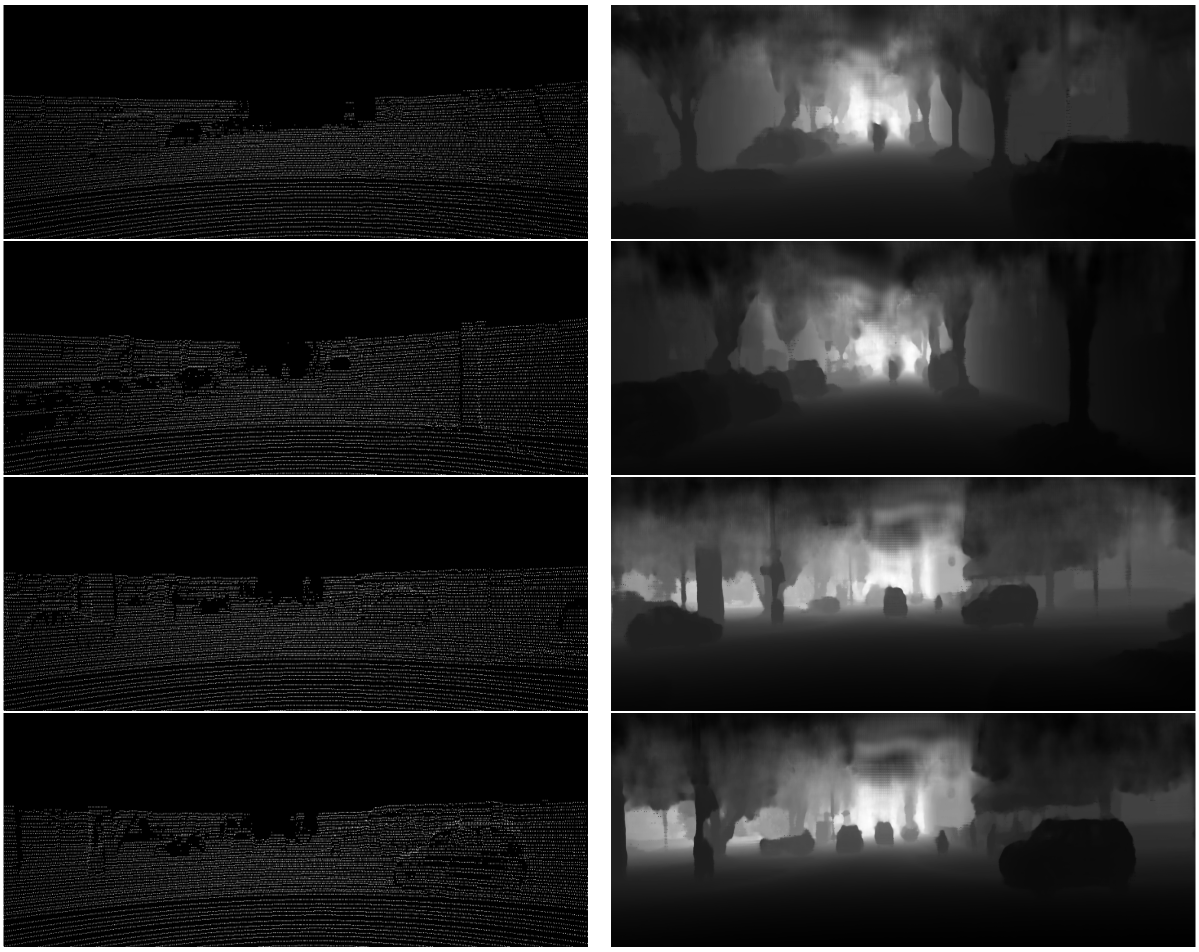


Figure 4.12: Converting SemanticKITTI sparse depth map into dense depth map using depth completion network shown in Figure 3.2, first column show the SemanticKITTI sparse depth map and second column show their corresponding predicted dense depth map

4.2.3. Label Generation

In the previous step, we generated a dense depth map for the SemanticKITTI frame, however, to train a semantic segmentation network we need to convert the dense depth map to a point cloud, and since we use a supervised learning method we would need point-wise labels. The depth completion network outputs a depth image for which we need to create semantic labels. One method could be using a human-based approach to label the densified point cloud. However, we use a different approach which depends on the voxel ground truth labels of SemanticKITTI.

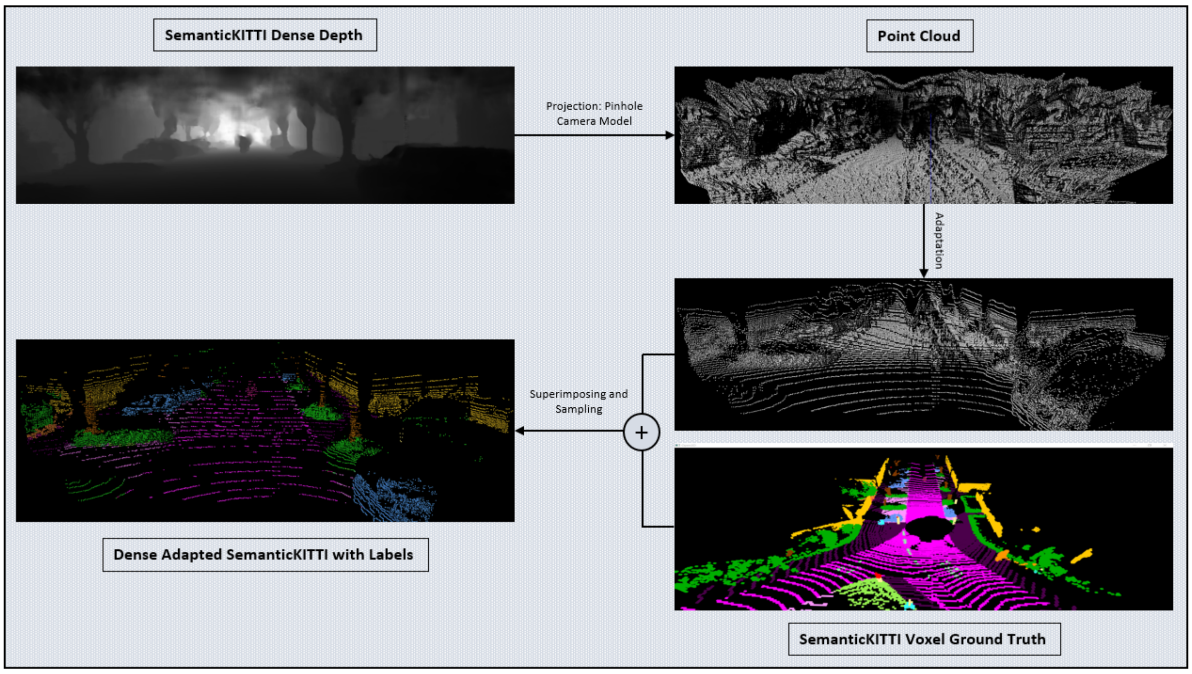

SemanticKITTI Dense Depth
Point Cloud
Projection: Pinhole Camera Model
Adaptation
Superimposing and Sampling
Dense Adapted SemanticKITTI with Labels
SemanticKITTI Voxel Ground Truth

Figure 4.13: Pipeline for generating labels for SemanticKITTI dense depth map created

We project the dense depth image to point clouds using the inverse pinhole camera model, however, after densifying the points in the frame lose their semantic meaning, Therefore, we superimpose the dense point cloud onto the space with voxels having the semantic labels from SemanticKITTI ground truth for each frame. Thus, giving the label for points lying in a voxel, the same as that of the voxel, and removing points that do not lie in a voxel. The superimposition of points and voxels has been shown in Figure 4.14

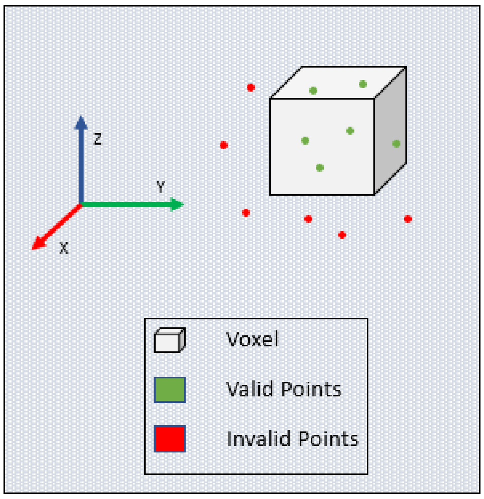


Figure 4.14: Superimposition of dense point cloud over voxels having ground truth semantic labels (Size not to scale)

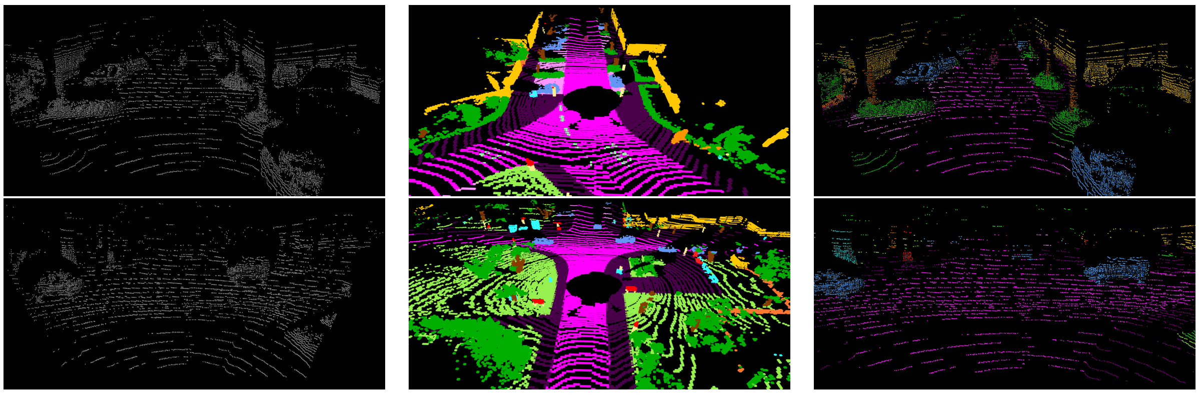


Figure 4.15: An example of Adapted dense SemanticKITTI with their semantic labels, the first column shows the adapted dense SemnaticKITTI, the second column shows the voxel ground truth for the respective frame and the third column shows the semantic labels generated for the adapted dense SemanticKITTTI frames

The adapted SemanticKITTI frames and their respective semantic labels generated are shown in Figure 4.15.

4.2.4. SemanticKITTI and Cirrus distribution Comparison after Densification

In Section 4.1 we saw that the distribution of the SemanticKITTI after adaptation is similar to that of Cirrus as shown in Figure 4.6. However, the density of the points available per square radian for adapted SemanticKITTI and Cirrus was very different with adapted SemanticKITTI being much less dense. Therefore, we adopted a point cloud densification scheme mentioned in Section 3.2.

The data distribution of the Densified SemanticKITTI can be seen in Figure 4.17. And again for a sanity check, we also check the data distribution of multiple sequences of Densified Adapted SemanticKITTI which can be seen in Figure 4.16

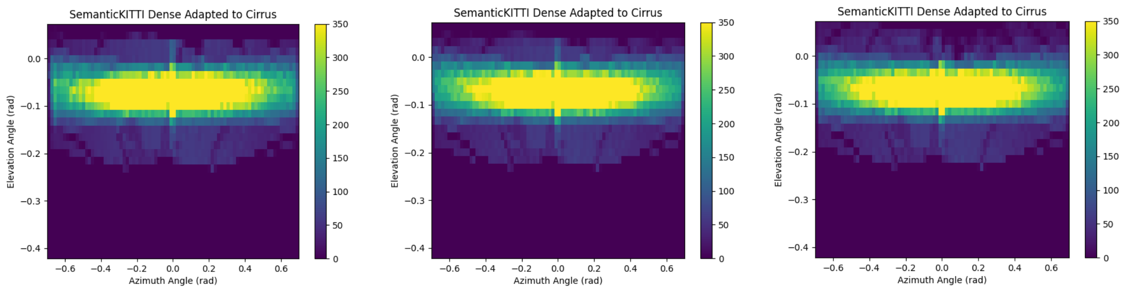


Figure 4.16: Data distribution for Sequences 00, 01 and 03 of Densified Adapted SemanticKITTI

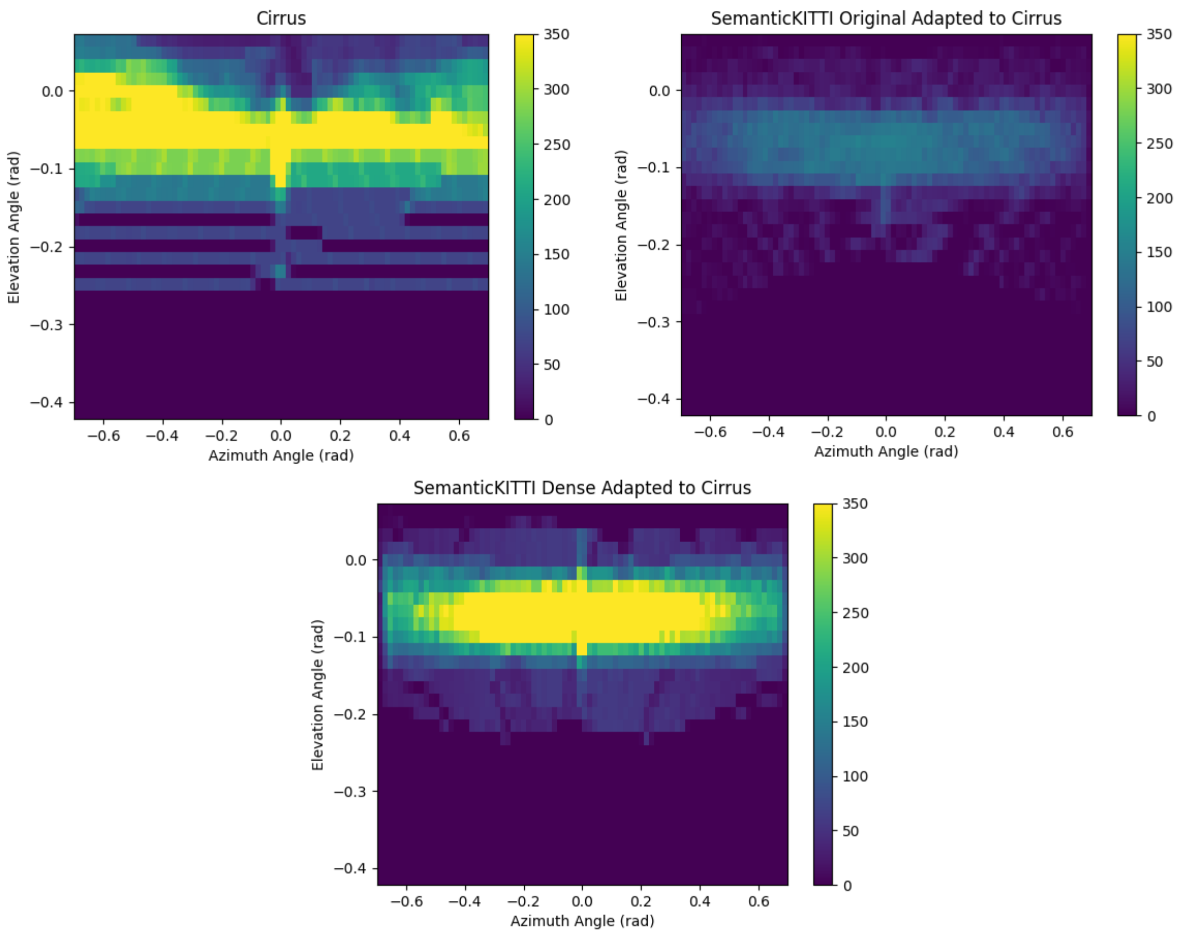


Figure 4.17: Comparison of data distribution of Cirrus, Adapted SemanticKITTI, and Dernsified Adapted SemanticKITTI

	Cirrus	**SemanticKITTI**	**Cropped SemanticKITTI**
Average points per frame	53k	104k	40k
Ratio of Cirrus to Dataset points/ frame	1	0.51	1.32

	Adapted SemanticKITTI	**Densified Adapted SemanticKITTI**
Average points per frame	8.2k	38.1k
Ratio of Cirrus to Dataset points/ frame	6.46	1.39

Table 4.2: Comparison of different customized dataset points to Cirrus dataset

From Table 4.2 we can observe that ratio of points of Cirrus to Densified Adapted SemanticKITTI in a frame is the second closest to 1 with a value of 1.39 after cropped

SemanticKITTI. However, the point distribution of Densified Adapted KITTI is close to that of the Cirrus dataset as can be seen in Figure 4.17. We can also observe that in comparison to Adapted SemanticKITTI the Densified Adapted SemanticKITTI dataset provides about 29.9k extra points per frame to learn from for the semantic segmentation model.

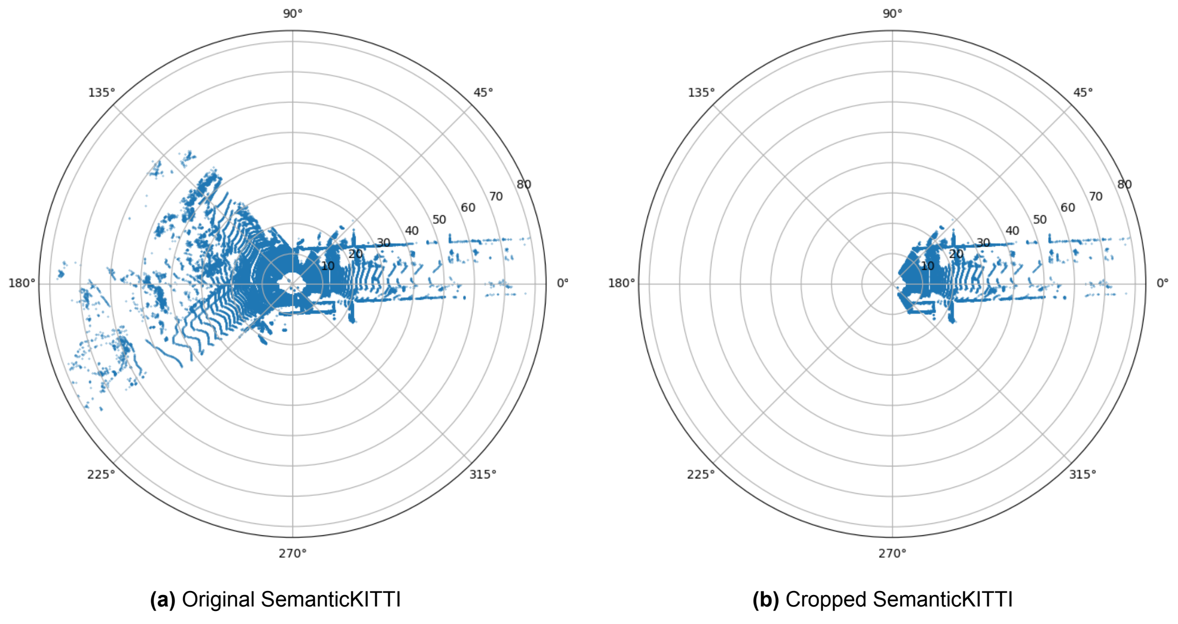
4.3. Experiment 3: Domain Mapping Strategies

SemanticKITTI has an abundant amount of annotated data and can be used to train a network in supervised methodology. However, since its characteristics differ significantly from the target dataset, therefore we need to make some adjustments to proceed. Therefore, different domain mapping strategies have been adopted to reduce the domain gap.

4.3.1. Domain Mapping Strategy 1: Data Cropping

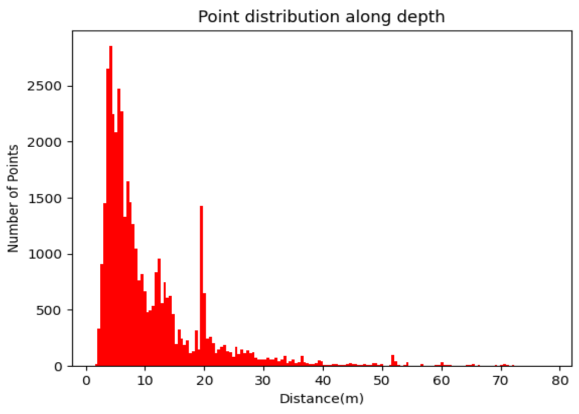
The semanticKITTI dataset was originally collected using Velodyne HDL-64E which has a 360^o Field of View along the horizontal axis (Azimuth). Whereas, Cirrus has a 120^o as FoV along the azimuth. Therefore, to match the source frame ratio to that of the target frame ratio we apply an angular filter to each frame of SemanticKITTI filtering the points lying outside the 120^o FoV along the azimuth. The filtration process is done by converting the frame of SemanticKITTI into polar coordinates and then removing the points lying outside range $[-60^o, 60^o]$. The frame before and after the filter is applied can be seen in Figure 4.18.

After angular filter, the horizontal FoV for semanticKITTI becomes the same as that of Cirrus. The analysis of the frame of cropped SemanticKITTI can be seen in Figure 4.19

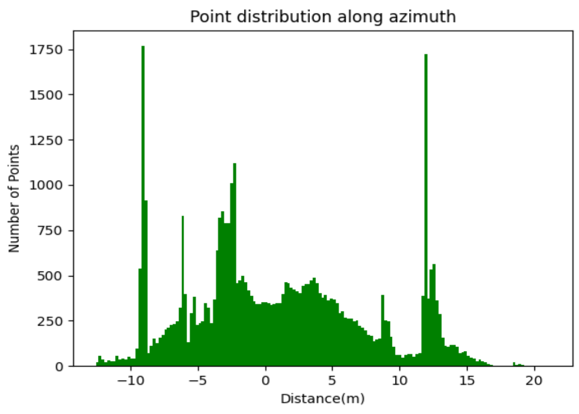


(a) Original SemanticKITTI **(b)** Cropped SemanticKITTI

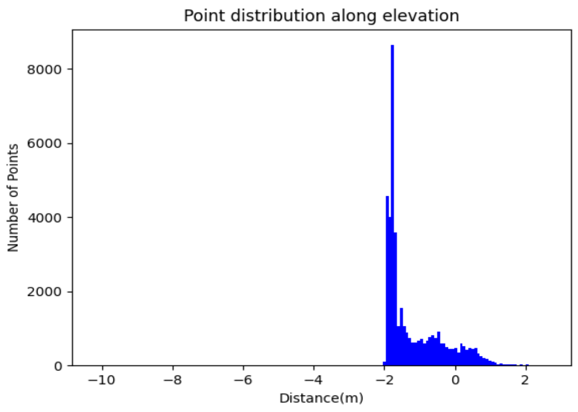
Figure 4.18: Bird-eye view of original and cropped SemanticKITTI



(a) X-axis: Depth



(b) Y-axis: Azimuth



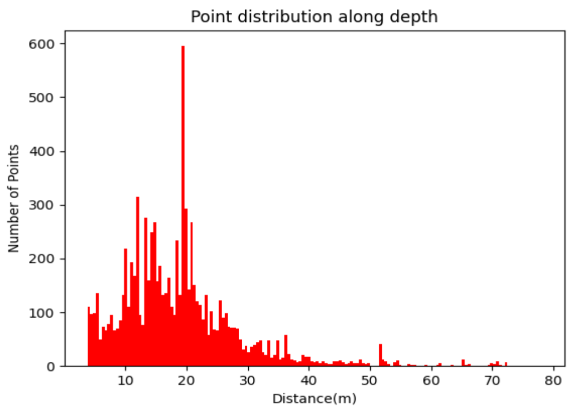
(c) Z-axis: Elevation

Figure 4.19: Data distribution of Cropped SemanticKITTI along different axis

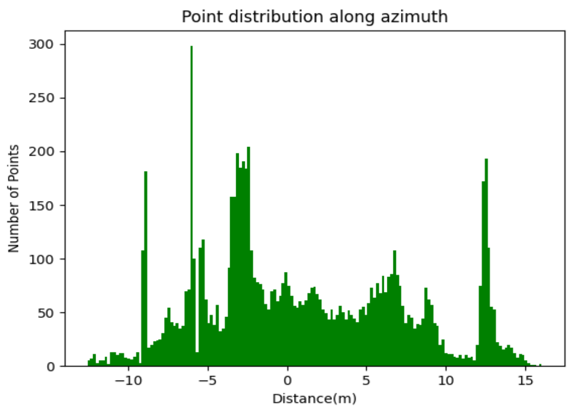
4.3.2. Domain Mapping Strategy 2: Data Adaptation

SemanticKITTI which is collected using a Motorized Optomechanical LiDAR has a uniform scanning pattern however, the scanning pattern of Cirrus is Gaussian along elevation as shown in Figure 4.4. Therefore, we use a sampling strategy mentioned in Section 3.1 on Original SemanticKITTI as a part of domain mapping to map SemanticKITTI (Source) distribution to Cirrus distribution.

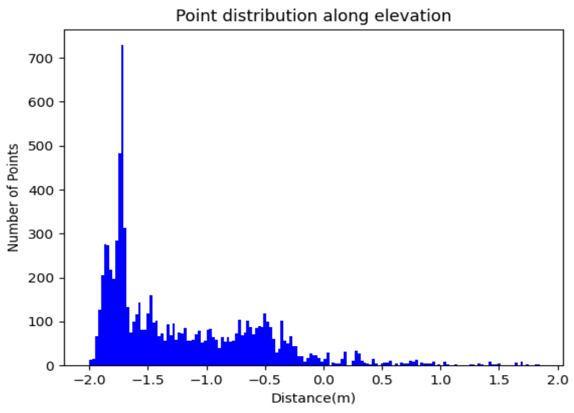
The analysis of point distribution after adaptation can be seen in Figure 4.20.



(a) X-axis: Depth



(b) Y-axis: Azimuth



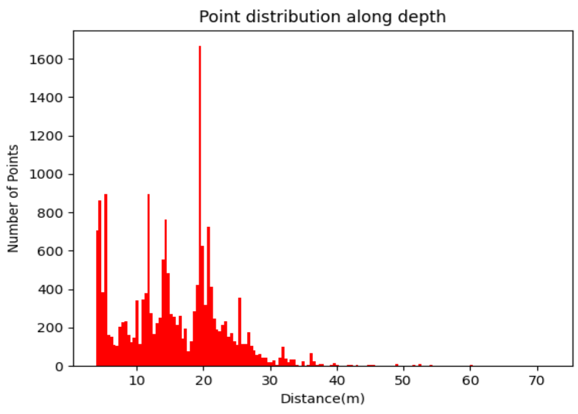
(c) Z-axis: Elevation

Figure 4.20: Data distribution of Adapted SemanticKITTI along different axis

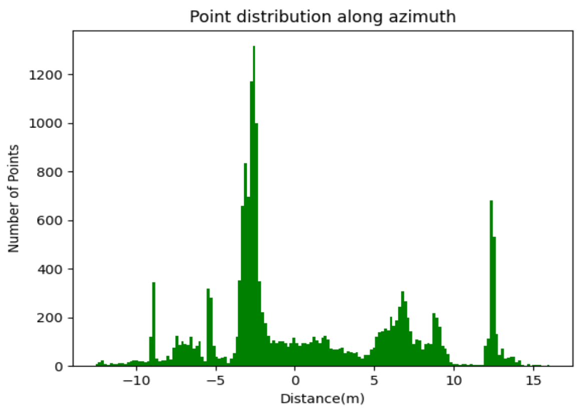
4.3.3. Domain Mapping Strategy 3: Data Densification and Adaptation

Finally, the domain mapping strategy used is densifying the point cloud and then adapting to the Cirrus using the same adaptation strategy mentioned previously. The advantage of using this methodology as mentioned in Section 3.2 is the availability of more data points to learn for the semantic segmentation network.

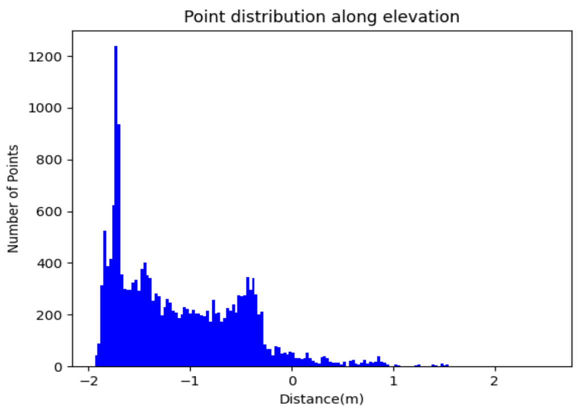
The analysis of point distribution after adaptation can be seen in Figure 4.21.



(a) X-axis: Depth



(b) Y-axis: Azimuth



(c) Z-axis: Elevation

Figure 4.21: Data distribution of Densified Adapted SemanticKITTI along different axis

4.4. Experiment 4: Semantic Segmentation

The training of the neural network presented in Section 3.3.2 was done on a LINUX-based server. The configurations of the system have been summarised in Table 4.3

Configurations	
Operating System	Ubuntu 16.04
CPU	Intel 4.80GHz, 8 Cores
GPU	Nvidia 1080Ti x 2
Python	v3.7
PyTorch	v1.7.1
CUDA Toolkit	v10.2
spconv	v1.2.1

Table 4.3: Configurations used for training semantic segmentation

To evaluate the effectiveness of the domain mapping we trained multiple models namely, Baseline SemanticKITTI, Cropped SemanticKITTI, Adapted SemanticKITTI, and Densified Adapted SemanticKITTI. For training and validation of each of the above models have been provided with differently pre-processed data according to varying domain mapping strategies which have been summarized in Table 4.5.

Each of the models was trained with the same hyperparameters to perform a fair comparison. Training of models was done on a remote in-house server comprising NVIDIA GPUs.

4.4.1. Hyperparameters

While training the model described in Section 3.3.2, multiple hyperparameters are set which govern the training of the model. These hyperparameters would be described in this section below. The hyperparameters have also been summarised in Table 4.4.

Learning Rate and Batch size

Learning rate governs the improvement of the error gradient generated while using the Gradient Descent approach [3]. Improvement while training is sensitive to learning rate, if it is too high it poses a risk of error explosion or if it is too low it might cause a large number of iterations to learn or might cause the optimization to get stuck in local minima. Since it is generally not possible to fit complete training data in the memory while training thus, a mini-batch of the data is used.

In this thesis, we used a constant learning rate of 0.001 and a mini-batch size of 4 frames. Increasing the mini-batch size reduces the training time, however, it is bottlenecked by the available GPU memory.

Grid Shape

Grid shape hyperparameter defines the size of voxel discretization of the 3D space of the point cloud. Since the architecture uses cylindrical voxelization, the grid defines the voxel in cylindrical coordinates. The grid shape used provides the resolution of the voxels, which if provided too fine results in high computational cost while if it is too coarse could lead to large discretization loss.

In this thesis, we used a cylindrical grid shape defining radius, azimuth, and height for which 480, 120 and 32 values were used respectively. As described in Section 3.3.2 the volume of the voxels increases along the radius dimension.

Input Feature Dimension

Section 3.3.2 describes that network is provided with certain features for each point consisting of voxel-based and point-based information. This hyperparameter decides the input feature size.

For the given network nine features are associated with every point. The feature vector of a point comprises Cartesian coordinates, Cylinder coordinates, and voxel-grid index.

Hyperparameters	Used Values
Grid Size	[480, 120, 32]
Feature Dimension	9
Number of classes	19
Batch Size	4
Max epochs	40
Learning Rate	0.001
3D Conv. Stride	2
Optimizer	Adam

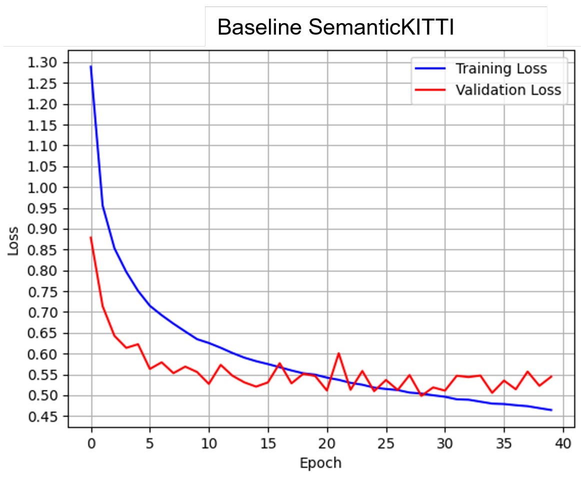
Table 4.4: Summary of hyperparameters for semantic segmentation

Model Name	Data Pre-processing		
	Cropped (Field of View Reduced)	**Adapted**	**Densified**
Baseline SemanticKITTI	NO	NO	NO
Cropped SemanticKITTI	YES	NO	NO
Adapted SemanticKITTI	YES	YES	NO
Densified Adapted SemanticKITTI	YES	YES	YES

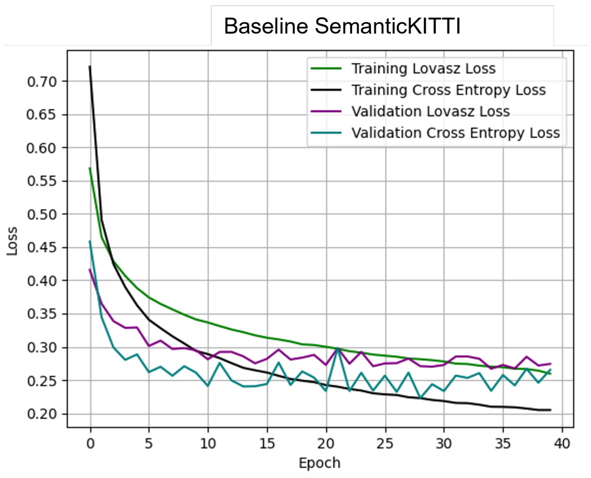
Table 4.5: Input data pre-processing for models

The model architecture was designed on PyTorch and uses CUDA for quick, parallel, and efficient computation. Initially, it was trained on a single NVIDIA $1080Ti$ GPU equipped with 3584 CUDA cores. It took an average of ≈ 77 hours of computational time to train a model on a single GPU. However, since multiple GPUs are available, the computational time could be reduced further by using multiple GPUs by adopting Distributed Data Parallel (DDP) training.

Distributed Data Parallel approach works by loading a copy of the model on different GPUs, each GPU action is controlled by processes on the CPU. Therefore, multiple processes could be parallelized with multiple CPU cores. Each of the processes communicates with each other and performs a similar task. During the process of training, a different mini-batch is loaded to each of the GPUs from the data. Each GPU acts independently of the other and performs forward pass, backward pass, and gradient calculation. The calculated gradients from different GPUs are accumulated and averaged, this averaged gradient is then sent to all the GPUs for the learning parameters correction to keep the model weights synchronized across all GPUs. Since the learning from multiple mini-batches is done in a single iteration thus, the training duration is reduced tremendously.

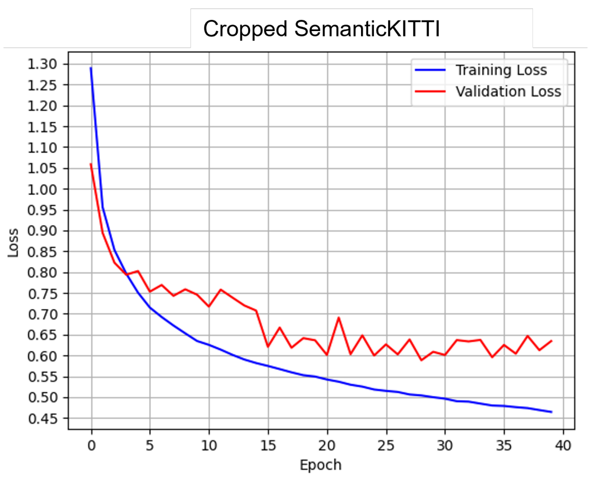


(a) Complete training and validation loss

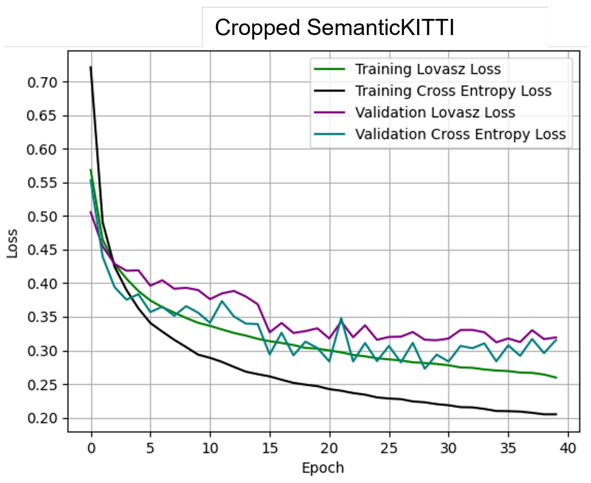


(b) Cross Entropy and Lovasz loss

Figure 4.22: Training and validation curve for Baseline SemanticKITTI

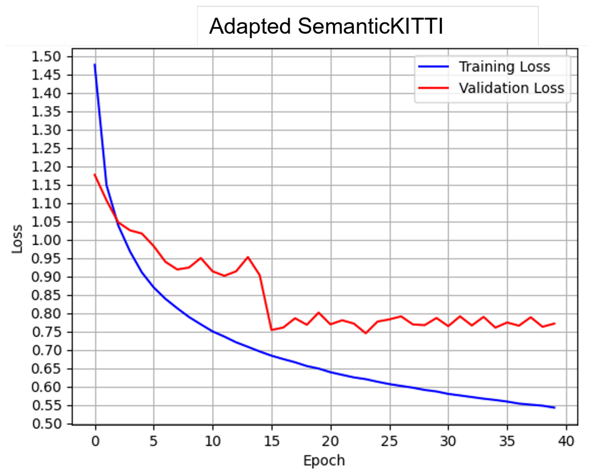


(a) Complete training and validation loss

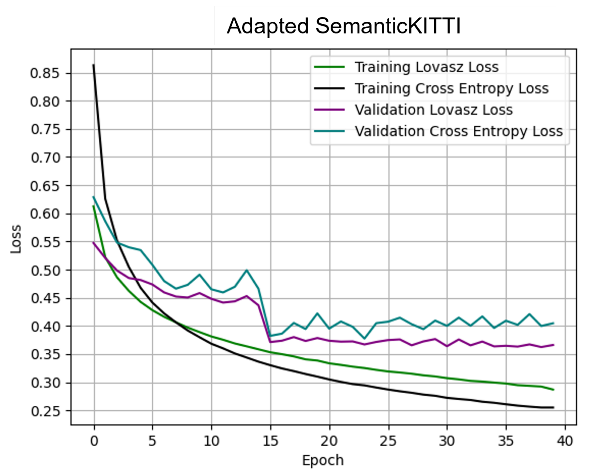


(b) Cross Entropy and Lovasz loss

Figure 4.23: Training and validation curve for Cropped SemanticKITTI

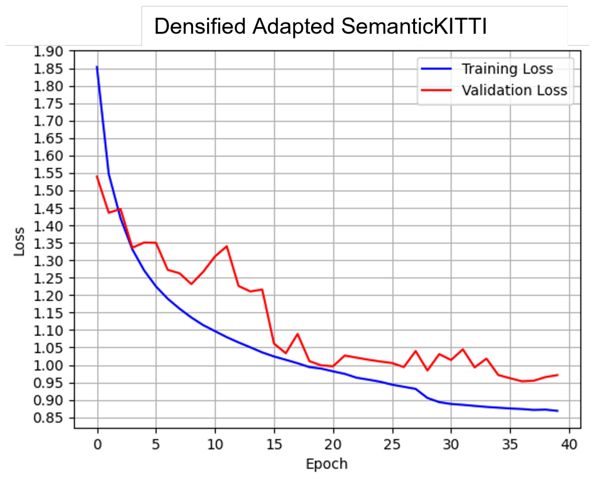


(a) Complete training and validation loss

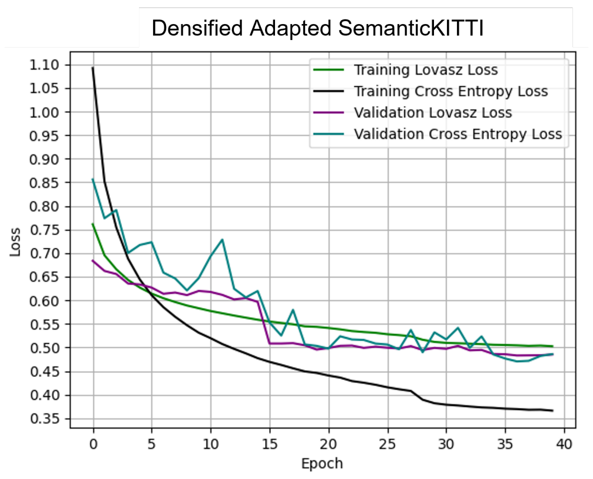


(b) Cross Entropy and Lovasz loss

Figure 4.24: Training and validation curve for Adapted SemanticKITTI



(a) Complete training and validation loss



(b) Cross Entropy and Lovasz loss

Figure 4.25: Training and validation curve for Densified Adapted SemanticKITTI

Figure 4.22, 4.23, 4.24 and 4.25 show the training and validation curves for Baseline SemanticKITTI, Cropped SemanticKITTI, Adapted SmanticKITTI and Densified Adapted SemanticKITTI respectively. All the models show that they learn from their respective input data. Baseline SemanticKITTI and Densifed SemanticKITTI show the best convergence of training and validation curves showing the least overfitting followed by Cropped SemanticKITTI and Adapted SemanticKITTI.

4.4.1.1. Results and Analysis

In the domain of semantic segmentation, a common metric for evaluation of results is Intersection over Union (IoU) also referred to as Jaccard index [28]. We will evaluate the performance of our trained model using IoU metric. IoU measures the ratio of true positive prediction given by the model over the population of ground-truth and prediction made. The mathematical representation of the metric can be seen in Equation 4.1. IoU can also be understood pictorially from Figure 4.26

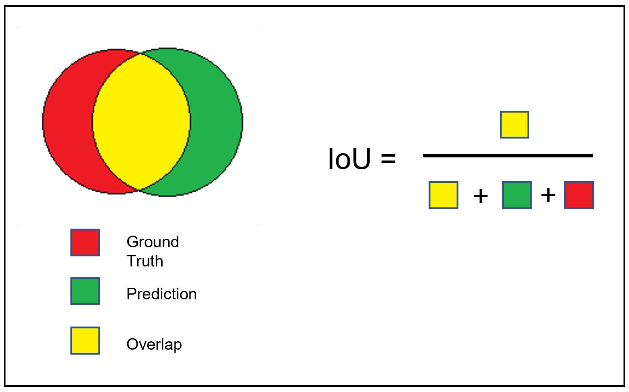


Figure 4.26: IoU pictorial representation

In the scenario of multiple classes, the performance of the model is calculated by calculating IoU for all the classes individually and then averaging them giving the mean Intersection over Union (mIoU) which mathematically is represented in Equation 4.2

$$IoU = \frac{TP_c}{TP_c + TN_c + FN_c} \tag{4.1}$$

$$mIoU = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + TN_c + FN_c} \tag{4.2}$$

Where, TP_c is correctly classified points for class c, TN_c are the points predicted other than c and also belong to a class other than c and FN_c are the points which belong to class c but are wrongly predicted.

Section 3.3.1.2, describes that SemanticKITTI consists of moving classes sub-classes for vehicular classes. Since the model does not train on temporal information of the point clouds, i.e. each point cloud prediction is independent of the previous predictions or any other modal information like Doppler, etc. Therefore, the model is incapable of differentiating between moving and non-moving classes which also limits its capability to predict such information on the Cirrus dataset.

This will provide the results of the multiple trained models and will emphasize them quantitatively and qualitatively. The quantitative section will emphasize the results of the models on the classes present in Cirrus and the qualitative section will pay more attention to classes which were predicted by the model but have no representation in Cirrus, like road, vegetation, etc.

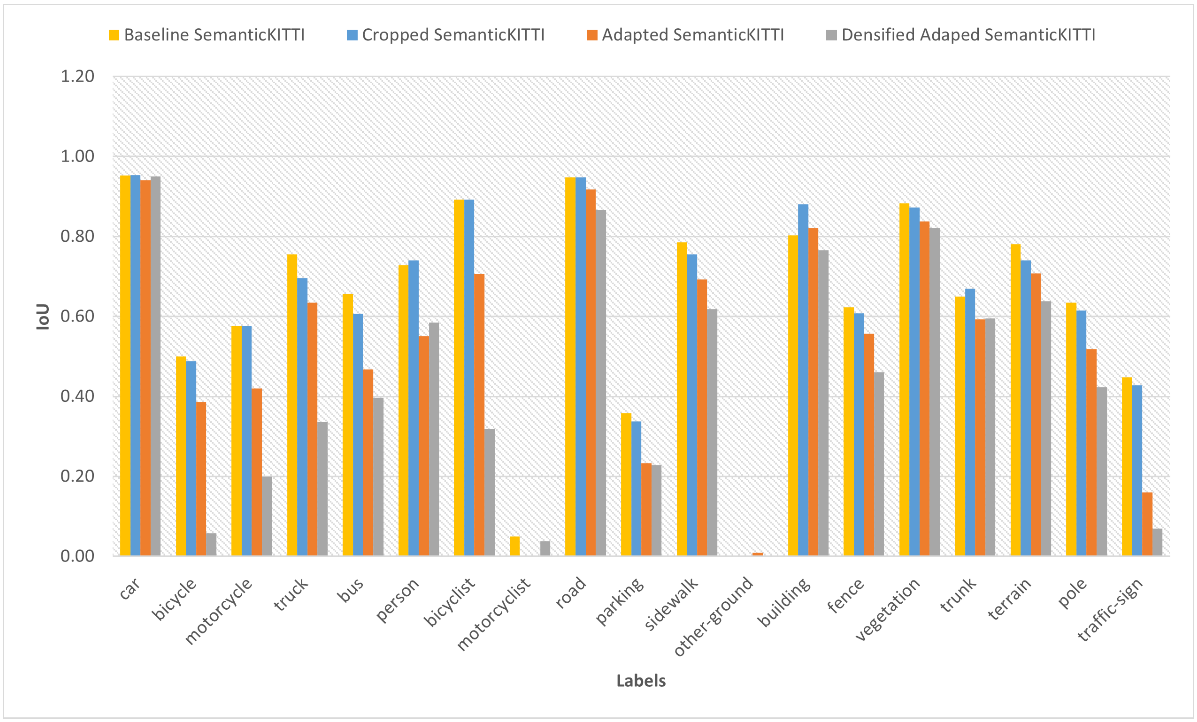


Figure 4.27: IoU comparison of trained models on SemanticKITTI validation

Quantitative

The performance of the models can be seen on the validation and the target data separately. Therefore, we first pay attention to the performance on the validation data by all the models which has been shown in Figure 4.27. It can be observed that all models perform well on the classes which provide a large amount of data to learn from such as road, car, building, etc. While the performance drops for classes like bicycle, motorcyclist, etc which have minor representation in training data. Moreover, The figure shows that for the majority of classes the performance (mIoU) of the Baseline, Cropped, Adapted, and Densified Adapted model decreases 0.63, 0.62, 0.53, and 0.44 respectively. The decrease in performance is quite drastic for classes such as bicycle, motorcycle, traffic-sign, etc while for classes such as road, car, etc it is negligible. The performance drop could be a result of high validation loss at the conclusion of training which shows that the hyperparameters selected need further tuning independently for different models. A significant performance drop can be observed in classes with smaller data representation in the validation dataset while for the classes with a majority representation of data the performance drop is minimalistic even with no hyperparameter tuning.

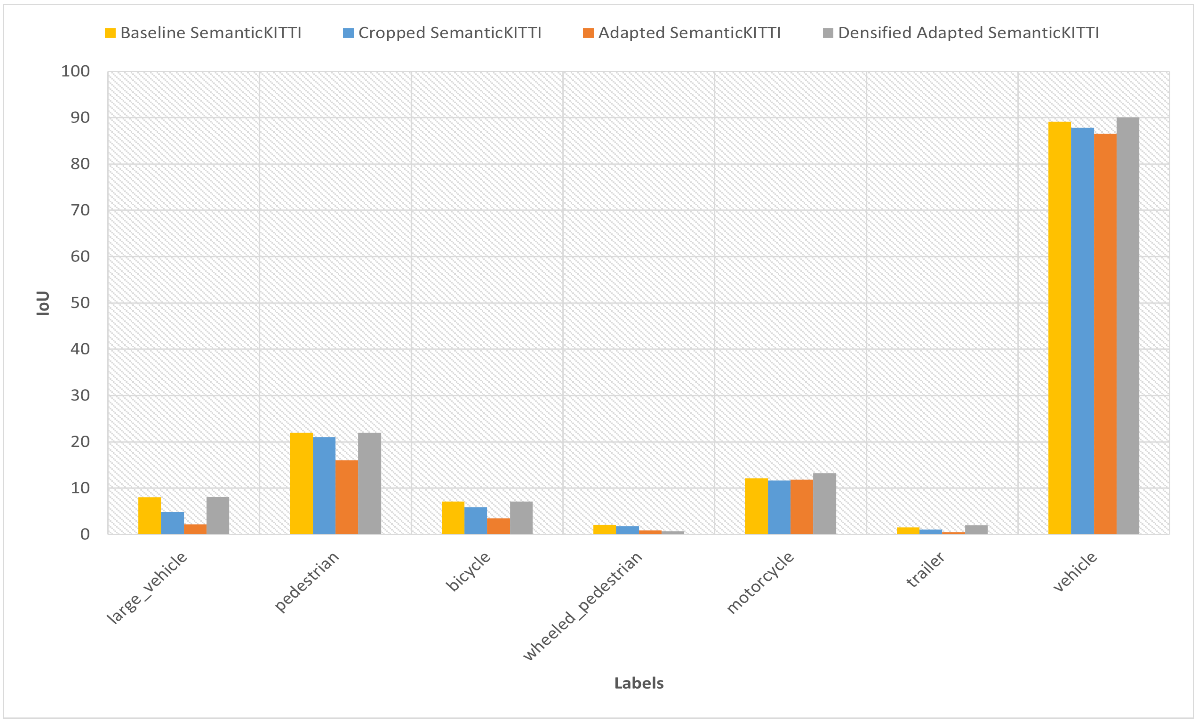


Figure 4.28: IoU comparison of trained models on Cirrus

Now, we will analyze the performance of all the models on the target dataset i.e. Cirrus dataset after label mapping which was discussed in Section 3.3. The performance trend of the models have changed with Densified Adapted model showing the best performance overall. The mIoU for Baseline, Cropped, Adapted and Densified Adapted models are 0.2027, 0.1918, 0.1734, and 0.2045 respectively. There is an insignificant difference in the performance of the models if judged upon mIoU however, the performance is extremely different ranging from $\approx 89\%$ on vehicle class to $\approx 1\%$ on trailer class. This highly skewed performance could be because of the data representation in the target dataset which has been shown in Table 4.6. It is evident that vehicle class forms a predominantly higher percentage of target data comprising over 85% of the target dataset. Therefore, a misprediction on one vehicle class point would reduce the performance on vehicle class only by a fraction while a misprediction of a trailer class point would result in a much higher performance drop of the model on trailer class and similarly for other classes. Therefore, the performance of models on vehicle class provides a better comparison. It is evident that Densified Adapted model performs better than Baseline with better mIoU even with only a fraction of training data in comparison to the Baseline, thus showing a positive transfer learning.

Sequence	Large-vehicle	Pedestrian	Bicycle	Wheeled-pedestrian	Motorcycle	Trailer	Vehicle
1	3229	420	-	-	79	502	81553
2	3872	10602	1026	287	225	27	148785
3	6154	33	74	-	-	96	106977
4	4504	-	495	-	95	102	100152
5	6031	21675	3209	189	42	-	174936
6	3522	28523	3545	192	708	57	187861
7	4265	56833	2973	775	684	-	191480
Total	31577	118086	11322	1443	1833	784	991744
%	2.73%	10.21%	0.98%	0.12%	0.16%	0.07%	85.73%

Table 4.6: Distribution of points per class in Cirrus dataset available for evaluation

Qualitative
As discussed in Section 3.3, there are some classes in SemanticKITTI that do not have semantic representation in Cirrus classes thus, the only option is to compare the performance of models on such classes qualitatively. In this section, we will discuss the performance of baseline, cropped, adapted, and densified adapted models on such classes particularly road and vegetation since they are one of the major concerns for ADAS applications. We discuss these under the highway and urban scenarios separately. The highway and urban frames from the Cirrus have been shown in Figure 4.29 and Figure 4.31 respectively.

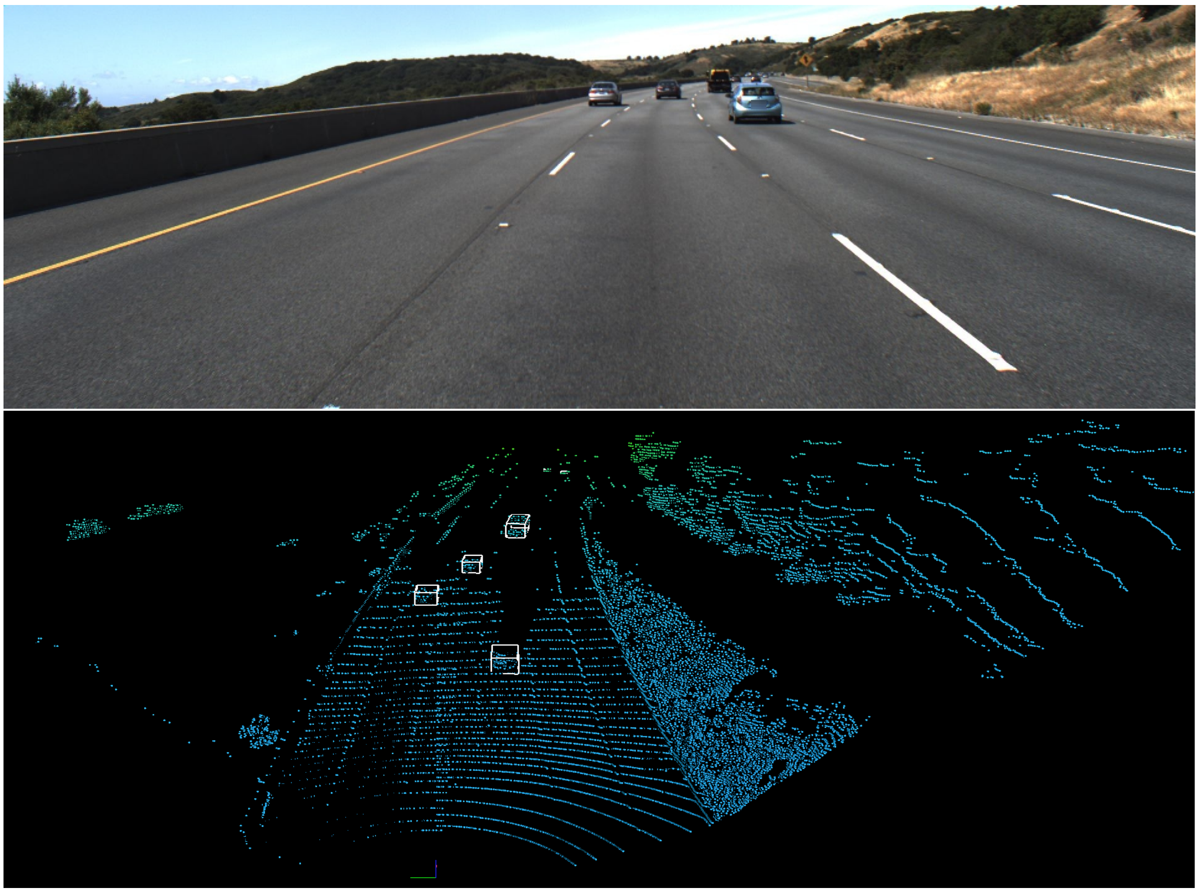


Figure 4.29: Highway scenario of Cirrus

The Baseline model's prediction on the highway scenario of Cirrus (target) can be seen in Figure 4.30. It can be observed that the model miss-classifies a major portion of the road as vegetation and also has trouble in the classification of vegetation near the end of FoV. The cropped model can predict the road class better than the Baseline SemanticKITTI, however, it still miss-classifies vegetation along the end of FoV as multiple other classes. Adapted SemanticKITTI model performs well on the classification of road and vegetation class in comparison to Baseline SemanticKITTI and Cropped SemanticKITTI in highway scenario as it correctly classifies the previously miss-classified road by Baseline as road class and also predicts vegetation with much better accuracy than Baseline and Cropped model. Densified Adapted model performs quite well in identifying road and vegetation along the end of FoV, it still has some miss-classifications of vegetation as buildings but visually performs best among other models.

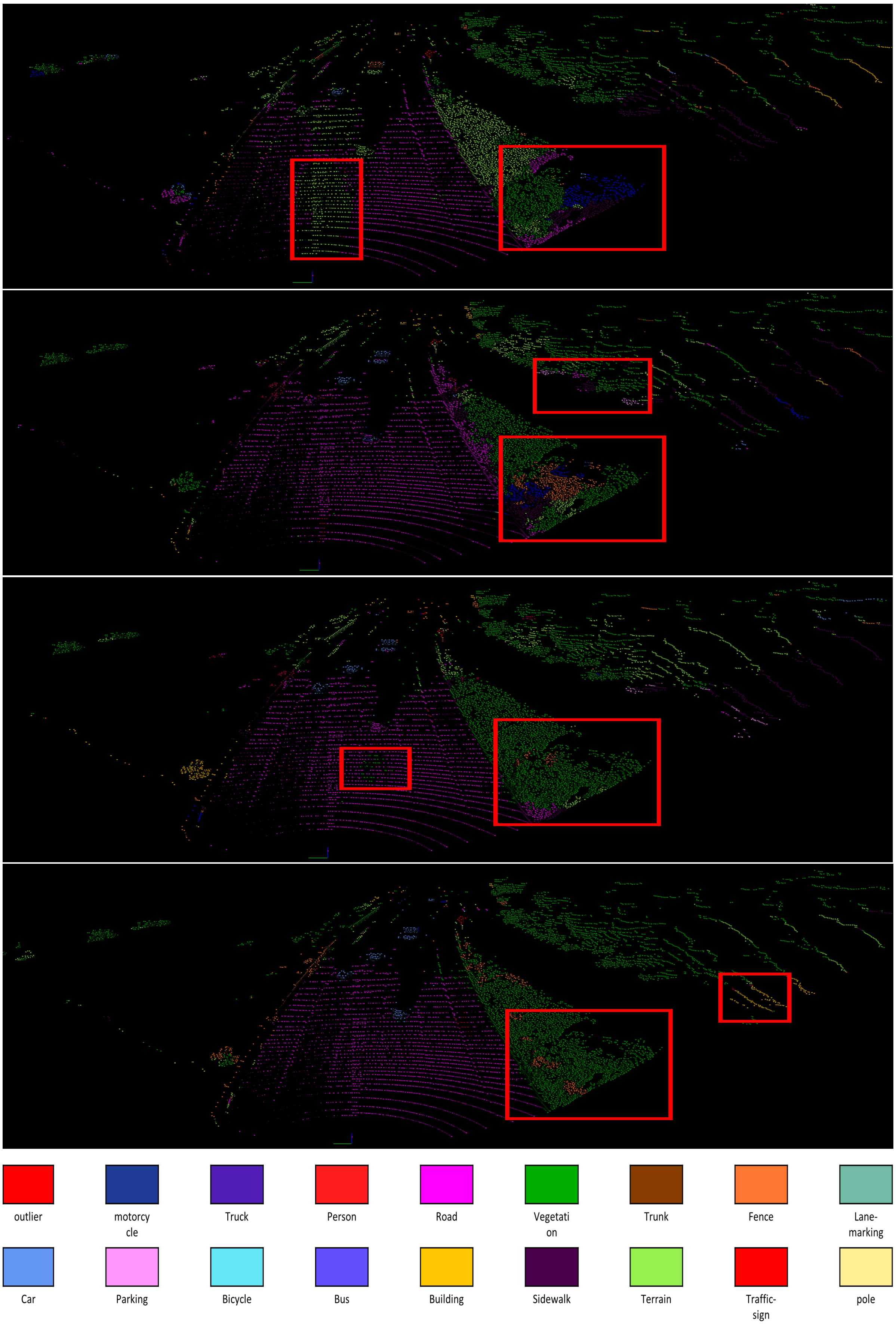


Figure 4.30: SemanticKITTI predictions on Cirrus in SemanticKITTI label space in Highway Scenario; 1) Baseline model 2) Cropped model 3) Adapted model 4) Densified Adapted model

In urban scenario Baseline model miss-classifies road as vegetation, in-fact has troubles even classifying vehicle as can be seen in Figure 4.32. The reason for such performance could be because of the domain difference between the source and target data. Cropped model also makes similar miss-classification therefore, we can not make much of an accurate statement regarding if cropping domain mapping technique helped reduce the domain gap between the source and target domain. Adapted model performs better predictions for road, however, miss-classifies vehicle as vegetation still. However, qualitatively it is visible that adaptation domain mapping strategy helped model perform better on Cirrus, thus, showing a positive transfer learning. Densified Adapted model performs well to identify the road and also the vehicle. However, still has trouble in classification of some other vehicles in the frame. However, qualitatively the Densified Adapted SemanticKITTI performs better than all other models showing that this strategy of domain mapping helped the model learn from the source dataset and generalize on target dataset.

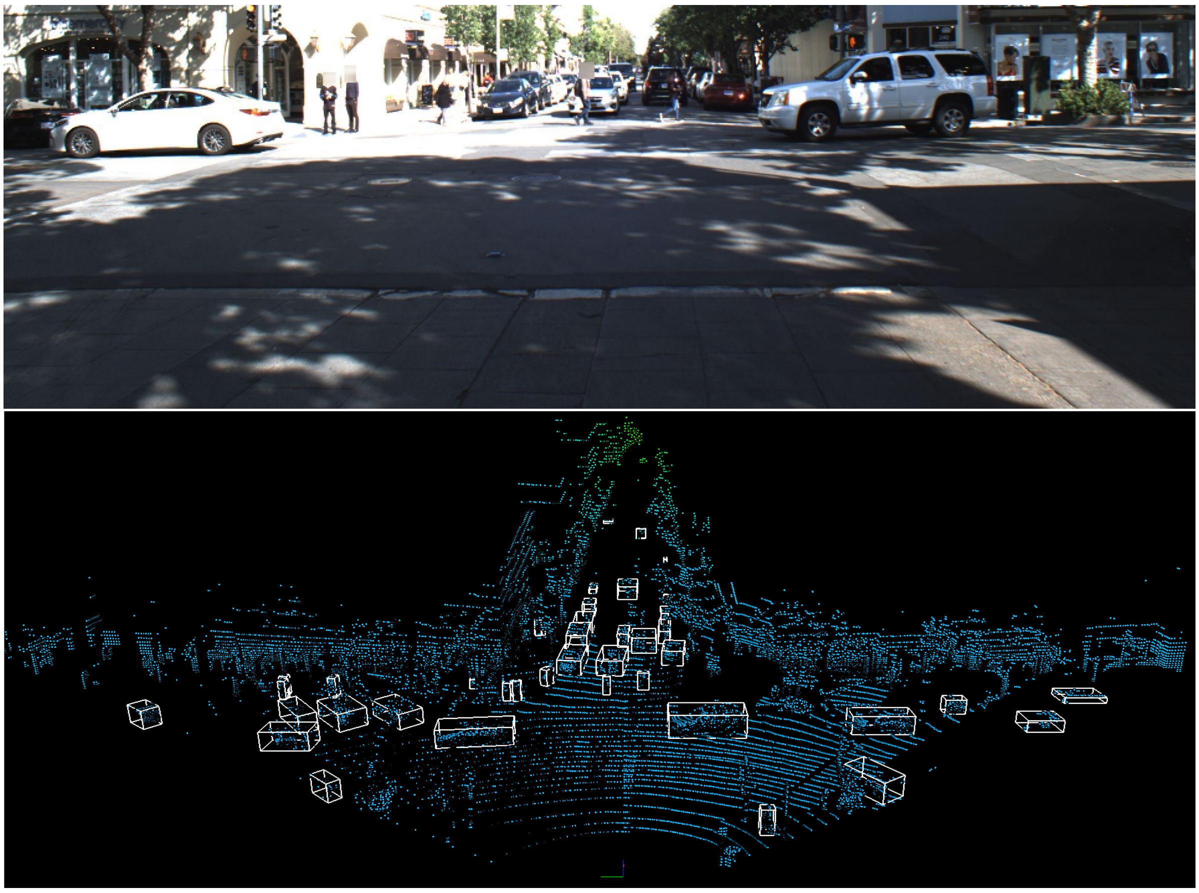


Figure 4.31: Urban scenario of Cirrus

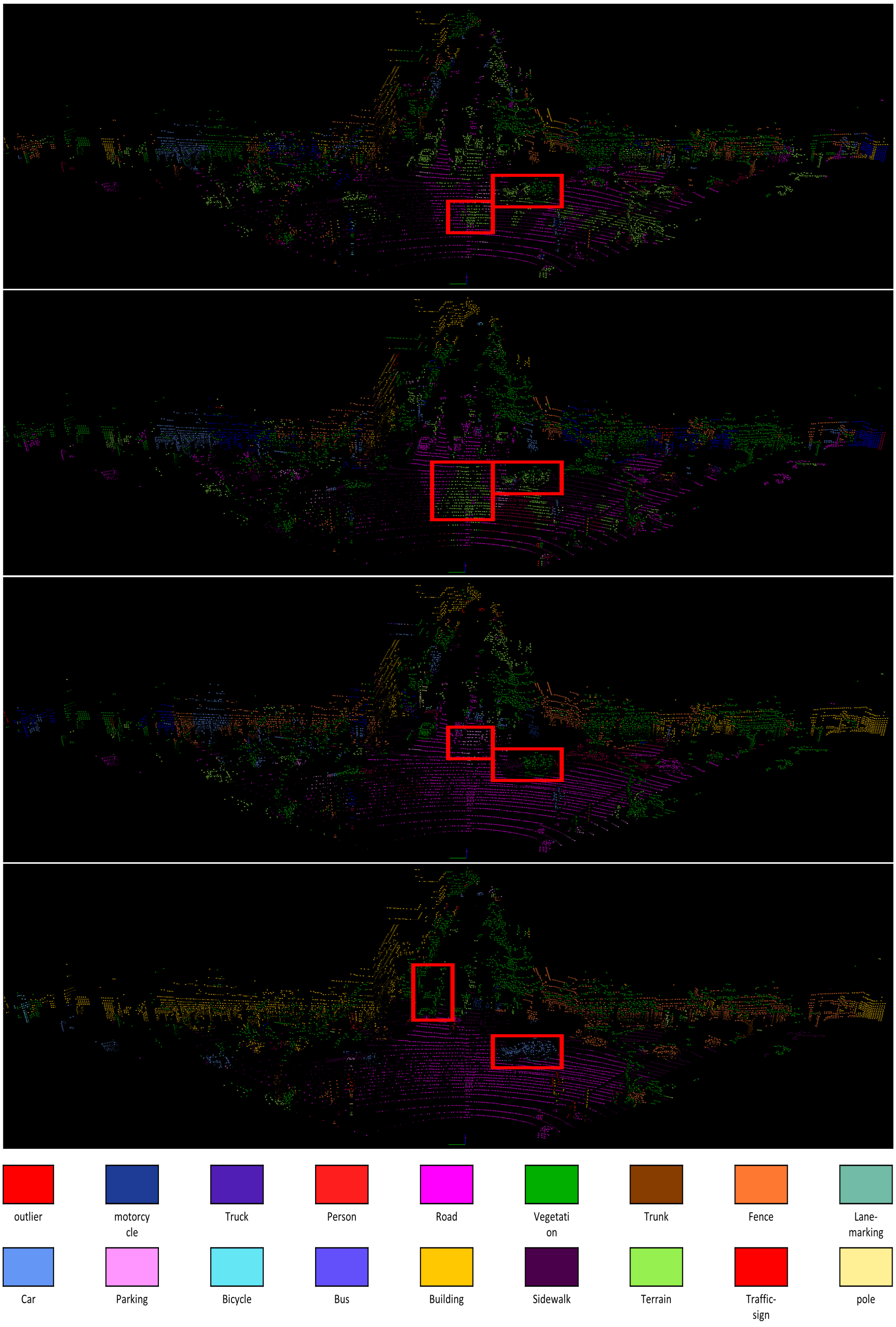


Figure 4.32: SemanticKITTI predictions on Cirrus in SemanticKITTI label space in Urban Scenario; 1) Baseline model 2) Cropped model 3) Adapted model 4) Densified Adapted model

After evaluating the performance of all four models in previous sections, we can conclude that Densified Adapted SemanticKITTI performs best on Cirrus both quantitatively (relative performance shown in Figure 4.28) and qualitatively, showing that the domain mapping strategy used i.e. densification and adaptation using scanning pattern helped reduce the domain difference between the SemanticKITTI (Source), collected using Motorized Optomechanical LiDAR and Cirrus (Target), collected using MEMS LiDAR.

5 Conclusion

In this thesis, we saw how transfer learning methodology can be used to learn from a Motorized Optomechanical LiDAR (Source Domain) and apply the learned knowledge to infer on a Micro-electromechanical Systems LiDAR (Target Domain) and thereby efficiently utilizing the previously accumulated information. This thesis showed that research of MEMS can be boosted by using TL methodologies instead of treating it as fresh research or “re-inventing the wheel”.

This thesis provided an in-depth explanation of the selection of TL approaches for point cloud which are described in Section 2.3 and carefully selects Domain Mapping as a potential candidate, which uses minimal information of the target dataset to map the source to a target dataset creating its proxy.

In Chapter 3, the methodology of adaptation, point cloud densification, and semantic segmentation are explained in detail. Section. 3.1 shows a scanning pattern based pipeline to adapt the source to the target domain i.e. SemanticKITTI to Cirrus by using an image-based sub-sampling methodology. Moreover, this section explains generating a plausible scanning pattern of a LiDAR from a dataset itself. This adaptation strategy shows an immense reduction of information therefore, to further reduce the gap between the source and target domain, this thesis provides a point cloud densification strategy explained in Section 3.2.

Since the SemanticKITTI and Cirrus have different FoV along the azimuth, the basic domain mapping strategy used for completeness is simply cropping the FoV of the source domain. Thus, three models trained on data processed with different domain mapping strategies namely Cropping, Adaptation, and Densification & Adaptation were compared to a model trained on unaltered original SemanticKITTI. These models showed different performances on the Cirrus dataset, showing that Densification & Adaptation domain mapping strategy performs best. It shows a 1.09% improvement in mIoU over the Baseline even with much fewer data points to train from and also no hyperparameter tuning. Moreover, it also outperforms the Baseline qualitatively by correctly predicting road, vegetation, etc where the Baseline fails. Thus, showing a positive knowledge transfer.

5.1. Reflection on Research Questions

This thesis answered the research question posed in Chapter 1.

Q. How can the domain difference between the two categories of LiDAR be reduced?
Answer. The domain difference between the two categories can be reduced by using a Domain mapping methodology of Domain Adaptation solution strategy of Heterogeneous Transfer Learning. One such approach to bring the domains closer in input feature space is by using a scanning pattern-based sub-sampling approach as described in Section 3.1.

Q. Does the density of information after adaptation stay the same in the source dataset?
Answer. The information reduction depends on the methodology used, however, using the Domain Mapping strategy mentioned in this thesis, the information density was reduced tremendously. Nonetheless, was compensated by using a point cloud densification methodology.

Q. How can the scanning pattern of a LiDAR be generated?
Answer. This thesis mentions a pinhole camera modeling strategy (Section 3.1.1) to generate a plausible scanning pattern for a LiDAR using its dataset.

Q. Is the semantic information lost after adaptation?
Answer. The methodology of point cloud densification used in this thesis causes the loss of semantic information of the point cloud. However, this thesis overcomes this disadvantage by using a voxel ground truth superimposition scheme (Section 4.2.3) to generate the semantic labels without manual labeling.

5.2. Limitation

The methodology of domain mapping presented in this thesis has some limitations which need to be taken into account to make amendments to the approach and therefore improve upon it.

First, the scanning pattern generated is a plausible scanning pattern and not the actual scanning pattern, therefore using it might not simulate the true data distribution of the target dataset.

Second, the adaptation process causes very high data compression, therefore, choosing a source dataset with an already sparse point distribution, for example, a dataset collected using a Velodyle VLP-16 might result in the adapted dataset being unusable because it loses too much information. Therefore, the selection of a rich source domain dataset is necessary.

Third, the point cloud densification pipeline described relies on the voxel ground truth of the source dataset to generate the point-wise labels for densified and adapted points. Therefore, a source domain dataset with voxel ground truth labels is required if semantic labels for densified and adapted point cloud is to be generated without manual labeling.

Fourth, since in this thesis the selected target domain dataset (Cirrus) provided ground truth labels as bounding boxes, the evaluation of performance was only done on the points inside of those boxes. Since the bounding boxes were provided for only moving objects like vehicle, pedestrian, etc thus, it limits the performance check only on them and not on stationary objects like road, buildings, vegetation, etc.

5.3. Future Work

This thesis presents many opportunities for improvement which could be taken up for further research.

First, the densification pipeline mentioned does not provide user control over the intensity of densification. It may cause the source to be densified much more than required thus, causing a further increase in the domain gap between the source and target. Therefore, more research can be done in creating a densification scheme with the intensity of output as a hyperparameter.

Second, the semantic segmentation acts as a proxy for evaluating the domain mapping strategy, however, it might be the case that the adopted domain mapping strategy only performs better for semantic segmentation and might give a different performance for other downstream tasks. Therefore, a task-less methodology for calculating the domain difference is required for accurately knowing the benefits of such adaptation schemes.

Third, for uniformity, the hyperparameters while training different models in the thesis were kept constant. However, hyperparameter tuning could benefit the performance even more.

Fourth, the semantic segmentation uses a cylindrical grid that is distributed uniformly, however, research into adaptive grid generation strategy could be studied for more uniform point distribution per voxel.

Bibliography

[1] Inigo Alonso, Luis Riazuelo Luis Montesano, and Ana C Murillo. “Domain adaptation in lidar semantic segmentation”. In: *arXiv e-prints* (2020), arXiv–2010.

[2] Pablo Arbelaez et al. “Contour Detection and Hierarchical Image Segmentation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.5 (May 2011), pp. 898–916. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2010.161`. URL: `http://dx.doi.org/10.1109/TPAMI.2010.161`.

[3] Pierre Baldi. “Gradient descent learning algorithm overview: A general dynamical systems perspective”. In: *IEEE Transactions on neural networks* 6.1 (1995), pp. 182–195.

[4] Jens Behley et al. “Semantickitti: A dataset for semantic scene understanding of lidar sequences”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9297–9307.

[5] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. “The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4413–4421.

[6] John Blitzer, Ryan McDonald, and Fernando Pereira. “Domain adaptation with structural correspondence learning”. In: *Proceedings of the 2006 conference on empirical methods in natural language processing*. 2006, pp. 120–128.

[7] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. “Unstructured point cloud semantic labeling using deep segmentation networks.” In: *3dor@ eurographics* 3 (2017), pp. 1–8.

[8] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[9] Holger Caesar et al. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[10] Rita Chattopadhyay et al. “Multisource domain adaptation and its application to early detection of fatigue”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.4 (2012), pp. 1–26.

[11] Wanli Chen et al. “Tensor low-rank reconstruction for semantic segmentation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 52–69.

[12] Xinjing Cheng, Peng Wang, and Ruigang Yang. “Depth estimation via affinity learned with convolutional spatial propagation network”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 103–119.

[13] Xinjing Cheng et al. “Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 10615–10622.

[14] Yunjey Choi et al. “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8789–8797.

[15] Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. “Transfer learning for activity recognition: A survey”. In: *Knowledge and information systems* 36.3 (2013), pp. 537–556.

[16] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. “Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving”. In: *arXiv preprint arXiv:2003.03653* (2020).

[17] Hal Daumé III. “Frustratingly easy domain adaptation”. In: *arXiv preprint arXiv:0907.1815* (2009).

[18] Mark De Deuge et al. “Unsupervised feature learning for classification of outdoor 3d scans”. In: *Australasian Conference on Robitics and Automation*. Vol. 2. University of New South Wales Kensington, Australia. 2013, p. 1.

[19] Miguel Díaz-Medina et al. “A Voxel-based Deep Learning Approach for Point Cloud Semantic Segmentation”. In: *Spanish Computer Graphics Conference (CEIG)*. Ed. by Dan Casas and Adrián Jarabo. The Eurographics Association, 2019. ISBN: 978-3-03868-093-2. DOI: `10.2312/ceig.20191206`.

[20] Xiaohan Ding et al. “Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1911–1920.

[21] Abdelrahman Eldesokey et al. “Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12014–12023.

[22] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[23] Biao Gao et al. “Are We Hungry for 3D LiDAR Data for Semantic Segmentation? A Survey and Experimental Study”. In: *arXiv preprint arXiv:2006.04307* (2020).

[24] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[25] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[26] Jakob Geyer et al. “A2d2: Audi autonomous driving dataset”. In: *arXiv preprint arXiv:2004.06320* (2020).

[27] Boqing Gong et al. “Geodesic flow kernel for unsupervised domain adaptation”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 2066–2073.

[28] Lei Han et al. “Occuseg: Occupancy-aware 3d instance segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2940–2949.

[29] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[30] M.A. Hearst et al. “Support vector machines”. In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: `10.1109/5254.708428`.

[31] Mu Hu et al. “Penet: Towards precise and efficient image guided depth completion”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13656–13662.

[32] Qingyong Hu et al. “Randla-net: Efficient semantic segmentation of large-scale point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11108–11117.

[33] Jiashen Hua and Xiaojin Gong. “A normalized convolutional neural network for guided sparse depth upsampling.” In: *IJCAI*. Stockholm, Sweden. 2018, pp. 2283–2290.

[34] Jing Huang and Suya You. “Point cloud labeling using 3d convolutional neural network”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 2670–2675.

[35] Mingyang Jiang et al. “Pointsift: A sift-like network module for 3d point cloud semantic segmentation”. In: *arXiv preprint arXiv:1807.00652* (2018).

[36] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[37] Artem Komarichev, Zichun Zhong, and Jing Hua. “A-cnn: Annularly convolutional neural networks on point clouds”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 7421–7430.

[38] Loic Landrieu and Martin Simonovsky. “Large-scale point cloud semantic segmentation with superpoint graphs”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4558–4567.

[39] H Andrew Lassiter et al. “Scan pattern characterization of Velodyne VLP-16 lidar sensor for UAS laser scanning”. In: *Sensors* 20.24 (2020), p. 7351.

[40] Felix Järemo Lawin et al. “Deep projective 3D semantic segmentation”. In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2017, pp. 95–107.

[41] Jiaxin Li, Ben M Chen, and Gim Hee Lee. “So-net: Self-organizing network for point cloud analysis”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9397–9406.

[42] Yangyan Li et al. “PointCNN: Convolution On

$$\backslash$$

mathcal {X} $-TransformedPoints$”. In: *arXiv preprint arXiv:1801.07791* (2018).

[43] You Li and Javier Ibanez-Guzman. “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems”. In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.

[44] Junyi Liu and Xiaojin Gong. “Guided depth enhancement via anisotropic diffusion”. In: *Pacific-Rim conference on multimedia*. Springer. 2013, pp. 408–417.

[45] Zhijian Liu et al. “Point-voxel cnn for efficient 3d deep learning”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[46] Bharat Lohani and Suddhasheel Ghosh. “Airborne LiDAR technology: a review of data collection and processing systems”. In: *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences* 87.4 (2017), pp. 567–579.

[47] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[48] D. Martin et al. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics”. In: *Proc. 8th Int’l Conf. Computer Vision*. Vol. 2. July 2001, pp. 416–423.

[49] Andres Milioto et al. “Rangenet++: Fast and accurate lidar semantic segmentation”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4213–4220.

[50] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[51] Yancheng Pan et al. “Semanticposs: A point cloud dataset with large quantity of dynamic instances”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 687–693.

[52] Jaehyun Park, Chansoo Kim, and Kichun Jo. “PCSCNet: Fast 3D Semantic Segmentation of LiDAR Point Cloud for Autonomous Car using Point Convolution and Sparse Convolution Network”. In: *arXiv preprint arXiv:2202.10047* (2022).

[53] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[54] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).

[55] Jiaxiong Qiu et al. “Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3313–3322.

[56] Christoph B Rist, Markus Enzweiler, and Dariu M Gavrila. “Cross-sensor deep domain adaptation for LiDAR detection and segmentation”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1535–1542.

[57] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[58] Ricardo Roriz, Jorge Cabral, and Tiago Gomes. “Automotive LiDAR technology: A survey”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[59] Amélie Royer et al. “Xgan: Unsupervised image-to-image translation for many-to-many mappings”. In: *Domain Adaptation for Visual Understanding*. Springer, 2020, pp. 33–49.

[60] Khaled Saleh et al. “Domain adaptation for vehicle detection from bird’s eye view LiDAR point cloud data”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[61] Ahmad El Sallab et al. “LiDAR sensor modeling and data augmentation with GANs for autonomous driving”. In: *arXiv preprint arXiv:1905.07290* (2019).

[62] Ahmad El Sallab et al. “Unsupervised neural sensor models for synthetic lidar data augmentation”. In: *arXiv preprint arXiv:1911.10575* (2019).

[63] Tixiao Shan et al. “Simulation-based lidar super-resolution for ground vehicles”. In: *Robotics and Autonomous Systems* 134 (2020), p. 103647.

[64] Chuanqi Tan et al. “A survey on deep transfer learning”. In: *International conference on artificial neural networks*. Springer. 2018, pp. 270–279.

[65] Jie Tang et al. “Learning guided convolutional network for depth completion”. In: *IEEE Transactions on Image Processing* 30 (2020), pp. 1116–1129.

[66] Maxim Tatarchenko et al. “Tangent convolutions for dense prediction in 3d”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3887–3896.

[67] Lyne Tchapmi et al. “Segcloud: Semantic segmentation of 3d point clouds”. In: *2017 international conference on 3D vision (3DV)*. IEEE. 2017, pp. 537–547.

[68] Hugues Thomas et al. “Kpconv: Flexible and deformable convolution for point clouds”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6411–6420.

[69] Marco Toldo et al. “Unsupervised domain adaptation in semantic segmentation: a review”. In: *Technologies* 8.2 (2020), p. 35.

[70] Larissa T Triess et al. “A survey on deep domain adaptation for lidar perception”. In: *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. IEEE. 2021, pp. 350–357.

[71] Larissa T Triess et al. “CNN-based synthesis of realistic high-resolution LiDAR data”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1512–1519.

[72] Jonas Uhrig et al. “Sparsity invariant cnns”. In: *2017 international conference on 3D Vision (3DV)*. IEEE. 2017, pp. 11–20.

[73] Wouter Van Gansbeke et al. “Sparse and noisy lidar completion with rgb guidance and uncertainty”. In: *2019 16th international conference on machine vision applications (MVA)*. IEEE. 2019, pp. 1–6.

[74] Xin Wang et al. “The evolution of LiDAR and its application in high precision measurement”. In: *IOP Conference Series: Earth and Environmental Science*. Vol. 502. 1. IOP Publishing. 2020, p. 012008.

[75] Ze Wang et al. “Cirrus: A long-range bi-pattern LiDAR dataset”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 5744–5750.

[76] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[77] Bichen Wu et al. “Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1887–1893.

[78] Bichen Wu et al. “Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 4376–4382.

[79] Li Yi, Boqing Gong, and Thomas Funkhouser. “Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15363–15373.

[80] Han Woong Yoo et al. “MEMS-based lidar for autonomous driving”. In: *e & i Elektrotechnik und Informationstechnik* 135.6 (2018), pp. 408–415.

[81] Jiaqian Yu and Matthew Blaschko. “Learning submodular losses with the Lovász hinge”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1623–1631.

[82] Hang Zhang et al. “Co-occurrent features in semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 548–557.

[83] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. “Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1607–1616.

[84] Yin Zhou and Oncel Tuzel. “Voxelnet: End-to-end learning for point cloud based 3d object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.

[85] Xinge Zhu et al. “Cylindrical and asymmetrical 3d convolution networks for lidar segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 9939–9948.

[86] Fuzhen Zhuang et al. “A comprehensive survey on transfer learning”. In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76.