12-2022

Instance Segmentation-based Depth Completion Using Sensor Fusion and Adaptive Clustering for Autonomous Vehicle Perception

Mohammad Z. El-Yabroudi
Western Michigan University

INSTANCE SEGMENTATION-BASED DEPTH COMPLETION USING SENSOR FUSION AND ADAPTIVE CLUSTERING FOR AUTONOMOUS VEHICLE PERCEPTION

Mohammad Z. El-Yabroudi, Ph.D.

Western Michigan University, 2022

Depth sensing is critical for safe and accurate maneuvering in robotics and self-driving car (SDC) applications. Most recent LiDAR sensors, such as Ouster and Velodyne, offer 360 degrees of scanning at the rate of ten frames per second, making them very appropriate for autonomous driving applications. However, LiDAR point cloud data show many shortcomings, especially its data sparsity and unassigned nature, making it very challenging to utilize in applications such as perception, 3D object detection, 3D scene reconstruction, and simultaneous localization and mapping.

In this study, a novel framework using instance image segmentation and the raw LiDAR data for the goal of depth completion is developed. The framework uses a custom-trained two-stage instance segmentation architecture to focus on target objects (e.g., cars, pedestrians, and cyclists) and a fusion-based two-branch guided depth completion encoder-decoder deep neural network to generate accurate dense depth information. Results from the extensive experimental work using the KITTI depth completion dataset indicate that the proposed method achieves better performance than the baseline model. Moreover, to address the raw unassigned nature of LiDAR

point cloud data, an adaptive estimation for the tuning parameters of the Density-Based Clustering of Application with Noise (DBSCAN) algorithm in SDC applications is proposed. This method utilizes a field-of-view division scheme and local insights about the LiDAR point cloud data to automate the estimation of the tuning parameters: epsilon and min_points. Experimental simulations using the KITTI object detection dataset achieved excellent clustering performance while waiving the need to the brute force tuning of parameter values.

Aiming to handle the challenges of the sparse and unassigned nature of LiDAR depth data, the key contributions of this dissertation include the development of a depth completion framework utilizing image instance segmentation features, the integration of object type within the depth completion deep neural networks, the development of an adaptive DBSCAN parameters-estimation technique, and the implementation of the instance segmentation-based depth-completion using sensor fusion framework. However, the overarching contribution is the introduction of a fundamental sensor fusion framework that fuses features and information from image instance segmentation and critical SDCs sensors such as LiDARs, RADARs, and cameras, and results in better perception and scene understanding.

INSTANCE SEGMENTATION-BASED DEPTH COMPLETION USING SENSOR FUSION AND ADAPTIVE CLUSTERING FOR AUTONOMOUS VEHICLE PERCEPTION

by

Mohammad Z. El-Yabroudi

A dissertation submitted to the Graduate Collage
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
Electrical and Computer Engineering
Western Michigan University
December 2022

Doctoral Committee:

Ikhlas Abdel-Qader, Ph.D., Chair
Bradley Bazuin, Ph.D.
Osama Abudayyeh, Ph.D.
Rakan Chabaan, Ph.D.

ACKNOWLEDGEMENTS

I want to thank all those who have supported, guided, and encouraged me through this long, challenging work. First and foremost, thank you to my advisor, Professor Ikhlas Abdel-Qader, who, from the first step until the end of this incredible Ph.D. journey, has paved the path to success with her deep knowledge, experience, and futuristic vision. Professor Abdel-Qader acts as a professional advisor and a close friend who always looks for the best choices that align well with my academic interest and success. Without her guidance, advice, assistance, support, and encouragement, I would never reach this milestone and would be lost. Thank you, Dr. Abdel-Qader, for giving me the opportunity to learn from you the academic knowledge and the tremendous soft skills you are always happy to share. I also want to express my gratitude and appreciation for your kindness, humanity, and understanding during the hard times I went through, especially when my family and I got COVID-19 with severe symptoms. I will always be in debt to you.

Meanwhile, I would like to express my most profound appreciation to the entire dissertation committee members for accepting my invitation to be a valuable and critical part of my research journey. This milestone will never be like this perfection without your helpful comments, support, and encouragement. I want to express my sincere appreciation to Professor Bradley Bazuin for all the fruitful discussions and valuable additions you made to my research work and for your trust and confidence in my skills by giving me the privilege and chance to teach several labs and courses

for CEAS students during this Ph.D. journey which enhanced my skills and enriched my experience.Thank you, Professor Osama Abudayyeh, for your valuable reviews,

suggestions, and directions. And a great appreciation for the incredible skills and knowledge I learned from you during the CCE6960 class.

To the final committee member and my industrial soul Professor Rakan C. Chabaan, thank you very much for your insights, industry-level information, and valuable emerging technology updates that aligned my research with industry advancements.

At home, I would like to thank my family, who surrounded me with endless support. Thanks to my father, Ziad, and my mother, Nisreen, who have always been available to talk, encourage, pray, and advise. Thanks to my hero, my wife Athar, who did a great job taking care of me, our kids, and our home. What you did is unbelievable. You are always a great model of patience and altruism. You have been an indispensable part of this journey and will be a crucial part of my life forever. Thanks to my kids, Zaid, Yousef, and Zainah, your voices always kept me motivated, encouraged and optimistic, and they will do so forever.

Mohammad Z. El-Yabroudi

TABLE OF CONTENTS

Table of Contents – Continued

CHAPTER

Table of Contents – Continued

CHAPTER

LIST OF TABLES

LIST OF FIGURES

List of Figures – Continued

List of Figures – Continued

CHAPTER 1

INTRODUCTION

Overview

Self-Driving Cars (SDCs), also known as Autonomous Vehicles (AVs) or driverless cars, have been envisioned since the middle of the 1980s. However, they have recently reached the implementation and testing phase. This is due to the significant improvements in essential technologies such as sensors and real-time computing, which made accurate real-time decisions possible. The SDC architecture consists of four modules: navigation system, environmental perception, path planning, and car control. The environmental perception module is designed to process information collected by various sensors such as cameras, radio detection and ranging (Radar), light detection and ranging (LiDAR), Sonar, Global Positioning System (GPS), and Inertial Measurement Unit (IMU). The environmental perception module also receives information from other road cars or infrastructure devices. The entire bag of information is then forwarded to the planning module, which guides the control module to drive the vehicle safely and correctly. This architecture is depicted in Figure 1, which shows the leading technologies, modules, and their interactions [1]–[4]. To control the advances in SDCs, the Society of Automotive Engineering (SAE) published classification criteria to define the levels of autonomy in autonomous cars, primarily concerned with the level of human interaction in the driving process. Figure 2 depicts the six SAE levels, which range from 0 as the basic car system wherein a human driver is essential, and five, where no human driver is needed in all circumstances [5]. The United States is still at level 2 of the SAE classification system; the vehicle provides one or more

automated systems while the motorist does the rest [6]. The slow development from one level to the next is due to many factors, such as regulations, testing, and technology. From a technological perspective, the performance of many critical systems within the SDCs is still not elevated to the completeness and perfection needed by higher levels of autonomous driving, such as levels 3, 4 and 5, or it requires huge computing resources that are not yet technically feasible, cost-effective or commercially available. For example, the depth completion process within the perception module of many recently published works still performs poorly in scenarios such as when objects are small, far, occluded, when severe weather conditions exist, or where they require high-performance computing resources and large datasets as well as extended training time[7] [8].

Sensors play a vital role in SDCs, and they vary in their technology, the range they span, and their immunity to external effects like lighting and weather conditions. LiDAR uses laser beam technology to measure the distance from objects. LiDAR provides three-dimensional (3D) information, essential to many SDCs tasks such as 3D object detection, collision prediction, and obstacle avoidance. LiDAR is very good in night scenarios as well as in inclement weather conditions. RADAR uses radio waves technology; standard systems are either 77 GHz or 24 GHz. Depending on the frequency, RADAR can be distinguished as short, medium, or long-range radar [9], [10]. Ultrasonic sensors use sound waves technology to emit high-frequency signals that measure the distance from objects after reflection. Ultrasonic sensors have a limited coverage area, usually a few meters; thus, their applications are limited to low-speed applications like automatic parking assistance systems. The camera's technologies are based on light reflection from the environment; images from the camera vary in quality based on different external conditions like weather and illumination. GPS systems utilize satellites for the instantaneous measurements of position and time, which are often used for navigation or geolocation. An inertial measurement

unit (IMU) is a combination of two technology: Micro Electro Mechanical Systems (MEMS) and Gyroscopes, and it provides information like acceleration forces and angular rates [3], [11].

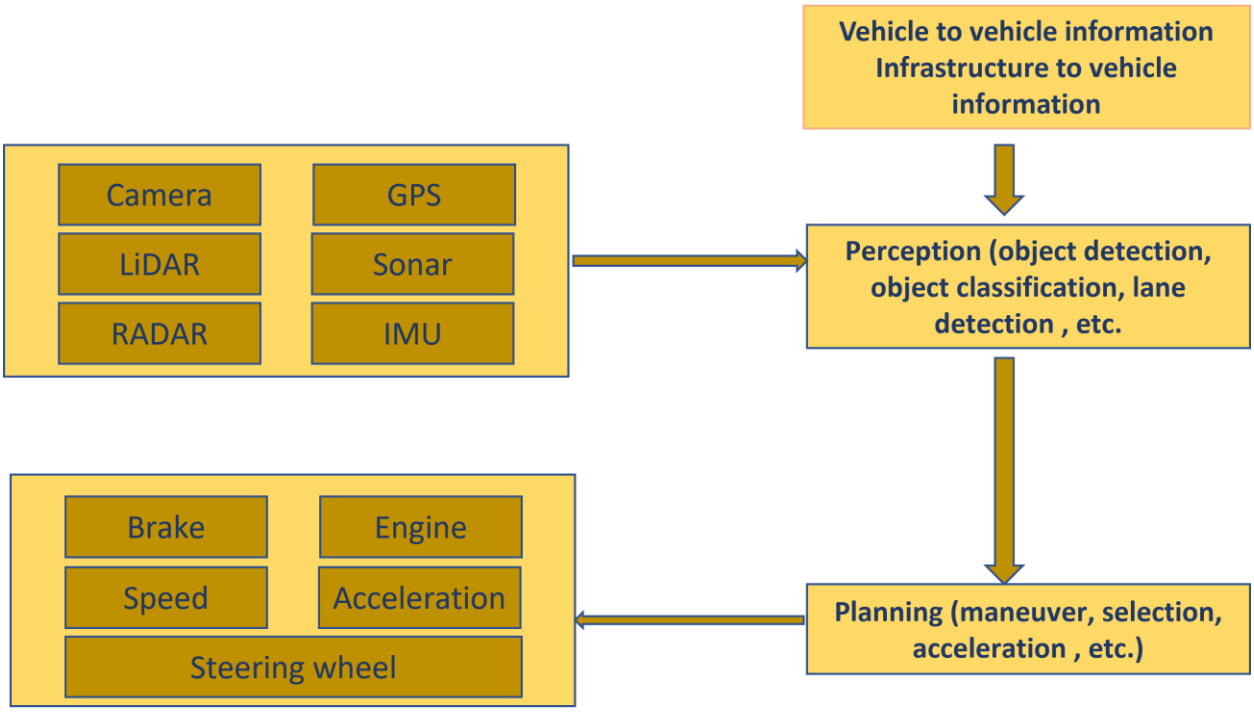

Figure 1. Self-driving car architecture shows the perception module and its interaction with other modules and information sources[1], [4]

Each sensor has its strength and weakness for application in SDCs. For instance, Camera sensors are affordable and very efficient in classification tasks, but they require high computational power, and their performance degrades in severe weather conditions. On the other hand, LiDAR is an expensive sensor that provides information in 3D format. RADAR is handy in bad weather conditions but has less angular accuracy and provides less information than cameras and LiDAR. These differences in performance capabilities between different sensors encouraged researchers to utilize the complementary properties of each sensor by combining them in a process usually called sensor fusion or data fusion. Data/sensor fusion benefits multiple tasks like object detection,

occupancy grid mapping, and object tracking [2]. More information will be provided in the background section.

The perception module is responsible for critical tasks like object detection and classification, semantic segmentation, and localization. The perception module uses single or multiple sensors to acquire the surrounding environment information and then apply different detection and classification algorithms to process the data and generate detections and classifications. Recently, 3D information has become a significant clue for applications like robotics, self-driving cars, 3D modeling, and augmented reality. Depth information can be captured by two general methods: active and passive. In passive methods, two cameras are used to generate depth information by utilizing parallax, intrinsic and extrinsic camera specs, and stereo-matching techniques. In active methods, structured light, shape from shading, shape from texture, and time of flight methods are used to estimate the depth of a given scene. LiDAR is a commonly used sensor within the active depth generation methods which rely on the time of flight to measure the depth of objects within a scene [10], [12]. Unfortunately, even with high-quality LiDARs, the produced depth maps are extraordinarily sparse and noisy. Therefore, a depth completion process is usually conducted to recover denser maps from sparse, noisy ones. To that end, a wide range of techniques have been proposed, which can be categorized by the modalities they use or the technology they exploit. In the first category, single or multiple sensors are used, where the former is usually referred to as non-guided depth completion, and the latter is guided depth completion. In the second category, pure image processing or artificial neural network are typically utilized. Recent solutions are based on deep convolutional networks [13].

Motivation

Self-driving cars have been designed to reduce road fatalities and fuel consumption by eliminating human errors. Although industrial progress of automation is at level 3, commercially available vehicles in the USA are still at level 2 of automation. Moving forward toward higher levels requires improvements in different technological aspects. Accurate and rich 3D information plays a critical role in advancing SDCs. The current state-of-the-art techniques that enhance 3D knowledge are promising and produce good results. However, the training requirement is very resource-intensive and time-consuming, and the performance on object edges or distant objects still suffers from high errors. Most current work focuses on the entire scene and does not treat objects of interest individually or separately. Additionally, applications such as robotics and self-driving cars require both fast training mechanisms and fast inference because in some scenarios, training could be needed in real-time while the SDCs or robots are on site. Moreover, using a vast number of resources and requiring long training time could limit other researchers' ability to study and improve the current state of the art.

This dissertation focuses on the perception module of SDCs, specifically on the sparse LiDAR point cloud depth completion and clustering, which are essential in many perception tasks, such as 3D object detection and classification. The dissertation has two main parts: in the first part, a sensor fusion and instance segmentation-based object depth completion framework is introduced, and in the second part, the LiDAR point cloud is clustered using DBSCAN with an adaptive and automatic parameters selection.

Problem Statement, Challenges, and Goals

This research aims to investigate a novel guided depth completion methodology and LiDAR point cloud clustering using density-based spatial clustering of applications with noise (DBSCAN) with automatic tuning parameters selection in a self-driving context.

Depth completion is a widely investigated technique in vast research areas like machine vision and image processing. It deals with methods that generate dense depth maps from sparse measurements. Although strategies that tackle this problem are vast, they generally fall into two main categories: guided and non-guided. In guided depth completion methods, a high-resolution color image is usually fused into the system to provide complementary information to the primary sensor (LiDAR). In contrast, in the non-guided depth completion methods, only LiDAR data is used to generate a dense depth map. Most of the current state of art is Deep Neural Network (DNN) based solutions. Still, they focus on the entire LiDAR point cloud without concentrating on the objects of interest like cars, pedestrians, or cyclists. As a result, DNN training usually requires a large amount of data and computing resources. As an alternative, this work will utilize image instance segmentation which provides exciting information like pixel-level object boundaries and object class, among others. This information can significantly help depth completion to focus on the objects of interest and thus provide higher accuracy.

The dense depth map is then consumed by different applications like 3D object detection and 3D object classification, which commonly apply point cloud clustering; deep neural networks and custom clustering algorithms are usually used. DBSCAN is a viral density-based clustering algorithm used in many applications. None of the previous works tackle the scenario when the LiDAR is installed on a moving car, like in SDCs or mobile robotics applications. In these

applications, the relation between the LiDAR points and the SDC. e.g., Euclidean distance or the LiDAR points interrelation can be considered. Furthermore, no previous works evaluated DBSCAN improvements on available SDCs datasets like the KITTI dataset. Therefore, this dissertation proposes an adaptive and automatic estimation for DBSCAN parameters in SDCs applications and evaluates the proposed work on the SDCs KITTI dataset.

The following challenges have been observed regarding the depth completion frameworks:

- Requires a lot of computing resources, especially for training.
- Applies algorithm on the entire sparse depth without focusing on objects of interest.
- Need hundreds of thousands of annotated training data files.
- Even with good computing resources, the training time is long.

The following challenges have been observed regarding LiDAR point cloud clustering:

- Manual selection for clustering parameters is not practical in dynamic environments like SDCs.
- Most current clustering works are based on deep learning techniques that require a long training time and an extensive dataset.
- Clustering objects that are very close to each other is still an issue.

This dissertation aims to introduce a new depth completion framework that focuses on the objects of interest by utilizing image instance segmentation and multi-level sensor fusion. Moreover, this dissertation provides an automated technique to estimate tuning parameters of the DBSCAN algorithm for LiDAR point cloud clustering in SDCs applications.

The goals for this work are:

Goal 1: Implement and evaluate a depth completion framework that utilizes image instance segmentation and sensor fusion to generate dense depth information for the objects of interest.

Goal 2: Develop a framework based on the DBSCAN algorithm to cluster LiDAR point clouds, focusing on automating the estimate of DBSCAN tuning parameters.

The goals of this work have been achieved through the following contributions:

Depth completion work:

i. Customized an instance segmentation framework to generate accurate object masks and object classes.

ii. Implemented a data structure and encoding scheme to combine instance segmentation features into a single 2D one-channel array.

iii. Fused information from different sensor modalities alongside instance segmentation features.

iv. Customized a depth completion framework that accepts interesting features like instance segmentation masks and object classes and generates dense depth measurements.

v. A comprehensive evaluation of the implemented framework the KITTI depth dataset.

LiDAR Point cloud clustering work:

i. Implemented a SDC field-of-view division scheme.

ii. Clustered LiDAR point cloud using DBSCAN but with automatic parameter selection, and for this purpose:

iii. Proposed equations that can estimate the DBSCAN tuning parameters automatically and dynamically.

iv. Evaluated the proposed work on the KITTI object detection dataset

.

Organization

The dissertation is organized as follows.

In Chapter 1, a general introduction to the topic of Self Driving Cars with emphasis on the perception module is provided. Chapter 2 presents a comprehensive background survey for the perception module alongside typical SDCs sensors. In chapter 3, the very related work is presented and discussed. The Proposed methods are presented in chapter 4, while chapter 5 provides the experimental results. Chapter 6 provides conclusions and future works.

CHAPTER 2

BACKGROUND

In 2021, in the United States of America, 42915 people died from car accidents, about 117 people per day [14]. Reducing car accidents on the roads and boosting driving efficiency are among the most critical objectives for advancing autonomous driving technology. Self-driving cars have been in the vision of many academia and industrial sectors since the 1980s, well-known examples from the 1990s are the Navlabs mobile platform, University of Pavia's and University of Parma's car "ARGO", and UBM's two vehicles: "VaMoRs" and "VaMP". Many challenges have been launched to advance autonomous vehicle technology, like those organized by the Defense Advanced Research Projects Agency (DARPA): DARPA Grand Challenge and DARPA Urban Challenge. Since then, many other competitions have also occurred [2]. SDCs have the same transportation capabilities as traditional vehicles but can also perceive their surroundings and self-navigating with little or no human interaction.

The J3016 "Levels of Driving Automation" standard was introduced in 2014 by SAE International, formerly known as the Society of Automotive Engineers (SAE), and it is continuously updated. The J3016 standard specifies six levels of driving automation, ranging from SAE level 0 (complete driver control of the vehicle) to SAE level 5 (full control of all aspects of dynamic driving activities without human intervention). Figure 2 depicts a high-level overview of these levels, frequently quoted and referenced by industry in the safe design, development, testing, and deployment of highly automated vehicles (HAVs).

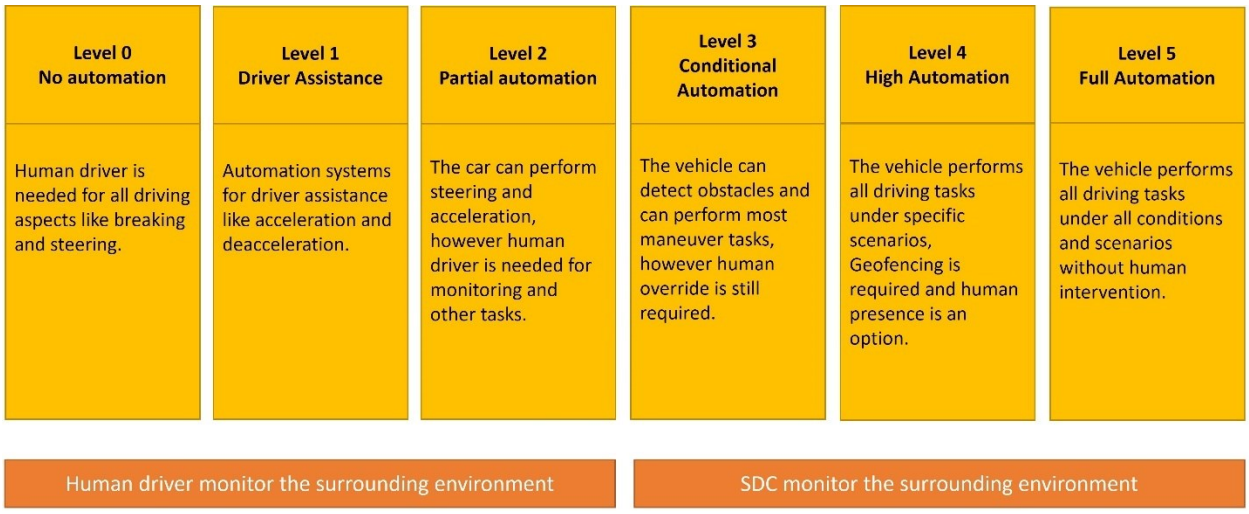

Figure 2. Automation levels published by the Society of Automotive Engineers J3016 standard [5]

This background section focuses on the perception module and provides an overview of the most common sensors available for autonomous vehicles. Also, it gives a brief overview of the main perception tasks like object detection and classification, depth completion, and instance segmentation. This section also provides an overview of the DBSCAN algorithm.

Sensors in Self-Driving Cars

Sensors are electronic-mechanical devices usually attached to autonomous cars to detect events or changes in the surrounding environment and report these detections to subsequent processing units for proper safe driving. Sensors can be classified in different ways; from an operational perspective, sensors fall into two categories: internal and external state sensors [9], [15]. Examples of internal state sensors are IMU, encoders, and positional sensors; this category measures internal dynamic values like 3D acceleration, angular rate, wheel speed, and load. Examples of external state sensors are Cameras, RADARs, and LiDARs. These sensors measure external information like specific distances and color image information about the surrounding

environment [8]. The following subsections will provide comprehensive information about popular sensors usually installed in SDCs.

Camera

Camera technologies are widely adopted to perceive the surrounding environment. Cameras detect light reflected off or emitted from the surrounding objects through a lens and project it onto a photosensitive surface. From a cost perspective, cameras are inexpensive, and they can detect moving and static objects; however, for moving objects, extra software and processing are needed to correlate multiple images (frames). Cameras are suitable for applications that need to identify road signs, traffic lights, and road lane markings, all color/intensity-related tasks. Figure 3 shows examples of camera applications with respect to the mount location. Camera systems can be either monocular (single camera) or binocular (multi cameras). Usually, AVs utilize both types. Single cameras provide rich color information but cannot provide depth information. On the other hand, multi-cameras can be installed side by side to form a stereo camera that can provide both color and depth information. Stereo cameras consist of at least two image sensors, separated from each other by a known distance, and they employ the same depth perception process used by animals; images from two cameras for the same scene implies slight differences, and these difference are used to calculate depth by exploiting a disparity map and epipolar geometry and triangulation methods [3], [11], [16]–[18].

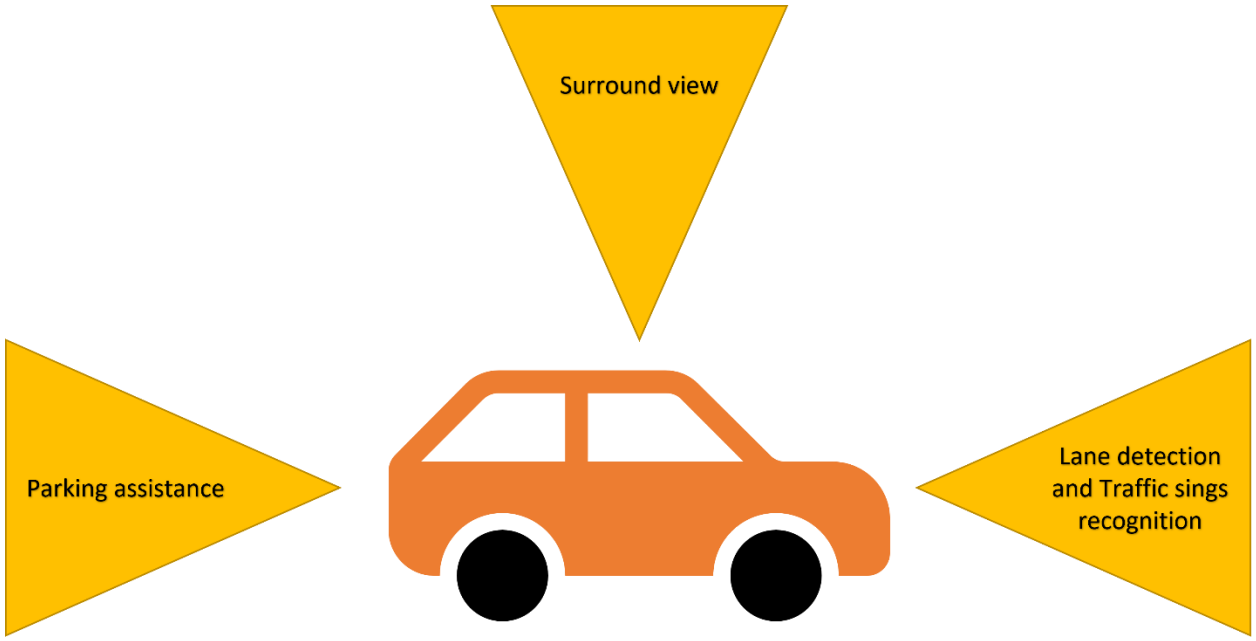

Figure 3. Camera applications with respect to the mount location on self-driving cars [18]

LiDAR

LiDAR is an external sensing device that works on the principle of producing laser light pulses that reflect off target objects. The equipment detects these reflections, and the time between emission and reception of the light pulse allows for distance estimation. LiDAR sensors can be categorized based on the information they deliver; three main categories can be identified: 1D, 2D, and 3D LiDARs. The most commonly used LiDAR type is the 3D LiDAR sensors. Their output usually consists of four values: x, y, and z coordinates and the intensity information of the objects within the surrounding environment. For autonomous driving applications, LiDAR sensors internally consist of multiple layers (channels) of emitting sensors; 64 or 128 channels are commonly employed to generate LiDAR images (or point cloud data) for self-driving context. On the other hand, 1D LiDARs only measure the distance of the surrounding environment using one

single coordinate, e.g., x coordinates. 2D LiDAR provides information about the angle, aka x and y coordinates. Figure 4 depicts a basic operation of the LiDAR.

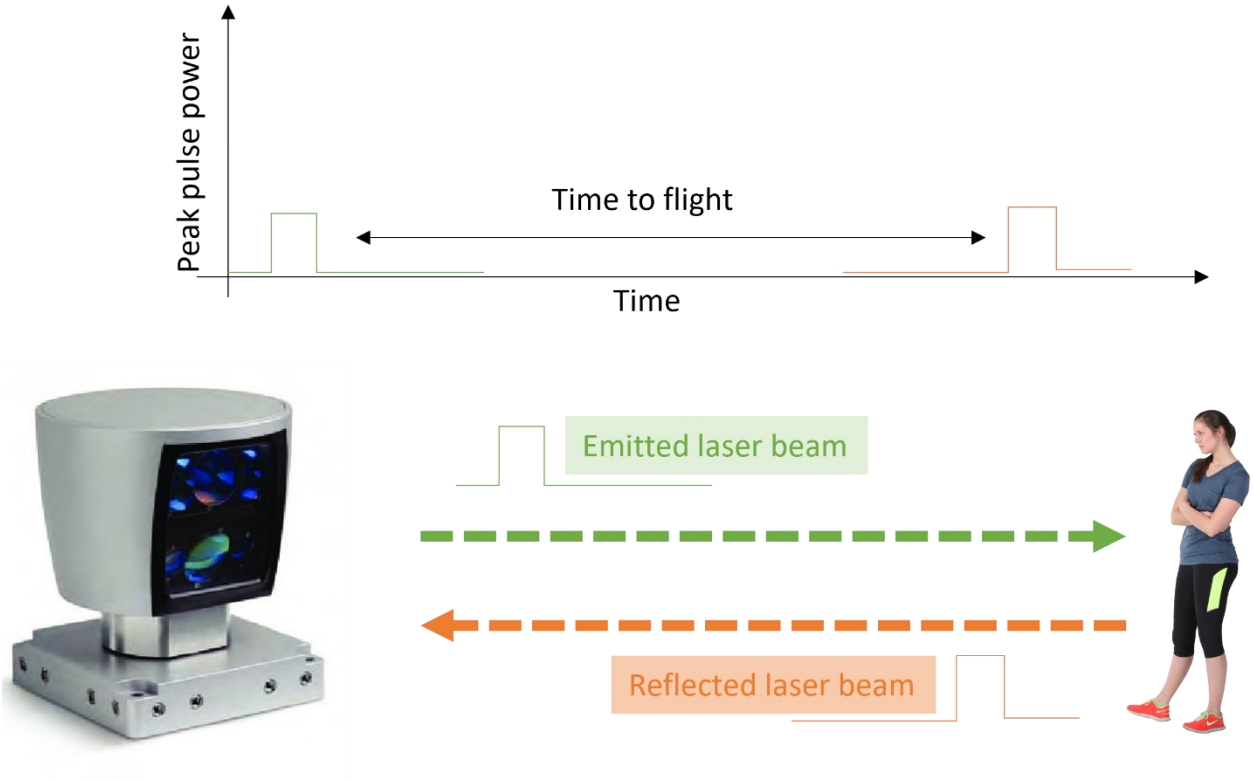

Figure 4. LiDAR operation mechanism illustration [19]

In general, due to their reliability and ability to provide high-quality perception in both day and night settings, 3D spinning LiDAR is the most popular LiDAR in the autonomous context [2], [18]. Figure 5 depicts a typical 360-degree LiDAR sweep. LiDAR beams are emitted at all 360-degree horizontal angular directions and then collected back. The sensor specification defines how far a laser beam can travel. For example, the KITTI dataset [16] uses Velodyne HDL-64E [20], which has 64 channels and can emit and measure beams up to 120 meters in distance. Velodyne

HDL-64E has a 360° horizontal and 26.9° vertical field of view with 0.08° angular resolution (azimuth) and around 0.4° vertical resolution.

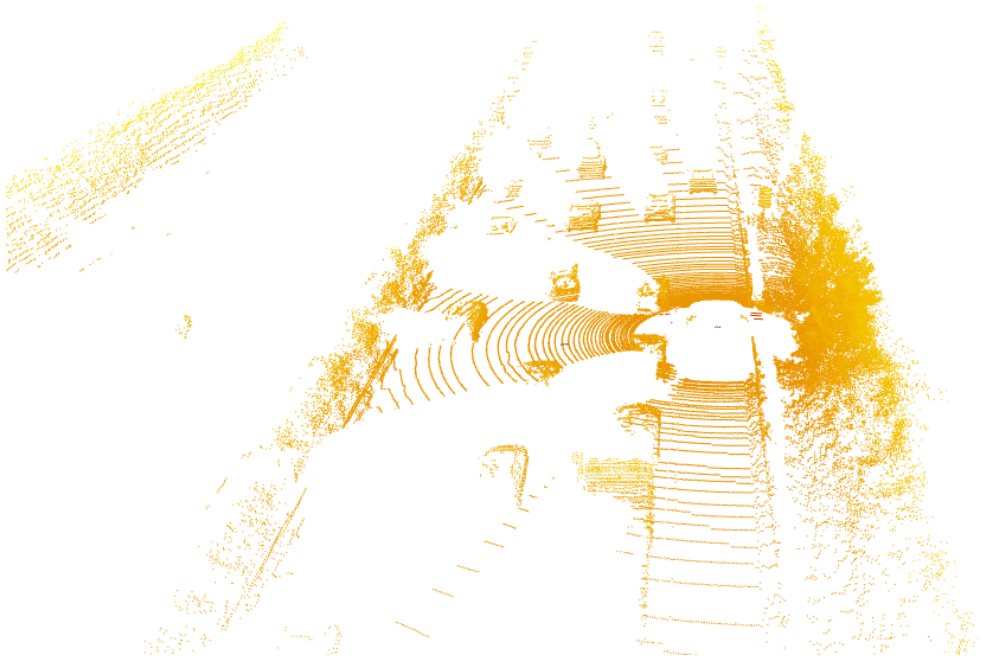

Figure 5. LiDAR point cloud sweep example [16]

RADAR

RADARs work on emitting periodic Electromagnetic (EM) waves within the region of interest and receiving reflected waves from targets for signal processing and determining range information. It can determine the relative speed and position of identified obstacles using the Doppler property of EM waves. The Doppler effect describes how relative motion between a wave source and its targets causes variations or shifts in wave frequency. When the target travels toward the radar system's direction, the frequency of the detected signal rises. Different operational frequencies are available in the market, 24 Gigahertz (GHz), 60 GHz, 77 GHz, and 79 GHz; the

lower the frequency, the lower the range, velocity, and angle resolution the RADAR provides. Radar can operate day or night in foggy, snowy, or cloudy situations because the propagation of EM waves (radar) is somewhat immune to unfavorable weather conditions and the function of radar is independent of the ambient illumination conditions. The misleading identification of metal objects around the perceived surroundings, such as road signs or guardrails, is one of the downsides of radar sensors, as are the difficulties in differentiating static, stationary objects [18]. Radar sensors are often mounted on multiple locations within SDCs, such as on the roof, around the top of the windshield, and behind the vehicle bumpers. Maintaining the precision of mounting positions and orientations of radars is critical since any angular misalignment could result in incorrect vehicle operation and lead to inaccurate or late detections of hazards around the vehicle. The three major types of automotive radar systems are Medium-Range Radar (MRR), Long-Range Radar (LRR), and Short-Range Radar (SRR). SRR is used for parking assistance and collision proximity warning, MRR is used for side/rear collision avoidance and blind-spot recognition, and LRR is used for adaptive cruise control and early detection[11], [18], [21], [22].

It is noteworthy that each sensor has its advantages and disadvantages; for example, LiDAR provides depth information and is immune to light variation; however, its data lacks color information. On the other hand, a Camera provides color information but is very sensitive to light variation and weather conditions. Sensor fusion combines data from multiple sensors and thus can provide a richer representation. Table 1 provides a comparison between the three primary sensors: Camera, LiDAR, and RADAR, from different factors and applications.

*Table 1. Comparison between different sensors' capabilities with respect to different factors or applications, x means the sensor is not appropriate for this application or factor, * means the sensor can work well with this task or factor, ** means the sensor is aptly performing with this task or application*

Factor	Camera	LiDAR	RADAR
Range	*	**	*
Resolution	**	*	x
Distance accuracy	*	**	**
Velocity	*	x	**
Color Perception	**	x	x
Weather conditions	x	*	**
Illumination Conditions	x	**	**
Application	**Camera**	**LiDAR**	**RADAR**
Object detection	**	*	x
Object classification	**	*	x
Lane detection	**	x	x
Depth completion	*	*	x

Perception Module

In a process very analogous to human visual cognition, the perception module examines raw sensor data and other information from other vehicles and infrastructure to generate an environmental understanding which is very important to many other modules like the planning module. Examples of the main tasks related to the perception module are object detection and classification, object tracking, semantic segmentation, and depth completion, among others.

Perception technologies in the state-of-the-art can be divided into two categories: computer vision-based and machine learning-based. The former tackles visual perception issues by defining them with explicit projective geometry models and utilizing optimization techniques to discover an optimal solution. On the other hand, Machine learning technologies depend on data to drive solutions, problems usually modeled like regression or classification problems. In classification problems, data are generally assigned into multiple classes (multi-class classification); for example, car, bicycle, or van. Alternately there may be two classes (binary classification); for instance, a LiDAR point is either a foreground or background point. In regression problems, values such as object heading and angles are regressed. In the subsequent sections, three important perception tasks are discussed: depth completion, object detection/classification, and instance segmentation [23]–[25].

Depth completion

Depth completion is a critical task in machine vision and robotics. LiDAR sensors can only provide sparse depth information, and when projecting these depth values to the image coordinate system, many pixels are left without depth information. For example, on the well-known and widely used autonomous driving dataset KITTI [16], only 5-7% of pixels have a valid depth measurement. Figure 6 shows an example generated from the KITTI dataset; the bottom image is the sparse LiDAR point cloud, while the top image provides the Camera 2D image for reference. On the other hand, applications like 3D object detection and classification can only work properly if there are enough valid depth points. To that end, depth completion is usually performed where a dense depth map is generated from a sparse set of measurements. This process can be done with or without the guidance of other modalities like camera data. Techniques that utilize camera data benefit from the structure information available within the images; thus, their performance usually

surpasses other methods. An example of the expected completed depth maps is shown in Figure 7.

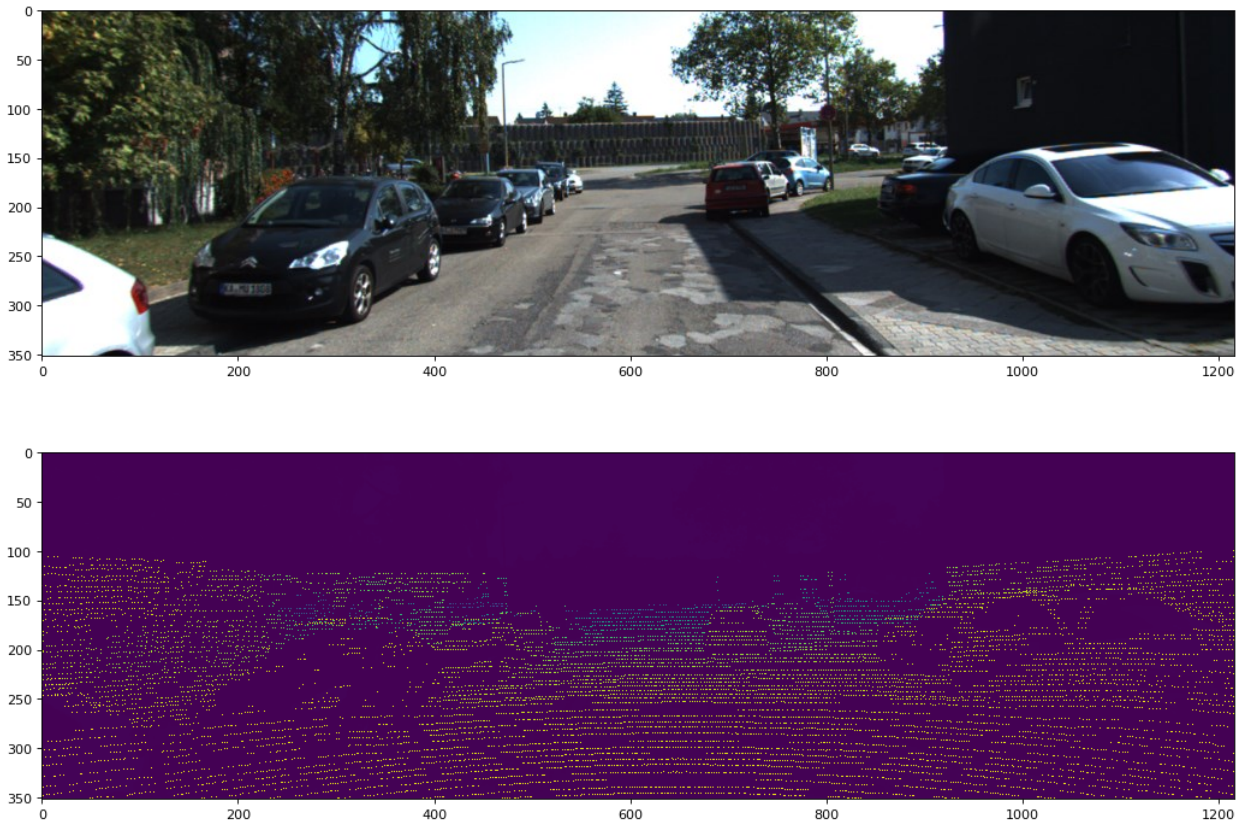

Figure 6. Sparse LiDAR point cloud (bottom) and the corresponding reference camera image (Top) [16]

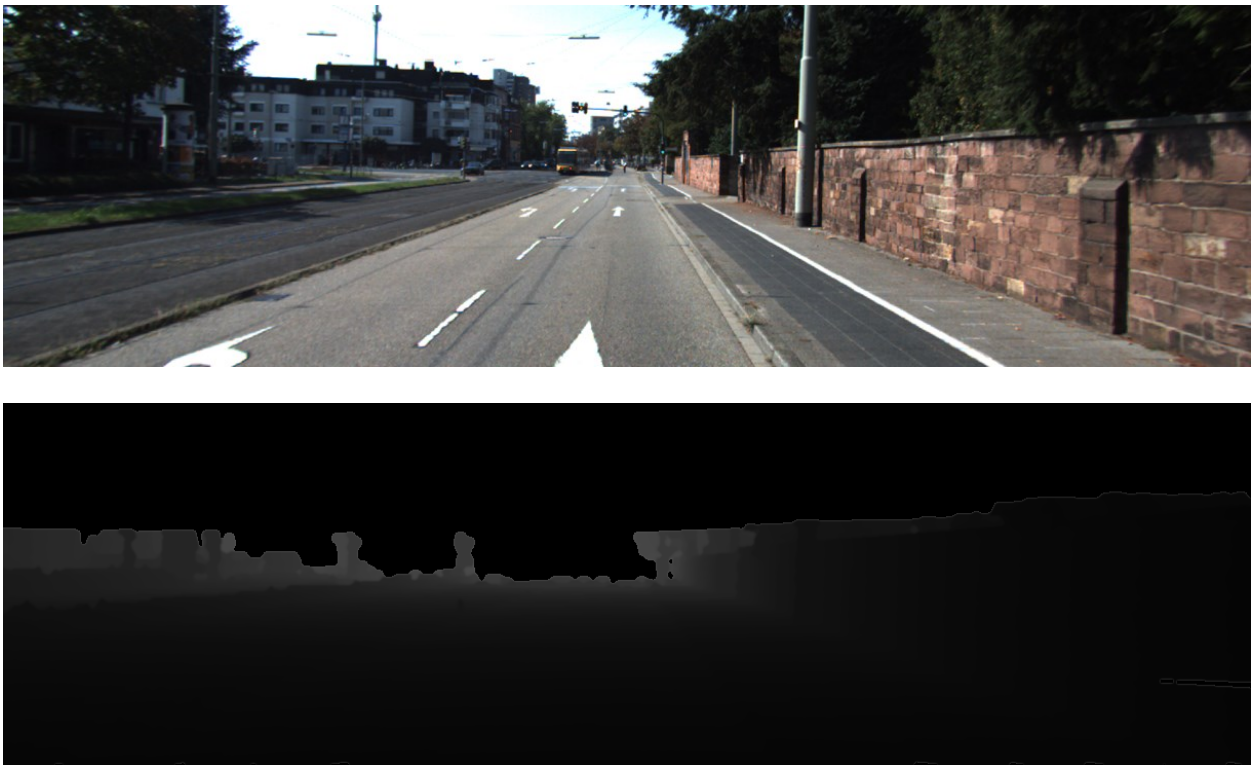

Figure 7. Example of the expected completed depth map image (bottom image), pixels with whiter color belonging to the far objects, and the camera image for reference (top image) [16]

a) Non-guided depth completion

In non-guided depth completion, methods only utilize LiDAR data. Examples of approaches include [26] used bilateral filtering in a non-guided depth completion framework to maintain the edges information, while [27] proposed a complete deep neural framework with sparse convolution layers which take into account the location of missing data to tackle the depth completion problem and only rely on the sparse LiDAR information. Moreover, [28] proposed CNNs that focus on the uncertainty of depth data in both the input and the output; the work uses an input confidence estimator to identify distributed measurements in the information; meanwhile, a normalized convolutional neural network is utilized to produce an uncertainty measure for the final output. [29] used compressed sensing techniques and Alternating Direction Neural Networks

(ADNN) to tackle the depth map completion problem. The adoption of ADNNs enabled the implementation of a deep recurrent encoder-decoder framework. Figure 8 shows a simplified framework for a non-guided depth completion framework where the only input is the sparse LiDAR depth map.

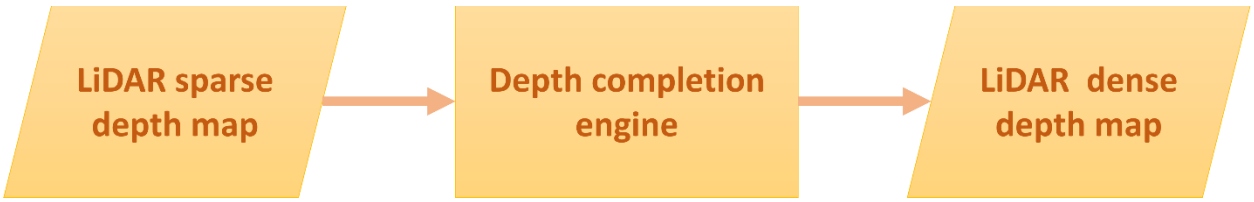

Figure 8. Simplified depiction of the non-guided depth completion framework [30]

b) Guided depth completion

Combining information from different but related sensors has led to remarkable performance improvement in many applications. Many fusion strategies have been proposed, and the field is still wide open for additional contributions. Different works suggest combining data from various sensors like LiDAR and Camera for depth completion problems. Figure 9 shows a simplified framework for a guided depth completion framework. [31] consider the depth completion a regression problem and utilize a single convolutional neural network that takes both RGB images and a sparse depth measurement as input and produces a dense depth map. The proposed network is an encoder-decoder style wherein the encoder is a resNet CNN followed by a convolutional layer. On the decoder side, upsampling layers are followed by bilinear upsampling. The work examined different loss functions and reported that L1 loss (2-1) produced better performance. [32] proposed a deep regression network with encoder-decoder style, where data

from LiDAR and Camera are fused within the network. Skip connections are used to pass features from encoding layers to the corresponding decoding layers.

$$L1 = \left| y_{actual} - y_{predicted} \right| \tag{2-1}$$

Where $y_{actual}, y_{predicted}$ are the ground truth and the predicted value, respectively.

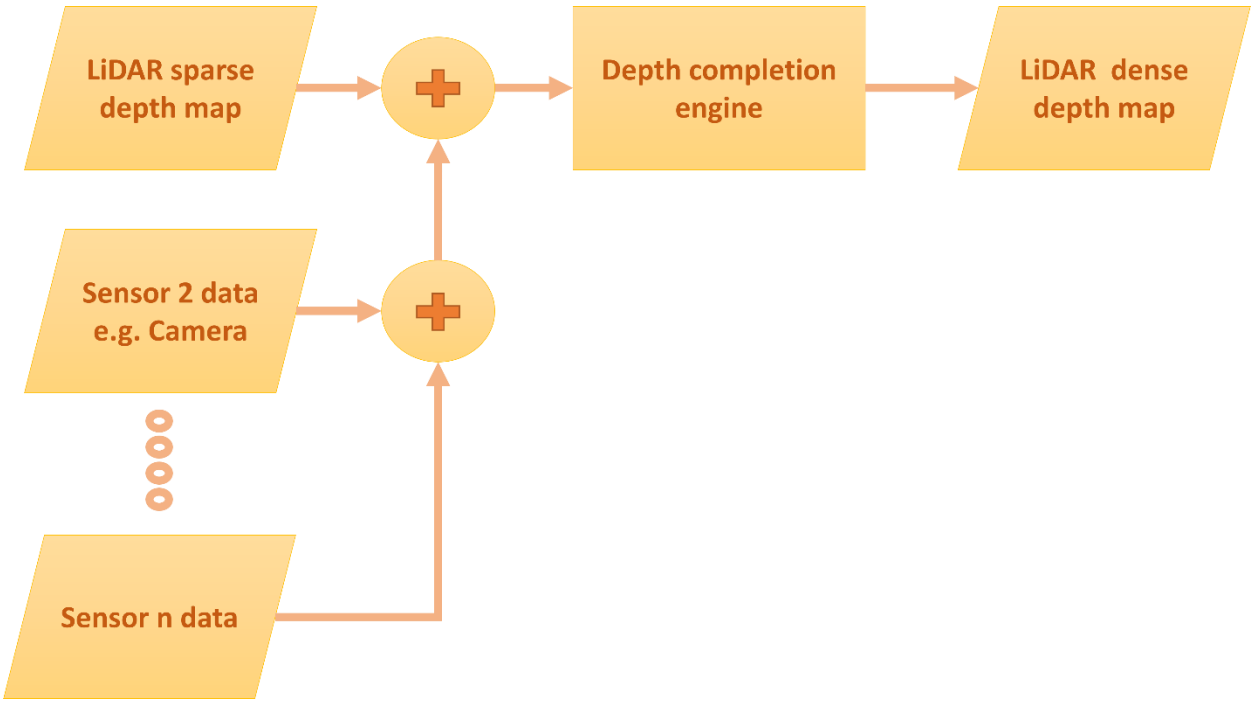

Figure 9. Simplified depiction of the guided depth completion framework [30]

Figure 10 depicts a general structure of an autoencoder deep neural network with skip connections. These connections can send the features map from the encoder layers directly to the corresponding decoder layers; this process is beneficial to provide the decoder layers with additional information that might be degraded during the encoding process. In addition, the authors

also proposed a self-supervised training framework that relies only on a sequence of RGB images and sparse depth images; that is, no ground truth is needed. The existence of nearby data is used to provide supervision signals. [33] provided two branches encoder/decoder framework with geometric encoding and multiple levels and modalities fusion. The authors fuse RGB and LiDAR data in one branch and fuse the generated semi-depth with the depth in another branch. Moreover, features are fused at different levels and between the two branches. [34] introduced DeepLiDAR framework, which consists of two separate pipelines, the first pipeline is used to generate surface normal from the sparse set of measurements, then the second pipeline is used to obtain a semi-depth map from RGB images. Finally, both surface normal as well as semi depth maps are fused and fed into another network which is trained to produce the final depth map. [35] proposed FusionNet, a two-branch framework, one branch for local features and the second for global features; for proper fusion. FusionNet generates a confidence map to fuse information from different branches adaptively. [36] suggest a more accurate sampling strategy and propose a deep neural network with a graph convolution module to overcome the limitations of the traditional square kernel.

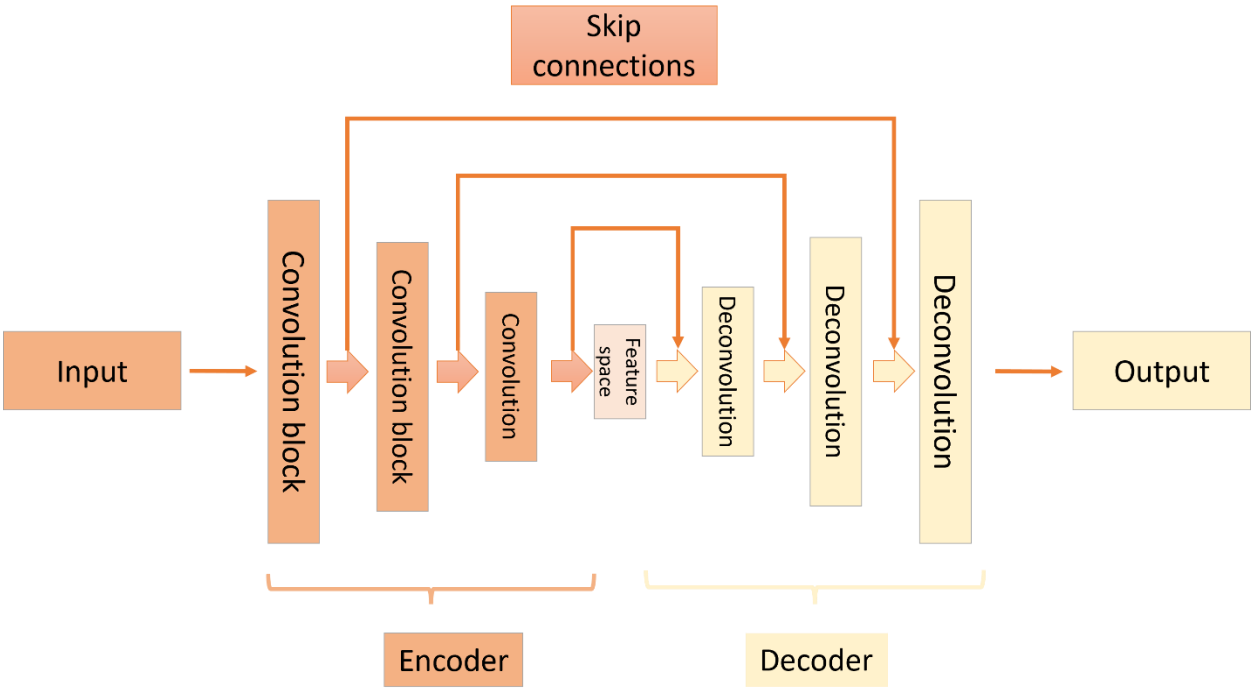

Figure 10. Autoencoder deep neural network with skip connections [30]

Object detection and classification

Milestones in Object detection and classification can be divided into two main categories: Conventional object detection and classification approaches and deep neural networks-based object detection and classification approaches. The first category piece together all approaches before the revolution of the deep neural network, while the second category conjoins all deep neural network-related approaches [24], [37].

a) Conventional object detection and classification approaches

Conventional object detection and classification approaches started emerging in 2001 and have depended on handcrafted features. In 2001, Viola and M. Jones [38] introduced the first real-time human faces detection approach, which has been generalized for different objects like cats

and dogs. The method is named the VJ detector and has three main techniques: integral image, feature selection, and detection cascades. Figure 11 shows the different shapes of the features used to detect faces. In 2005 Another critical milestone was the Histogram of Oriented Gradients (HOG) [39], which was considered a significant improvement for incorporating scale invariant feature transforms and shape contacts. In 2008, the Deformable Part-based Model (DPM) [40] was introduced based on the divide and conquer approach, wherein the training is considered the proper way for decomposing while the inference is regarded as an ensemble of detection.

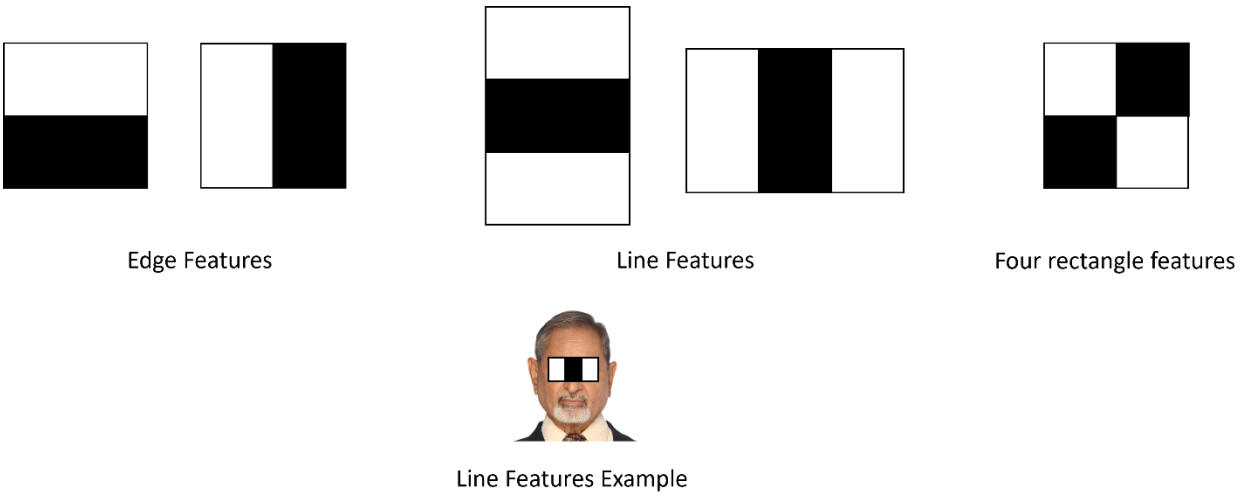

Figure 11. Viola and Jones features and usage example [38]

b) Deep neural-based object detection and classification approaches

Due to the recent advancements in computing capabilities and the tremendous improvements in machine learning and deep neural networks, object detection and classification have been developed and enhanced dramatically. Deep neural networks have been utilized to learn high-level features from images, and since 2012 many object detection and classification works have been published. Object detection and classification methods fall into two main groups: two-stage and single-stage detectors. Where the former relies on two stages to accomplish the task, one

for proposals generation and the other one for refinement. However, the latter performs the detection and classification in one single stage.

i. Two-stage object detection and classification

Great examples for the two-stage group are Region-Based Convolutional Neural (RCNN) [41], wherein a selective search technique is used to generate proposals regions. Then the proposals are fed into a CNN for feature learning. A Support Vector Machine (SVM) is used to predict the object's presence and class within the proposed region. Spatial Pyramid Pooling Networks (SPPNet) [42] advanced the field by introducing the Spatial Pyramid Pooling layer, which allows different input images or region sizes. Fast RCNN [43] allows for simultaneous detection and regression, Fasters RCNN [44] was introduced after Fast RCNN, and it is considered the first near real-time detector, which utilizes a separate Region Proposal Network which generates proposals with a minimal amount of computation resources. Feature Pyramid Networks (FPN) [45] were introduced after faster RCNN and had a top-down architecture to create high-level semantics at all scales.

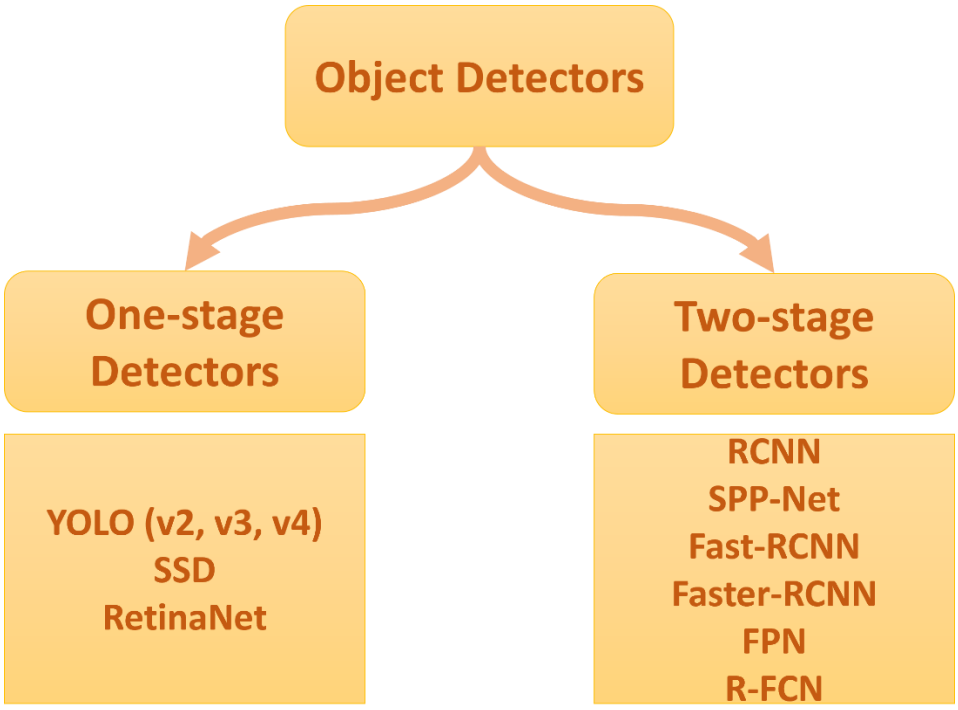

Figure 12. Summary of the available one-stage and two-stage object detectors [24]

ii. One stage object detection and classification

An excellent example of the one-stage group is: You Only Look Once (YOLO) (Redmon et al., 2016), which utilizes one network to localize and recognize objects within images. Other examples include: [46] introduced Single Shot MultiBox Detector (SSD), a multi-reference and multi-resolution detection technique to overcome the drawback of YOLO in detecting small objects. RetinaNet [47] is a single stage with a new loss function called focal loss, which was introduced and used to overcome the extravagant background and foreground instances imbalance.

Figure 12 shows a summary of the available object detectors, while Figure 13 highlights the main differences between one and two-stage object detectors.

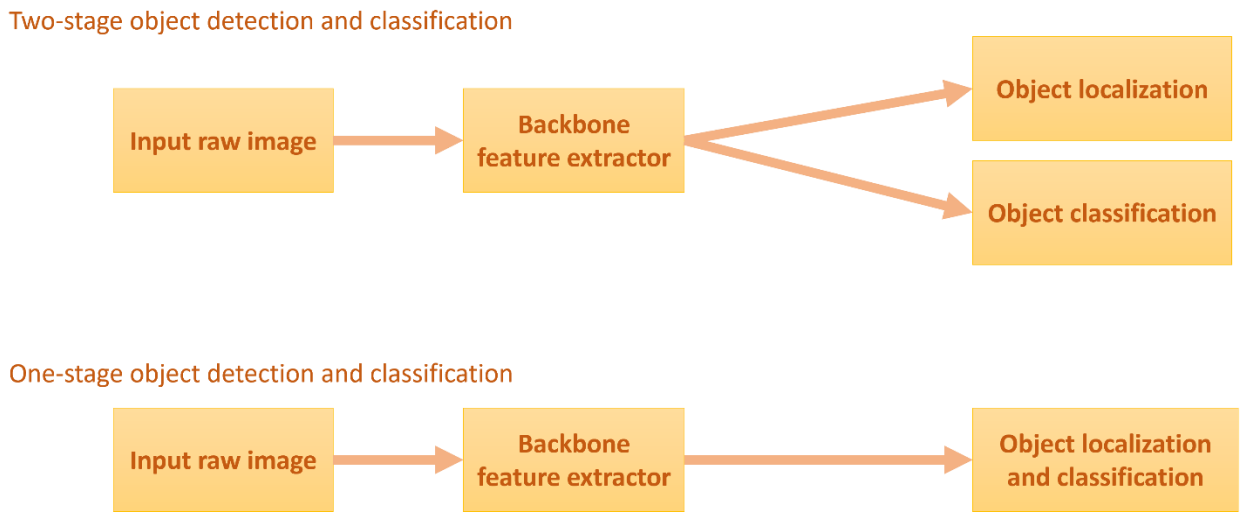

Figure 13. Object detectors architecture - Two (top) and one (bottom) stage object detectors [48]

Instance segmentation

Image segmentation is considered a fundamental problem in computer vision. It is an essential visual understanding element in many applications like medical image analysis, self-driving cars, and video surveillance. Under the Image segmentation umbrella, three main subfields are available: semantic segmentation, instance segmentation, and panoptic segmentation [49]. In semantic segmentation, image pixels are classified with semantic labels but without distinguishing between different objects within the scene. On the other hand, instance segmentation extends the concept of semantic segmentation by providing accurate object detection capabilities at a pixel level, thus providing a precise mask for each object within the scene. Therefore, the instance segmentation process solves two problems together: object detection and semantic segmentation. Panoptic segmentation combines the characteristics of both semantic and instance segmentation [50]. Figure 14shows the different segmentation techniques applied into randomly selected frames from the KITTI dataset.

Algorithms that tackle image segmentation task are numerous and can be grouped into two main categories: early non-deep learning methods like thresholding, region growing graph cuts, and active contours; and deep learning methods, which in recent years have produced impressive performance enhancements and a paradigm shift in image segmentation field.

Since 2014 many deep neural network-based image segmentation algorithms have been proposed, the most popular ones fall into two categories: two-stage and one-stage. In the two-stage category, algorithms perform two subtasks: detection and segmentation. Depending on the order of these

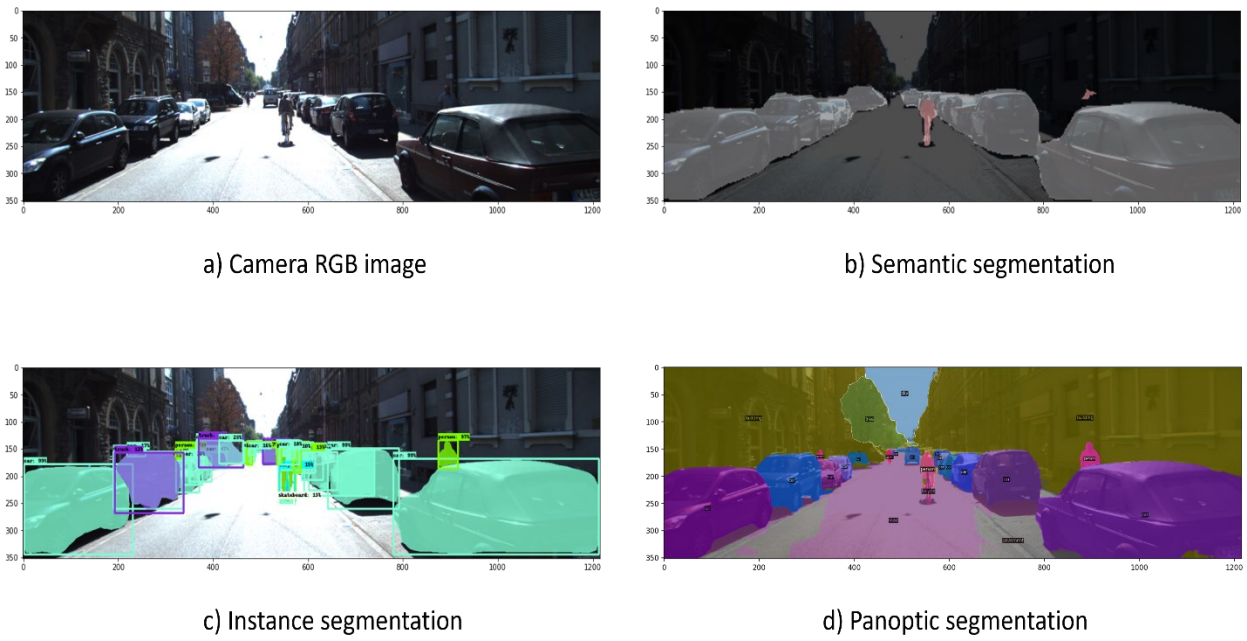

Figure 14. Different image segmentation techniques: a) reference RGB image b) semantic segmentation c) Instance segmentation d) Panoptic segmentation [16]

subtasks, two-stage instance segmentation can further be divided into two methods: top-down methods and bottom-up methods. The former is a detection-based instance segmentation method wherein top-level bounding boxes are first generated, then a foreground segmentation is conducted, while the latter is a segmentation-based method that starts with pixel-level segmentation and then uses clustering to group similar pixels. Implementations for top-down methods can be grouped into four categories: dense sliding windows, multi-level features, R-CNN, and contour information. Regional Convolutional Neural Networks (R-CNN) and its improved versions fast, and faster RCNN have achieved good performance. R-CNN algorithms detect objects by providing bounding boxes and class information; inspired by the R-CNN method, many instance segmentation methods have been proposed. For example, Mask R-CNN, which is based on the faster R-CNN architecture, instead of bounding boxes Mask R-CNN generates a high-quality segmentation mask while also detecting all objects in the scene [51].

In the one-stage category, algorithms consider the relationship between detection and segmentation within one DNN, this technique has faster performance, but the network is complex. Methods within this category are either anchor-based methods or anchor-free methods. Representative examples of the anchor-based methods are InstanceFCN, FCIS, TensorMask and YOLACT. On the other hand, representative anchor-free methods are SOLO, SOLOv2 and CenterMask [49], [51], [52]. For example, SOLO divides the camera image into multiple grids; these grids encode the relative position of the potential instances; the network consists of two main branches: the category branch, which is used to generate the semantic category, and the mask branch, which is used to predict the instance mask.

Autonomous Driving Datasets

Data is considered the fuel of deep neural networks and machine learning algorithms. Data plays a vital role in generating accurate nonbiased models. However, data collection is costly in terms of time and money and requires careful labeling and calibration, especially in autonomous driving applications, wherein multiple sensors are utilized and must be synchronized and logged carefully. Fortunately, many research and industrial companies are currently releasing their datasets for public use; some of these datasets focus on a single sensor, while others provide a bag of information from multiple sensors. Also, datasets are application specific, and some datasets are prepared for object detection, tracking, and classification, while others are for lane detection localization and behavioral analysis. A summary of the publicly available datasets that can be utilized for different applications, including instance segmentation, depth completion, object detection, and object classification tasks, is presented as follows:

- The Apollo open platform (Apollo, 2018) prepared by baidu Inc. and released in 2018, contains data from camera and LiDAR as well as vehicle data.

- Argoverse [53], prepared by Argo AI and released in 2019, data were collected in the USA and specifically in Miami and Pittsburgh; the data are for Camera, GPS, and LiDAR.

- Deep drive dataset [54] released by UC Berkeley. Dataset was collected in San Francesco Bay area and New York using a camera only but collected in diverse traffic conditions like urban, rural, and highways. It is also recorded in different daytime like night and non-night situations.

- Cam vid [55] is also a Camera only generated dataset recorded in the UK by the university of Cambridge in 2009.

- Cityscapes [56] is collaboration between multiple companies from multiple cities. The dataset mainly concerns vision algorithms, specifically semantic-related algorithms like scene labeling and object detection.

- Diamler de Ag, MPI-IS, and TU Darmstadt worked together to prepare the dataset recorded in Germany, Switzerland, and France. The dataset has information from the camera and other sensors. However, LiDAR was not used in this dataset.

- Elektra [57] was collected in Spain and Barcelona by the autonomous university of Barcelona and polytechnic university of Catalonia, and they only used a camera and infrared to collect the dataset.

- JAAD dataset [58] was collected in Ukraine and Canada in 2016 by York University, and it only utilizes Camera sensors.

- KAIST [17] multi-spectral dataset collected in 2015 by KAIST university, data collected from Camera, LiDAR, GPS, and Infrared for a scene in South Korea for both night and non-night scenarios and with a focus on pedestrians.

- KITTI (ccfv et al., 2013) Was released in 2011 and collected by Karlsruhe institute of technology and Toyota technological institute, the data were collected in Germany – Karlsruhe; Camera, LiDAR, and GPS are the primarily used sensors, and the dataset is suitable for a wide range of applications like object detection and classification(2D and 3D), and semantic segmentation and instance segmentation, and this dataset is the one that will be used in this dissertation and more information about it will be provided later in this section.

- NuScenes [59] dataset collected by Nu Tonomy Inc. and Aptiv for scenes in Boston, USA, and Singapore. It uses Camera, LiDAR, and GPS. It is recorded in different weather and light conditions.

- Udacity dataset [60] was released in 2016 for scenes in Mountain View, USA. The dataset was collected by different sensors like Camera, LiDAR, and GPS.

- Caltech Ped [61] is another camera-generated dataset collected by the California Institute of Technology in Los Angeles, USA, and released in 2009.

- NightOwls [62] was released in 2018 and collected by Oxford University, Max Planck Institute, and Continental Corp. The dataset was collected at different UK, Netherlands, and German sites.

- TUD-Brussels Ped [63] was collected in 2009 for pedestrian detection applications and was collected by Max Planck Institute for informatic in Belgium.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Clustering algorithms can fall into nine categories: partition, hierarchy, fuzzy theory, distribution, density, graph theory, grid, fractal theory, and model-based [64]. Density-based clustering algorithms are handy for data that are separable by density level, wherein clusters assemble data in dense regions with different shapes [65], [66]. This is a typical scenario in mobile robotics and SDCs wherein surrounding objects like cars, trucks, and pedestrians have different characteristics. Density-based spatial clustering of applications with noise (DBSCAN) [67] is a density-based clustering algorithm that is very appropriate for clustering in applications in which the number of clusters is unknown, data is noisy and data dimensions are high. DBSCAN is

controlled by two critical fixed, global parameters ε (epsilon), which is the point neighborhood search distance (also defined as a cluster radius), and minPts, which is the minimum number of neighboring points needed to form a cluster from a given point. The user selects these parameters' values, which may vary from one dataset to another and are applied globally to the entire dataset. Moreover, slight changes in these values will affect the clustering performance and a minute change may result in merging two or more neighboring objects into one cluster or producing some noise points as legitimate clusters.

DBSCAN defines core point, border point, and noise point expressions to distinguish the data points. A core point is a point in the dataset with minPts neighborhood points within the ε radius. A border point is a point within a core point neighborhood but has fewer points than the minPts within the ε radius. A noise point is a point that neither satisfies the core point definition nor the border point definition. Figure 15 shows the main point types.

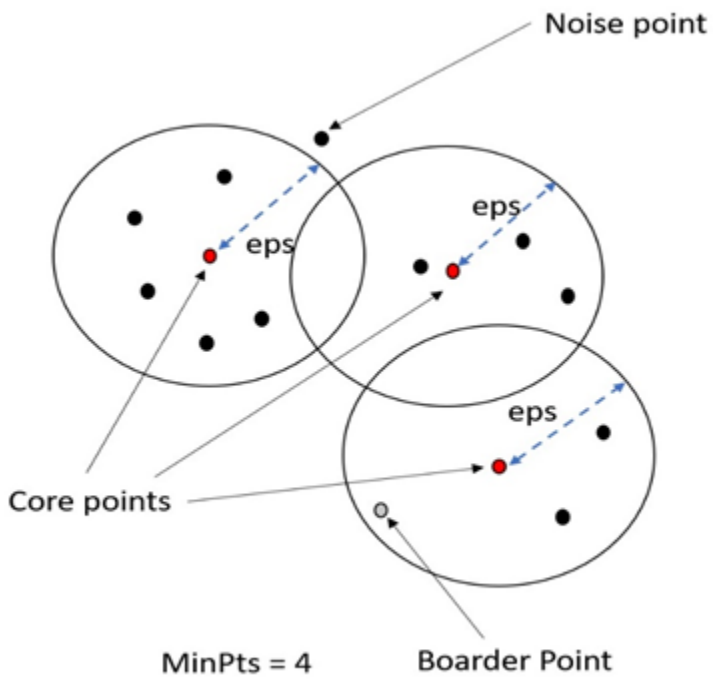

Figure 15. DBSCAN points types: red points are core points, the gray point is a border point, and the black point outside the circles is a noise point [67]

Important relations in DBSCAN concerning ε and minPts are the directly density-reachable, the density-reachable, and the density-connected relations. The directly density reachable relation can be defined as follows: for any two points p and q, a point p is directly density reachable from point q if p belongs to the q points neighborhood and q is a core point. However, density reachable relations relax the restriction in the directly density reachable relation by allowing for a chain of connectivity; that is, a point p is density reachable from point q if there is a chain of points where each consecutive points are directly density reachable. Finally, point p is density connected to point q if there is another point m which is density reachable from both p and q.

CHAPTER 3

RELATED WORK

This dissertation deals with the LiDAR point cloud depth completion and clustering. Many works have been done to improve the performance of these tasks, and solutions usually fall into two main categories: machine learning and deep neural network techniques. In the following sub-sections, related works to these topics will be discussed.

Depth Completion-Related Work

Deep neural networks-based depth completion

Many works have been published to improve the depth completion task for self-driving cars. [33] proposed strong networks for efficient depth completion, namely ENet and PENet; the backbone network consists of two main branches: color–dominant branch and depth dominant branch; the former accept both the LiDAR sparse depth map and the color Depth map but the branch focus on color information throughout an encoder/decoder like network, the later accept both the semi-dense depth map generated from the first branch alongside the sparse depth map. Both branches generate confidence arrays that combine the depth generated from each branch in an adaptive way. Extra refinement is also proposed using an improved implementation of [68] work. Although [33] work achieved outstanding results, the proposed work is very complex. It requires extensive computing resources (the GitHub repository mentioned 4 GPUs), and the training process is very long and could take several days.

Moreover, the results have been achieved after using around 86K annotated frames. [69] proposed a generative adversarial network (GAN) based for depth completion purposes. This work

considers the unguided depth completion problem wherein only sparse depth information is available. The final goal is to train a generator with a minimum loss and obtain the model parameters. Equation (3-1) show the main equation to solve.

$$\widehat{\theta_g} = \arg\min \frac{1}{N} \sum_{n=1}^{N} L^G \ (G_{\theta_G}(I_n^{SP}), I_n^{DN}) \tag{3-1}$$

where, L^G is the generator loss function, I_n^{SP} the training sparse depth samples, I_n^{DN} dense depth samples, θ_g are the CNN parameters to solve for the best minimum loss. It worth noting that L^G is a combination of different loss functions like adversarial loss, normal loss, and pixel loss. The network consists of three main parts: dense residual blocks, hourglass attention mechanism, and residual in residual dense blocks. [70] proposed a multitask generative adversarial network that works for both semantic segmentation and depth completion; they used the semantic segmentation output as input to improve the depth prediction accuracy. This work is guided as it uses the RGB images alongside the sparse LIDAR depth information. The authors introduced multi-scale pooling blocks within the network to extract features from different levels. The architecture has two main branches, one for semantic segmentation and the other for depth completion. Which yields two generators and two discriminators. [71] proposed a fully convolutional neural network for depth completion task; the network architecture is GAN and only utilizes sparse LiDAR information. The network comprises six convolutional layers, and ReLU follows all middle layers.

Image processing-based depth completion work

Work focusing on depth completion task using image processing is limited, [72] proposed a depth completion framework that depends entirely on image processing techniques and can run on CPU only. The framework consists mainly of several image processing operations like dilation, morphological operation, and filtering. At the time of publication, the performance of the proposed work was very promising; however, recently published work that utilizes deep neural networks and sensor fusion surpasses this image processing approach. [72] work utilizes LiDAR point cloud only. [73] also works on depth images super-resolution, wherein they focus on the 3D points and use repetitive structures to recognize similar patches throughout different levels.

Clustering Related Work

The related work discussion is divided into two main parts. The first part focuses on some promising generic DBSCAN improvements. In contrast, the second part focuses on the DBSCAN related works in the domain of mobile robotics and SDCs LiDAR data processing.

DBSCAN is a well-known density-based clustering technique that has been used in various fields and applications [74]–[78]. However, due to the DBSCAN limitations in handling sparse data and variable densities, many DBSCAN variants have been proposed to overcome these limitations and improve performance. Surveying all of these variants is beyond the scope of this work; however, some of the recent works that suggest ideas to automate the process of estimating DBSCAN tuning parameters are highlighted. In A-DBSCAN [79] Kth-Nearest Neighbor (K-NN) graph is used to select data core samples. Instead of ε and minPts, A-DBSCAN uses k values and

noise presence. They proposed ADBSCAN using three main steps; in the first step, the algorithm splits the dataset into disjointed subgraphs, then in the second step, it filters any subgraph that does not lie in the dense area; and finally, it assigns subgraphs into clusters where close subgraphs are assigned to the same cluster. In Approximate Adaptive DBSCAN [80] a quadtree-based density layer tree was used to partition the dataset equally without the need for an extra parameter or merging process. The same work also uses (K-NN) to further improve the AADBSCAN by solving the static ε drawback of AADBSCAN. [81] proposed a generic Adaptive DBSCAN which first starts with a random value for ε and then evaluates the performance; if no clusters are detected, then it increases ε by 0.5, and this process will repeat until 95% of the data has been exhausted. The proposed method requires an assignment for the expected number of clusters which is not practical in many applications. [82] proposed method to estimate the value of ε and minPts, the method requires another parameter K to draw K-dist graph required to find the knees that corresponds with the density changes. Knees also used to estimate a set of ε values which then used to estimate the minPts.

Although DBSCAN is widely utilized in a wide range of clustering applications, minimal work has been done to automatically and adaptively estimate DBSCAN tuning parameters within applications like robotics and self-driving cars. [83] proposed an estimation approach based on the average K-NN. Their process consisted of the following steps: 1) it normalizes the data, and this step is only needed if the LiDAR data combines both coordinates and color information, 2) it builds a spatial index and uses both the spatial index and different distances calculations it estimates the clustering parameters, and 3) clustering takes place using the different estimated ε. [84] proposed an Adaptive Searching-DBSCAN which starts with initial values and is updated in each clustering iteration. The authors suggested equations to estimate both initial and subsequent values. The

proposed equations consider different parameters like expected target width and number of points per location. However, none of the equations considered the location of the LiDAR nor the relative distance of the points to the LiDAR.

[85] uses the infrastructure LiDARs to detect and track pedestrians at intersections, to improve DBSCAN and automatically select the tuning parameters, they took into consideration the properties of the installed LiDAR. They introduced two equations to find both the minimum horizontal and minimum vertical distance between LiDAR points; also, three zones division scheme was introduced to change the values of the tuning parameters based on the object's locations.

CHAPTER 4

METHODOLOGY

This work focusses on the perception module of self-driving vehicles and utilizes information from two sensor modalities: LiDAR and Camera. The first part of the thesis addresses the depth completion problem wherein missing depth information in the sparse LiDAR point cloud data is estimated. An instance segmentation-based guided depth completion method is proposed that utilizes both LiDAR point cloud and raw camera images. The second part of this thesis addresses the clustering of the LiDAR point cloud problem. A density-based clustering algorithm is investigated and enhanced by adding an adaptive and automated technique to estimate clustering parameters.

Instance Segmentation-Based Depth Completion Framework Using Sensor Fusion

Depth completion aims to obtain dense depth maps from sparse depth measurements. It is a critical component in many vision applications and has rapidly growing significance for autonomous driving. Accurate depth is essential for many perception tasks such as 3D object detection and shape recognition, 3D mapping, and localization. These tasks are at the heart of many applications, including augmented reality, SDCs, and robotics. Due to the limited detection range, environmental interference, and cost considerations, frequently used depth sensors such as LiDAR, RGB-D sensor cameras, and Time-of-Flight (TOF) cameras produce sparse depth measurements. For example, the top-of-the-line Velodyne HDL- 64E LiDAR sensor costs around $75,000 but can only deliver sparse data with vertical resolution/angular resolution of 0.4/0.08 [86]. On the other hand, dense depth maps are required in many high-level applications such as

3D object detection, 3D scene reconstruction, and simultaneous localization and mapping (SLAM). Depth completion or generating dense depth maps from sparse depth data has become a popular technique to bridge the gap between sparse and dense depth maps.

Depth completion techniques fill in the gaps in a sparse depth map either without or with the help of a reference image. The latter uses structure information from the guidance image to improve performance, attracting increased study attention. However, the input depth map is irregular, sparse, and noisy. Moreover, the color image and the depth map are from two different sensor modalities; therefore, the image-guided depth completion task has unique challenges. As a result; various sparse invariant convolutions, uncertainty exploration, and multi-modality fusion algorithms have been developed to overcome these challenges. In addition, several modern approaches use multi-scale characteristics, surface normal, semantic information, or context affinity to further increase performance [87].

Most of the recent work in the domain of depth completion is deep neural networks based; although the current state of the art achieved very good performance, the generated depth at object boundaries or for distant objects still suffers from inaccurate depth values. In addition, none of the previous works provided a customized approach to completing selected objects instead of the entire scene. Many factors can degrade depth completion process performance at objects boundaries or for distant objects, such as the lack of sufficient point cloud and when some objects are very close or even occluding each other. Figure 16 shows selected images from the KITTI dataset [16] wherein objects (cars) close and occlude each other.

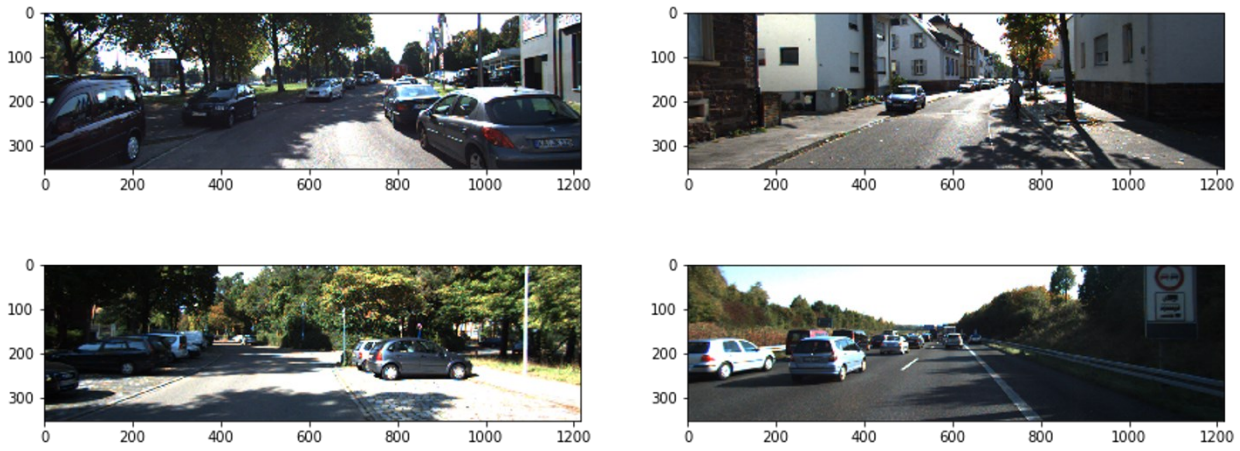

Figure 16. Selected images from KITTI dataset showing scenarios where objects like cars are very close to each other or occluding each other [16]

Most of the recent depth completion work [33], [68], [70], [88] applies algorithms to the entire sensors data without paying attention to the object's boundaries and object's types. Accurate objects boundaries and objects types are valuable features and can guide depth completion frameworks more accurately. In the last few years, instance segmentation has gained huge interest from the industry and academia, because it combines both object detection and segmentation capabilities in a pixel level accuracy, and provides different unique scene insights like the number of objects, pixel-level object boundaries, type of object and distinguish between objects that fall into the same type; making it an up-and-coming technology to guide the depth completion process. Using Instance segmentation in depth completion has many benefits, for example:

- Accurate pixel-level object boundaries.
- Accurate object detection, especially for far objects.
- Accurate object classification.

Using instance segmentation in depth completion will allow for objects level depth completion, thus performing the depth completion for the objects of interest instead of depth completion on the entire scene.

In this dissertation, an object-based depth completion framework is proposed and implemented. Figure *17* shows a high-level presentation of the implemented framework.

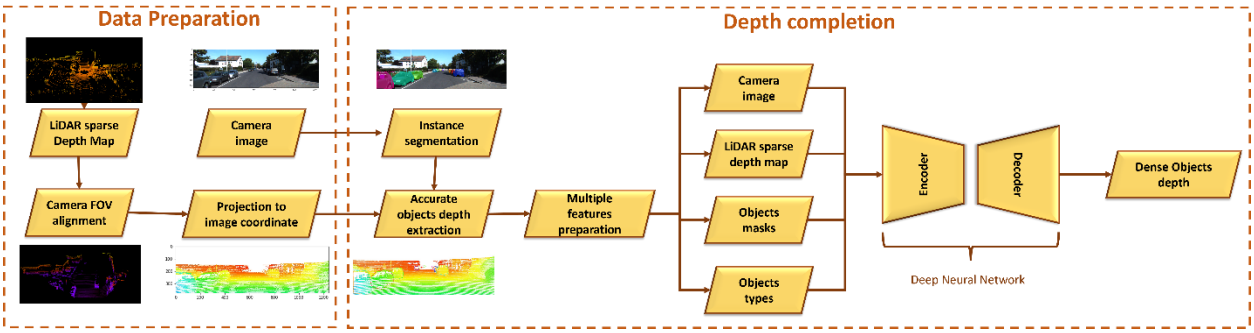

Figure *17*. Proposed Instance Segmentation-based Depth Completion Framework

The framework consists of the following steps:

1- Input Data Preparation

The proposed framework uses two leading sensing technologies: remote LiDAR sensing and optical image sensing (Camera). These modalities are from different coordinate systems and have a non-identical field of view. For example, the used LiDAR device can capture 360° of the surrounding environment with a 26.9° vertical field of view, 0.08° angular resolution (azimuth), and around 0.4° vertical resolution [89]. On the other hand, the camera sensor can only sense the vehicle's front view. Therefore, the LiDAR point cloud is cropped to match the camera field of view and projected from the LiDAR coordinate system into the camera coordinate system. Figure

18 shows the raw LiDAR point cloud, the cropped LiDAR point cloud, and the LiDAR point cloud projected into the camera 2D coordinate system. It is worth mentioning that LiDAR points in Figure 18-d are colorless but colored and enlarged for better visualization.

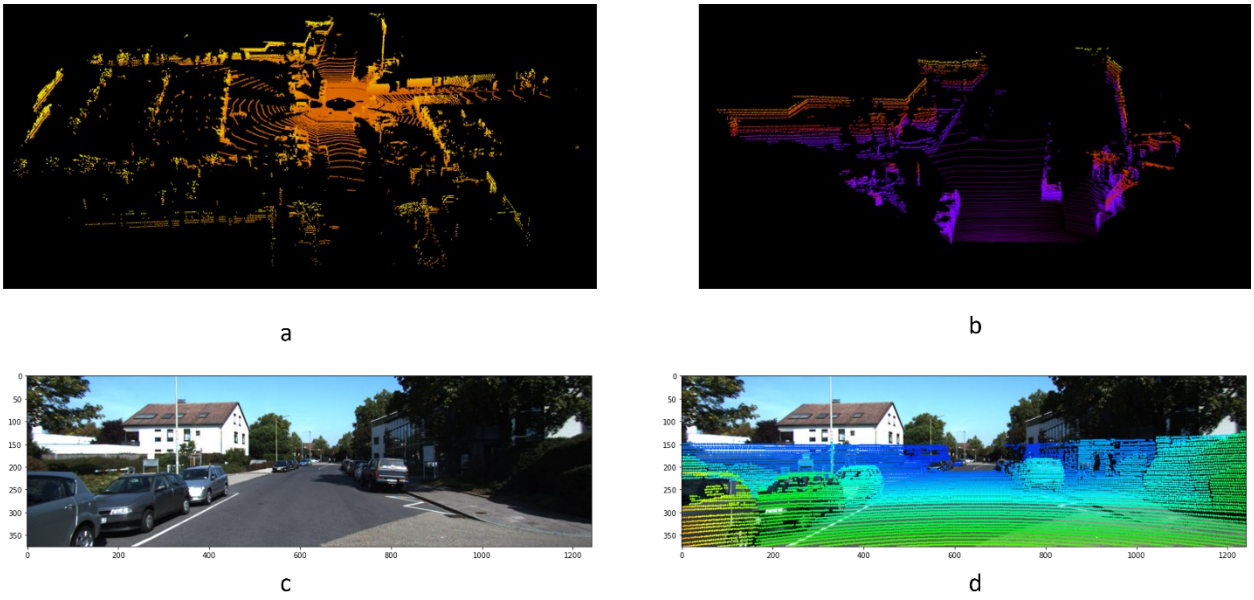

Figure 18. LiDAR point cloud in different FOV and projection a) LiDAR raw point cloud with 360° field of view b) LiDAR point cloud cropped to match camera sensor field of view c) camera sensor reference image d) LiDAR point cloud projected into optical sensor 2D coordinate system and color-coded (cold color for far objects while warm colors for near objects) [16]

To project a 3D point $x = (x, y, z, I)^T$ from the LiDAR coordinate system into a 2D point $y = (u, v, I)^T$ in the i'th camera sensor coordinates system, equation (4-1) [16] is used.

$$\boldsymbol{y} = P^i_{rect}\boldsymbol{x} \tag{4-1}$$

Where $\boldsymbol{P^i_{rect}}$ is the rectifying rotation matrix, and I is the intensity of the LiDAR point.

2- Instance Segmentation

For Instance Segmentation, the Mask R-CNN architecture has been selected. Although there are many other alternatives in the literature, Mask R-CNN is unique for different reasons. First, it is based on a stable and supported R-CNN object detection architecture. Second, Mask R-CNN provides mask information and inherits information from the R-CNN like object bounding boxes, detection scores, and object types. Finally, Mask R-CNN is common and has been trained on many big generic datasets like COCO [90], which make the process of generalizing it to other tasks faster. Mask R-CNN is a two-stage instance segmentation framework; the first stage is used to scan the input image and generate proposals which are objects potential areas. The second stage performs object classification, bounding boxes regression, and mask generation. Figure 19 depicts a high-level representation of the Mask R-CNN instance segmentation framework.

Mask R-CNN uses a backbone network to extract features wherein an RGB camera image is converted into a feature map and further feed into a feature pyramid network (FPN). Deep neural network computer vision literature has many strong feature extraction backbones like AlexNet [91], VGG-16 [92] GoogLeNet [93] and ResNets [94],

The resulting feature map is forwarded into a region proposal network (RPN). The primary purpose of this network is to find areas in the feature map that contain potential objects. RPN uses a sliding window technique that scans anchors with different sizes and aspect ratios. For each anchor, the RPN generates two outputs: anchor class and a bounding box. The anchor class classifies each box into foreground or background, where foreground means a potential object exists. The content of each proposed region is classified, and an accurate bounding box for each classified object is regressed.

It is worth mentioning that all of the previous operations belong to the faster R-CNN architecture [43], but the next step is the unique part of Mask R-CNN [95], wherein a segmentation mask is generated in parallel with the classification and bounding boxes regression operation. Figure 19 depicts the instance segmentation process.

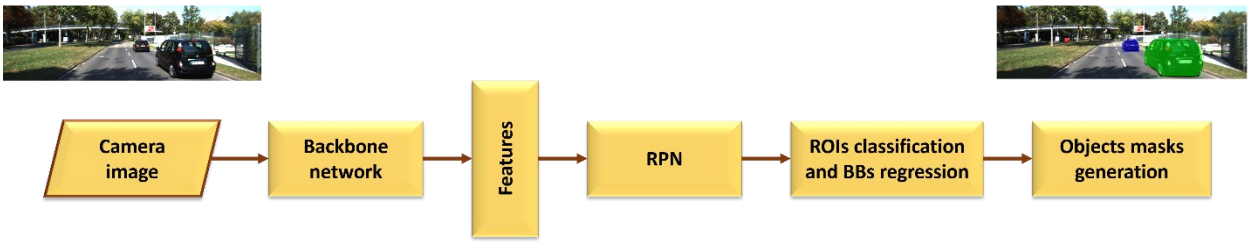

Figure 19. Instance segmentation main processes [95]

Mask R-CNN is already pre-trained on a wide range of big public datasets like COCO [90]. However, to make it more suitable for the self-driving domain, a transfer learning approach has been adopted to add more data from the KITTI instance segmentation dataset [16] while preserving knowledge from the COCO dataset [90]. Transfer learning is the ability to transfer already acquired knowledge from a previous task to a new related task. This approach is prevalent in image knowledge-based tasks due to the underlying shared initial processes like detecting edges, colors, and brightness, among others. Figure 20 depicts the concept of transfer learning; task 2 is similar to task 1; e.g., task 1 is to detect people in a generic setting, while task 2 is to detect cyclists on the road. Task 1 dataset is usually bigger than task 2, contains more genetic information, and requires more training time; on the other hand, task 2 dataset is generally smaller than task 1 and requires less training time.

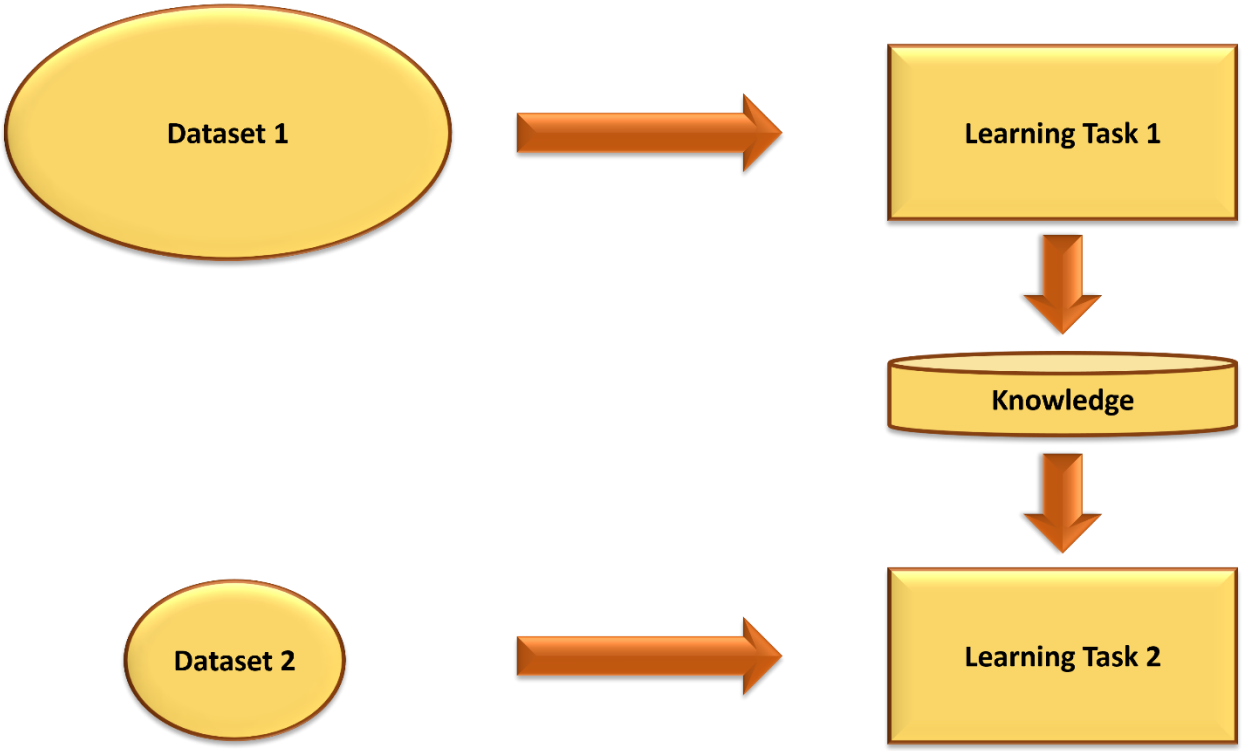

Figure 20. Transfer learning concept, knowledge from task 1 used to facilitate task 2 [96]

Transfer learning can be performed in two ways. The first way is to allow all model layers to update and adjust weights based on the new training dataset. The second way is to freeze all layers except the last layers (heads) and only train and adjust the final layers [96]. The second approach is adopted in this dissertation since the similarity between the two tasks is very big, and thus the previously learned weights are reasonable to preserve. During training, the original Mask R-CNN loss functions have been used, which are: classification loss, bounding box regression loss, Mask loss (binary cross-entropy loss), and combined loss [95], which are formulated in equations (4-2), (4-3), (4-4) and (4-14), respectively.

$$L_{cls}(p_i, \hat{p}_i) = -log\ (p_i^c) \tag{4-2}$$

Where $\hat{p}$ is the predicted probability for the proposed region belonging to class c.

$$L_{box} = l_1^{smooth}(t_i - \hat{t}_i) \tag{4-3}$$

Where t_i , $\hat{t}_i$ are the ground truth coordinate of the bounding box and the predicted coordinate, and l_1^{smooth} is a combination of Least Absolute Deviations and Least Square Errors.

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \, log \, \hat{y}_{ij}^k + (i - y_{ij}) log(1 - \hat{y}_{ij}^k) \right] \tag{4-4}$$

$where \; y_{ij}$ is the groundtruth lable of pixel at i, j location $and \; \hat{y}_{ij}^k$ is the predicted label of pixel at i, j.

$$Combined \; loss = L_{cls} + L_{box} + L_{mask} \tag{4-5}$$

The output of the instance segmentation step contains the following meta information:

- Objects ids: a list contains all detected objects in the given input image: {3, 3, 3, 3, 2, 1} where here these number corresponds to different objects, e.g., cars, pedestrians, or cyclists.
- Detection scores: a list contains confidence scores for all detected objects: {0.99, 0.97, 0.99, 0.91, 0.85, 0.25}.
- Region of Interest ROI array: an array consists of the dimensions of bounding boxes for each detected object. [[Object 1 ROI], [Object 2 ROI], …, Object n ROI]
- Masks array: an array consists of the object's mask. Pixels belonging to a legitimate object are assigned a value of 1, while pixels outside a legitimate object are assigned a value of 0.

Figure 21. Instance segmentation example that applied into a randomly selected KITTI frame with all results' attributes shown; each object has a type, confidence score, bounding box, and mask [16]

An Example of all information resulting from the instance segmentation module is provided in Figure 21; each object has a class name, e.g., car, score, e.g., 0.95, bounding box (dashed rectangle), and mask (the overlayed color on each object).

Instance segmentation output is huge, especially the instance segmentation masks; because it increases the storage requirement dramatically (if a 1D mask array needs 1 MB of storage, then an image with five objects needs 5 MB). Therefore, we designed a data structure and encoding scheme to solve this problem that dramatically reduced storage requirements. The proposed technique embeds all essential information into one array consisting of object ids, scores, and object indexes within the mask array. The proposed technique is presented in Algorithm 1 which describes the main operation of this encoding scheme. This algorithm will generate a small footprint of the needed instance segmentation information. In training, instance segmentation information is prepared in the shape of a 2D image where pixels corresponding to potential objects are assigned the object class id value. In contrast, other pixels will have a zero value. Figure 22

depicts the encoded instance segmentation feature masks and object types into a single 2D 1-channel array.

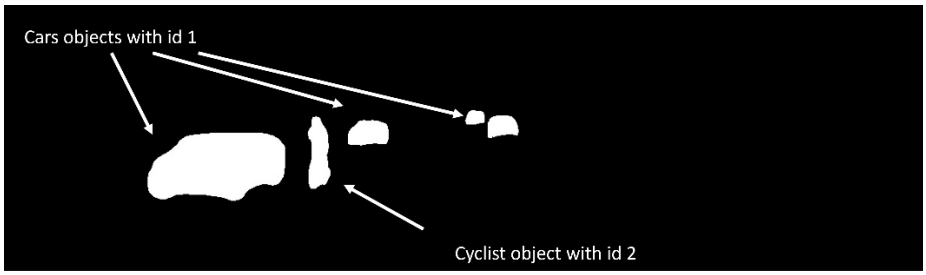

Figure 22. Instance segmentation objects' masks and types encoded into a single 2D 1-channel array

ALGORITHM 1: ALGORITHM TO COMPRESS INSTANCE SEGMENTATION MASKS AND OBJECTS TYPE

```
Function    Compress instance(i)

Input       i: array - RGB image with size (w,h,c)
Output      r: array with size (w,h,1)

1:          //calling instance segmentation model
2:          R = instance_segmentation_Detect(i)

3:          //Create empty arrays to hold the output
4:          Fram_instance_seg_info = []
5:          masks_idx = []

6:          //append all class ids to the output array
7:          fram_instance_seg_info.append(R['class_ids'])
8:          //append all detection scores to the output array
9:          fram_instance_seg_info.append(R['scores'])

10:         //loop over all detected objects
11:         for j in range(len(R['class_ids'])):
12:             obtain indexes of the pixels belonging to objects
13:             mask_idx = np.where(R['masks'] [j] == True)

14:             //append indexes to the array
15:             masks_idx.append(mask_idx)


16:         //append the full masks indexes array to the output array
17:         fram_instance_seg_info.append(masks_idx)

18:         return fram_instance_seg_info
```

3- Depth Completion

Depth completion deals with the problem of estimating a dense set of depth measurements from a sparse input. Let us assume the dense output is D and the sparse input S, then D can be estimated using a network N with parameters θ as formulated in equation (4-6).

$$D = N(S, \theta) \tag{4-6}$$

Equation (4-6) applies to non-guided depth completion problems where only a sparse LiDAR point cloud is used. However, accurate dense depth maps can be obtained by combining multi-sensor information. A commonly used method is fusing data from RGB cameras and LiDAR in early or late fusion mechanisms. This technique is formulated in equation (4-7) for RGB image I:

$$D = N(S, I, \theta) \tag{4-7}$$

In this research, we updated this formula and added another exciting piece of infor-mation to improve the guided depth completion performance. Instance segmentation pipelines provided useful information like object types, objects locations in a bounding box format, and pixel-level mask for detected objects. This information is very beneficial in guiding the DNN. Thus we revisited equation (4-7) and included the instance segmentation information: object type and object mask, as in equation (4-8):

$$D = N(S, I, M, T, \theta) \tag{4-8}$$

Where M and T represent the object's pixel level mask, and object types, respectively.

The parameter θ is optimized during the training by minimizing the loss function given a ground truth sample gt as in equation (4-9)

$$\hat{\theta} = \text{argmin}\, \mathcal{L}\,(D, \text{gt}) \tag{4-9}$$

Instance segmentation will generate accurate information about all objects of interest within the camera field of view. The generated segmentation mask and the classification results of the objects of interests will be used to accuretyly guided the depth estamtion network.

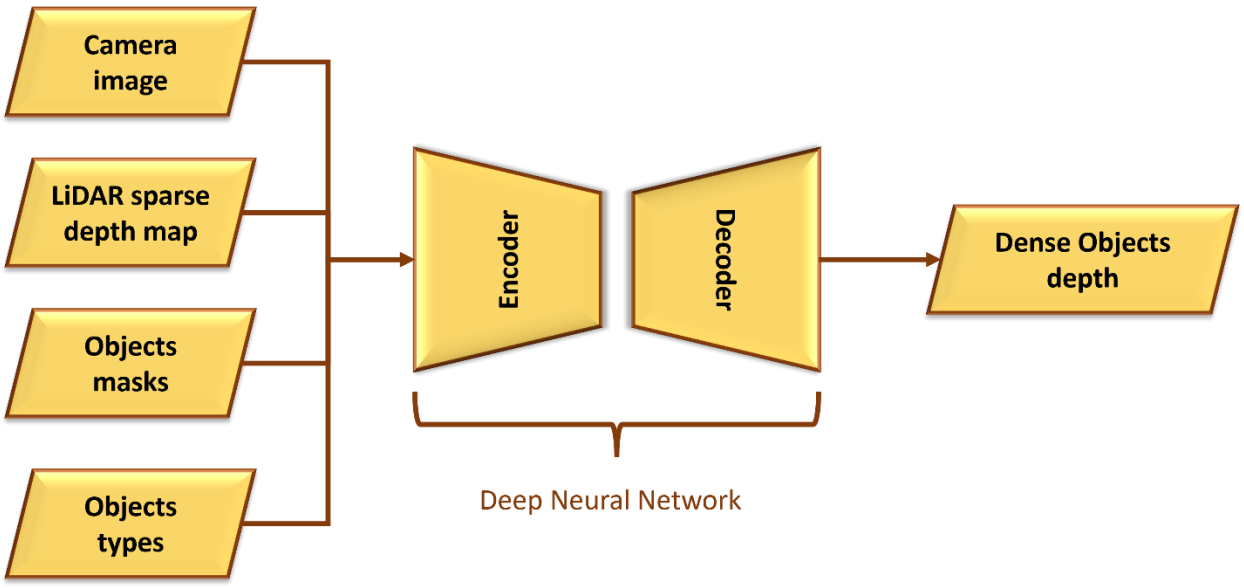

Figure 23. High-level presentation of the proposed depth completion DNN with the inputs and output

Figure 23 depicts a high-level presentation of the depth completion DNN. The DNN will accept the depth information, object types, object boundaries (masks), and RGB color information and learn through a convolutional layer organized in an encoder-decoder-like framework to predict the depth of the missing pixels for each object. The input for the network is an array of five

channels where the first three channels represent the RGB information, the fourth channel holds the depth information, and the fifth channel contains the mask and object type.

Table 2. Example of, the shape of the input of the depth DNN, please be informed that these entries are not adjacent and are selected for illustration purposes only

Pixel number	RGB camera information			Depth information	Mask and object type
	R	G	B	D (m)	Mask_object
0	120	115	33	10	3
1	110	100	30	0	3
2	200	120	100	25	3
3	30	200	110	70	2
4	60	90	130	3	0
5	55	33	20	1	0
6	40	120	200	1.5	2

Table 2 shows an example of the shape of the input for the depth DNN. For example, the second pixel has the value of (110, 100, 30) for the color information and 0 for the depth, and 3 for the mask_object (3 represents the class of the car) channel, which means this entry belongs to

a legitimate point but miss the depth value. In contrast, pixel numbers 4 and 5 belong to pixels outside objects of interest because mask_object is 0 (0 represents background).

a. Network architecture

To evaluate the effectiveness of instance segmentation in guiding the depth completion process, a two-branch convolutional neural network backbone similar to [33], [88] is used. The network consists of two branches; the first accepts three different input information: RGB image, depth map, and instance segmentation mask and objects of interest types. Each branch implements an encoder-decoder network with symmetric skip connections. Autoencoder architecture is a well-known DNN architecture with a proven performance record in solving noise reduction and data improvement problems[97]. The depth completion process aims to fill in the gap between the sparse LiDAR point cloud points by estimating this missing information from the available sparse information. The encoder consists of one convolutional layer and ten basic residual blocks [94]. The decoder comprises five deconvolutional layers and one convolutional layer. Each convolutional layer is followed by batch normalization and relu activation layers. In this network, information is fused in different levels. Initially, input data is fused following an early fusion approach, and then another step of fusion is also conducted between two branches. Features from the first branch's decoder part are also fused into the encoder part of the second branch. A final late fusion process is also conducted by fusing the two semi depth maps into a final dense depth with the help of confidence values generated by each branch. The fused depth map $\widehat{D}_{Fused}(u, v)$ is calculated unising equation (4-10). Figure 24 provided more details about the network architecture.

$$\widehat{D}_{Fused}(u,v) = \frac{e^{C_{B1}(u,v)}.\widehat{D}_{B1}(u,v) + e^{C_{B2}(u,v)}.\widehat{D}_{B2}(u,v)}{e^{C_{B1}(u,v)} + e^{C_{B2}(u,v)}} \quad (4\text{-}10)$$

Where $C_{B1}(u,v)$, $C_{B1}(u,v)$, $\widehat{D}_{B2}(u,v)$, $\widehat{D}_{B2}(u,v)$ represent the confidence map from the first branch, the confidence map from the second branch, the estimated depth from the first branch, and the estimated depth from the second branch, respectively. (u,v) represents the pixel location.

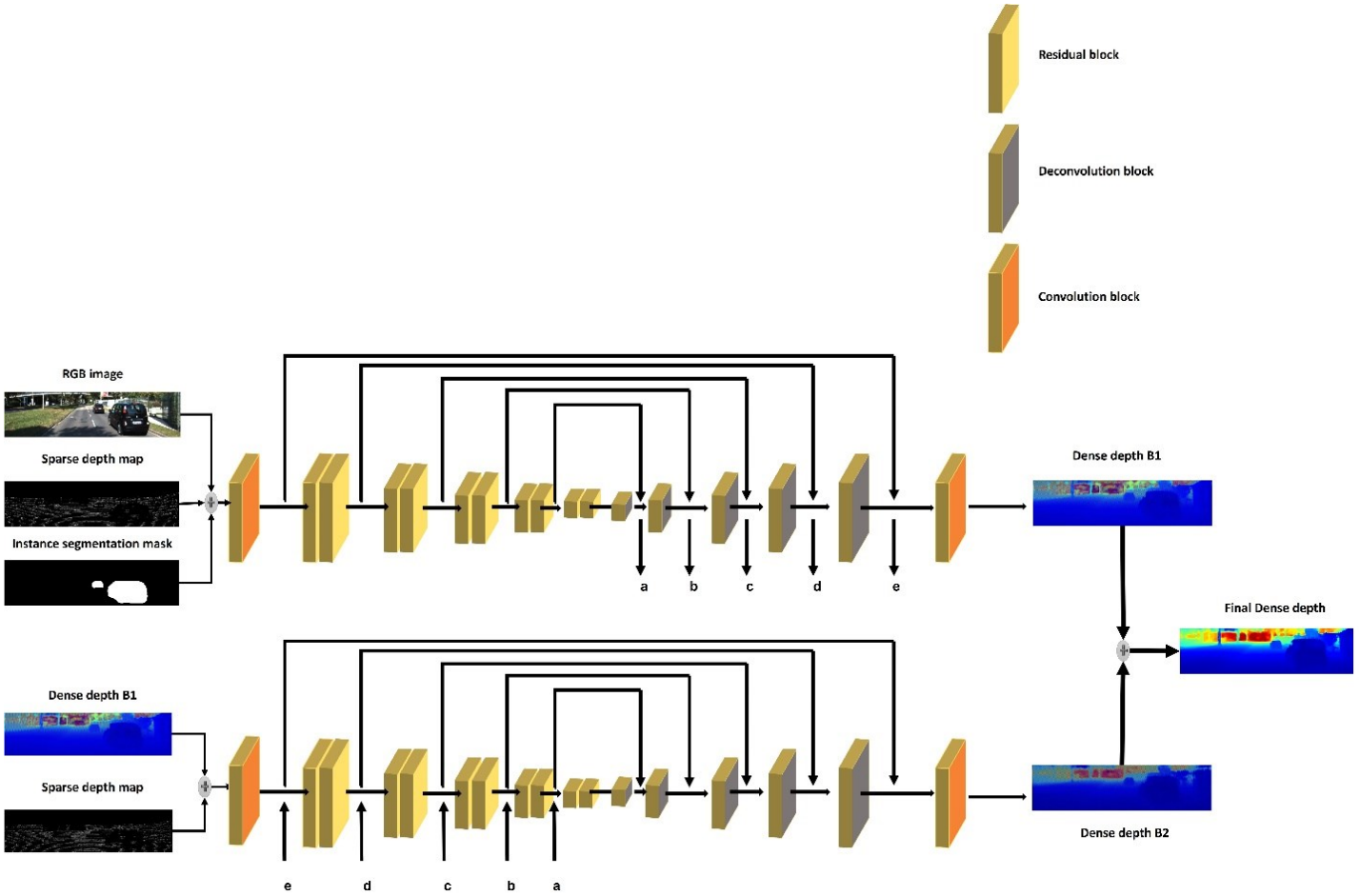

Figure 24. Proposed network architecture, the top part depicts the first branch, which takes the instance segmentation as input, while the lower part represents the second branch which takes branch one depth and the sparse depth as an input

A DBSCAN LiDAR Point Cloud Clustering Framework

Light Detection and Ranging (LiDAR) technology is used in applications like urban road detection, planning, robotics, and Self-Driving Cars (SDCs). In robotics and SDCs, LiDAR is used for environment perception and understanding [98], [99]. However, LiDAR data is unordered and sparse. Grouping LiDAR data based on some criterion benefits many LiDAR-based real-time applications like object detection and classification [100],[83], [101].

DBSCAN clustering was built based on the assumption that the data has a consistent density; thus, using fixed global parameters seems appropriate. However, density is not uniform in real-world scenarios, so setting these parameters to constants should be addressed. The autonomous vehicle is immersed in a very dynamic environment, and the data from the LiDAR is very sparse and non-uniform. Accurate object detection in a LiDAR sweep demands an adaptive estimation of both ε and minPts.

In this work, I propose to model the system's dynamics and automatically and adaptively estimate the DBSCAN parameters ε and minPts using empirical equations and the FOV division scheme.

- **Foreground Points Extraction**

Extracting foreground points from the entire LiDAR point cloud is vital to the success of any LiDAR data clustering process; it reduces the number of points dramatically, thus simplifying the clustering task and making it much faster. To distinguish LiDAR foreground points from background points, I followed the same approach of [102] in removing background points and reducing the total number of points using DNN. That is, all LiDAR 3D point clouds are fed into

encoder/decoder DNN to learn discriminative features, saving the resulting features into a vector, and then fed to a classification head; the classification head classify the points into two groups: foreground and background points, that is label: '0' for background points and label: '1' for foreground points. For training, the ground truth objects bounding boxes within KITTI have been used to distinguish foreground points from background points. Using the following assumption: All points inside the ground truth bounding boxes are considered foreground points, and the remaining points are considered background points. The same parameters proposed by (Shi et al., 2018) have been adopted for training. Figure 25 shows the foreground segmentation framework where the output of the segmentation head is either foreground points (orange points) or background points (gray points).

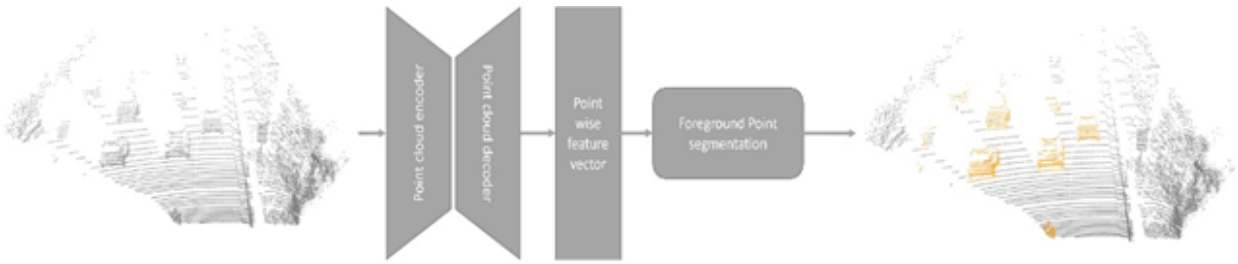

Figure 25. Foreground segmentation framework, the final output on the right shows orange points as the foreground, while gray points are the background [16]

Additionally, Figure 26 provides another example of the segmentation process. Figure 20-a is the original point cloud, while Figure 26-b shows the resulting foreground points from the camera FOV perspective bounded by the white lines. The segmentation process marks all irrelevant points as a background, such as trees, walls, and road infrastructure.

As shown in Figure 26, the camera FOV was used as a reference. It is worth noting that the original LiDAR data size in Figure 26-a is 18,876 points. After background removal, it drops to 5,158 points, which is almost a 75% reduction, which is very beneficial to any clustering algorithm.

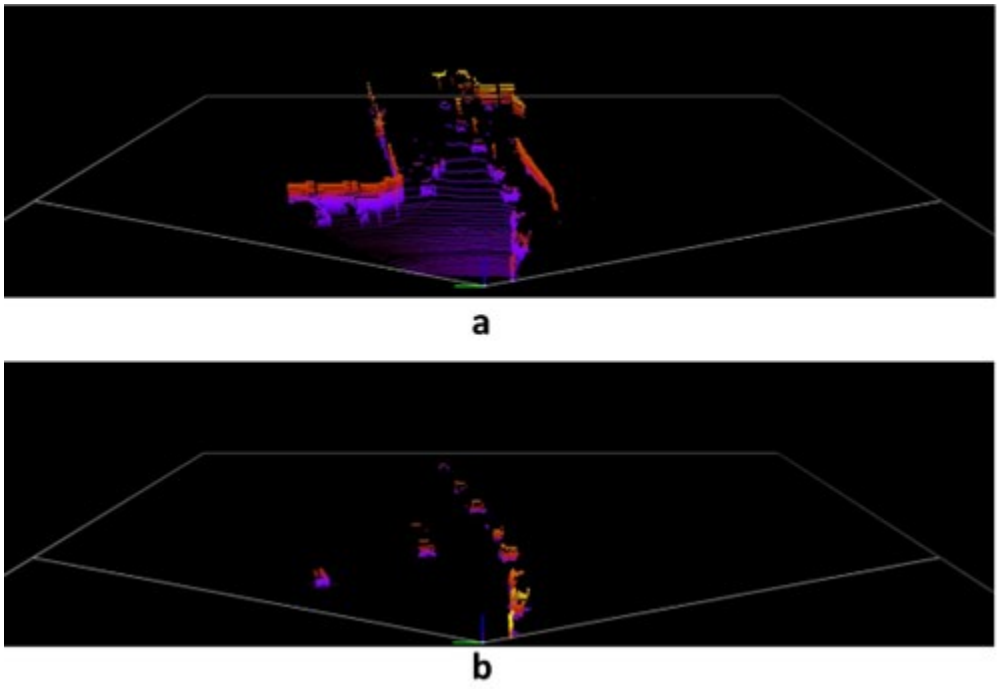

Figure 26. a) Raw 3D LiDAR point cloud b) Foreground point cloud [16]

- **Adaptive Density and Ego Vehicle Location Aware DBSCAN (ADEVLA-DBSCAN)**

DBSCAN can be applied to LiDAR point cloud clustering, but using global fixed tuning parameters ε and minPts will negatively impact the overall performance. They may cause it to be unusable in real-time critical applications like SDCs and mobile robotics. Different clustering simulation experiments have been performed using DBSCAN with different tuning parameters to elaborate more on this. And visually and manually examined the results. Figure 27 shows the effect

of using different ε and minPts on the same LiDAR sweep, a wide range of ε and minPts, has been used. However, the values shown in Figure 27 are (0.5, 1, 1.5, 2) and (9, 8,15) for ε and minPts, respectively. ε has a significant impact on the final points per cluster; for example, in Figure 27-b and Figure 27-e, changing the ε value from 0.5 to 2.0 allowed DBSCAN to detect and cluster the far object pointed by the white arrow, however, due to this change in ε, extra noise added to the orange cluster and two objects have merged together as a single object. Similar scenarios also exist in Figure 27- a, d, c, and f.

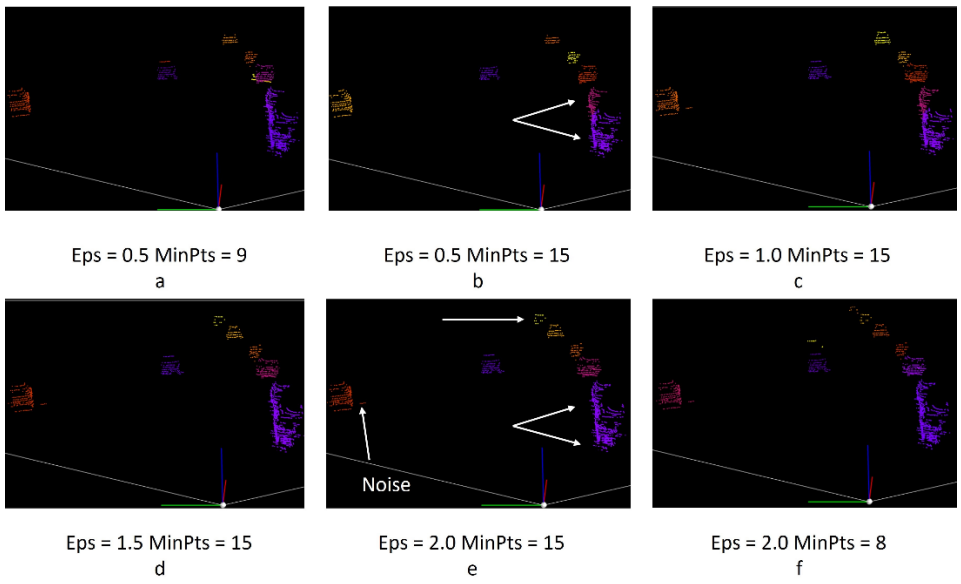

Figure 27. DBSCAN with different Eps and minPts [16]

To reduce computations and to get insights from the LiDAR sweep locally, the LiDAR FOV is divided into regions and the regions into cells. The number of regions and cells is selected by considering the LiDAR and camera FOV and the objects/lanes expected length and width. Road lanes width can be up to about 3.5m [103], and vehicles length is usually between 3 ~ 5 meters for mid-size vehicles [104]; therefore, taking into consideration the LiDAR FOV, dividing the ego

vehicle FOV into eight regions and eight cells is very reasonable. Figure 28 depicts the division scheme and the main parameter definitions.

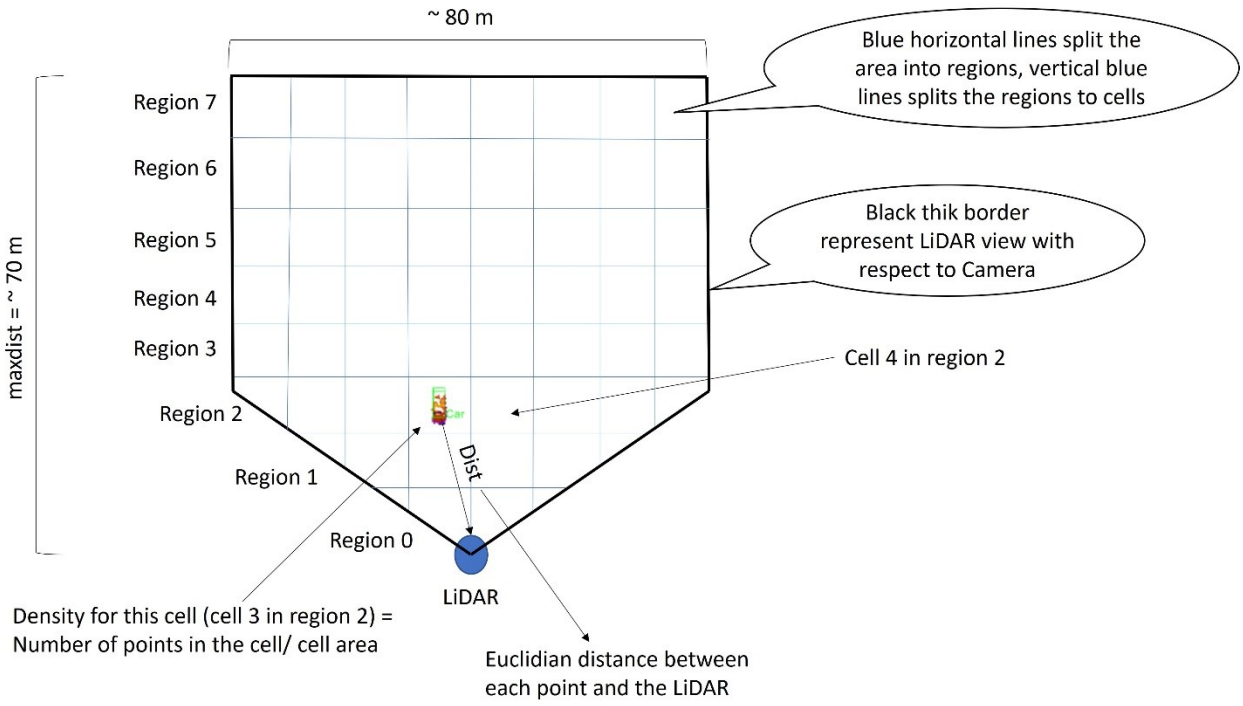

Figure 28. Proposed FOV division and parameters definitions

Based on the division scheme and different simulation experiments, two equations are also proposed that can estimate the values of ε and minPts for each LiDAR point individually and locally. Equation 4-3 calculates Eps while equation 4-4 calculates minPts.

$$Eps = \frac{\left(\frac{cell\_diagonal\_len}{FOV\_diagonal} * 100\right) * \left(\frac{dist}{maxdist}\right) * regionId}{\text{sqrt}(density)} \tag{4-11}$$

$$MinPts = \frac{total\_regions * \left(\frac{Eps}{minPtsDist[regionId]}\right)}{regionId} \tag{4-12}$$

Where $dist$ is the Euclidean distance between the point and the LiDAR(origin), $maxdist$ is the LiDAR max depth. It is 70.2 m for the used LiDAR in the KITTI dataset, $total\_regions$ is the total number of regions, $regionId$ is the id of the region where the point is located, and $cell\_diagonal\_len$ is the diagonal distance within cells and $FOV\_diagonal$ the diagonal distance within the entire FOV. Also, the $density$ is the density of points per cell and is calculated in equation (4-13) as follows:

$$density = \frac{\#of\ points\ per\ cell}{cell\ area} \tag{4-13}$$

And $minPtsDist$ is the minimum horizontal distance between LiDAR points, and it is calculated with reference to the middle of each region using the equation (4-14) below:

$$minPtsDist\ [\text{regionId}] = 2 * \tan\left(\frac{LiDAR_{H_R}}{2}\right) * region_{MD} \tag{4-14}$$

Where LiDAR_{H_R} is the LiDAR horizontal resolution and $\tan\left(\frac{LiDAR_{H_R}}{2}\right)$ is the tan of the half horizontal LiDAR resolution, which is 0.4 degrees for the used LiDAR in the KITTI dataset, $region_{MD}$ is the distance between the middle of the region and the LiDAR. The equation applies triangle relations to find $minPtsDist$, and Figure 29 provides more parameters' illustrations for equation (4-14).

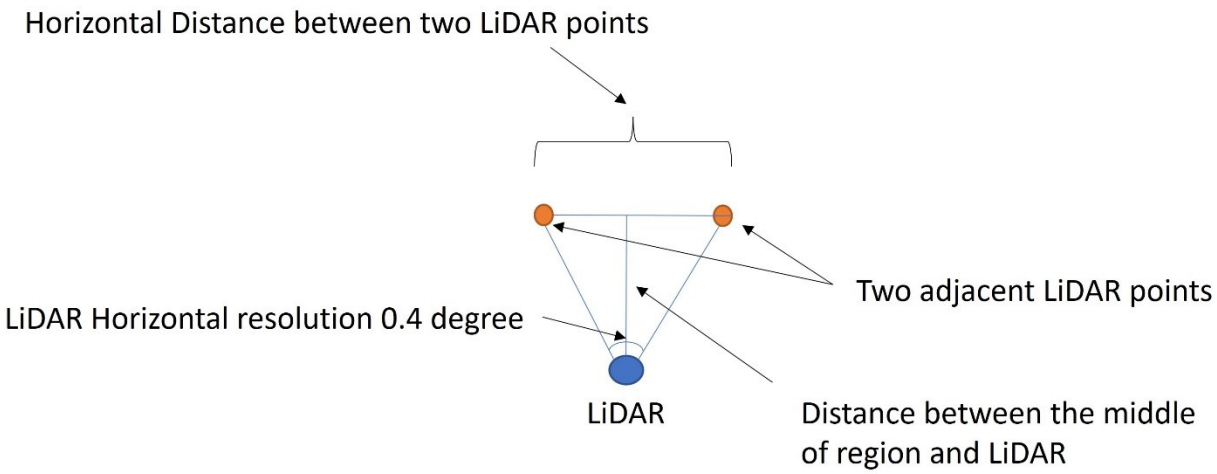

Figure 29. MinPtsDist parameters illustration

CHAPTER 5

EXPERIMENTAL RESULTS

Dataset

In this dissertation, the well-known KITTI Vision Benchmark Suite [16] has been selected and used to evaluate the proposed methodology and to generate different visual and numerical results. KITTI raw data are transformed, customized, and processed to support various research areas like stereo, flow, scene flow, depth prediction, depth completion, object detection, object classification, object tracking, road lane detection, and semantic segmentation. KITTI contains about six hours of driving in various set-ups, like driving in rural areas, highways, and urban areas. The dataset is well-calibrated, synchronized, and labeled. The dataset was captured in Karlsruhe, Germany, by a moving VW station wagon equipped with four Camera sensors (colored and monochrome), a LiDAR scanner (Velodyne HDL-64E), and high-precision GPS/IMU (OXTS RT3003). The raw data is divided into five main categories: Road, City, Residential, Campus, and person. Data are also saved into three folders: Images containing images captured by the four cameras. OXTS includes the GPS and IMU values and Velodyne, which has the LiDAR data points. The proposed work is related to object detection and depth completion; thus, in section 5.1.1, detailed information about KITTI object detection is provided, in section 5.1.2, detailed information about the depth completion dataset is provided. At the same time, in section 5.1.3, detailed information about the instance segmentation dataset is also provided.

KITTI object detection dataset

The KITTI Object detection dataset contains around 7481 training images and 7518 testing images. The total number of labeled objects is 80256. Objects include cars, pedestrians, cyclists, trams, people sitting, and misc. Another essential piece of information is the level of occlusion and truncation for each labeled object. Each label contains the following interesting information: frame id, track id, truncation level (from 0 to 2) and occlusion level (from 0 to 3), object angle, bounding box information in 2D format or 3D format, location, and rotation around the y-axis. For the three major object types of KITTI: cars, pedestrians, and cyclists, Table 3 provides the total number of labels for each object type distinguished by the level of complexity.

Table 3. KITTI dataset main classes statistics for the training split

Class	Total instances	Easy	Moderate	Hard
Cars	11017	3153	4893	2971
Pedestrians	2113	1186	653	274
Cyclists	597	378	179	40

In the clustering work, the 2D KITTI object detection dataset is used, wherein bonding boxes are provided in the form of 2D rectangles. LiDAR data are raw and provided in a 360-degree range. Thus, the LiDAR data is also cropped and aligned with the Camera's field of view. Figure 30 shows an example from the KITTI object detection dataset. Figure 30-a shows the camera frame

with 2D bounding boxes, Figure 30-b shows the camera frame with 3D bounding boxes, and finally, Figure 30-c shows LiDAR point cloud from the camera point of view with both 3D bounding boxes and labels.

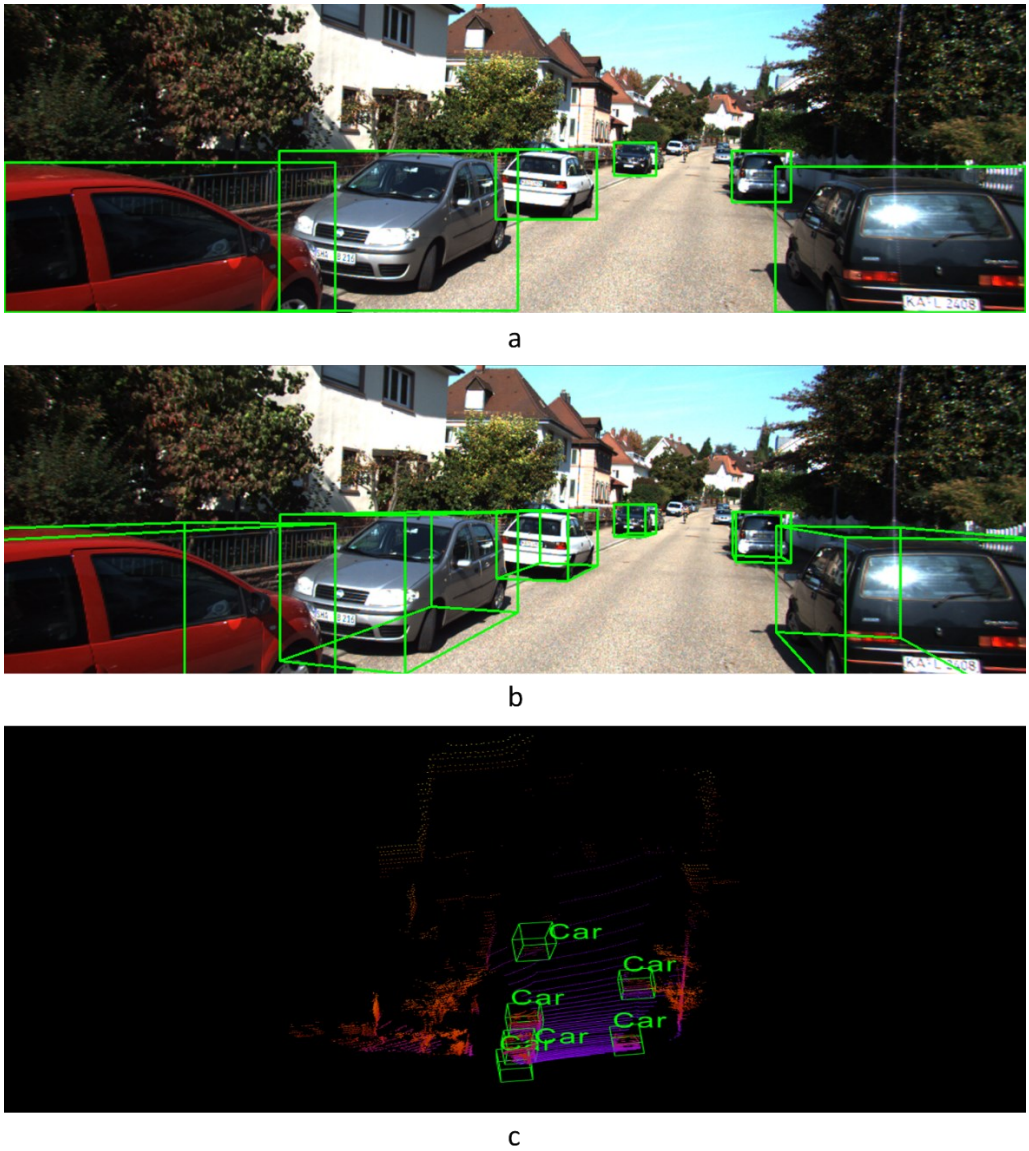

Figure 30. KITTI object detection dataset example a) camera image with 2D bounding boxes b) camera image with 3D bounding boxes c) LiDAR point cloud with 3D bounding boxes and labels from camera FOV [16]

KITTI depth completion dataset

The KITTI depth completion dataset is a large-scale dataset created based on the raw KITTI dataset. which consists of 85,898 training data frames from RGB cameras and LiDAR; the dataset also has 1K validation data frames. We observed that the RGB images are extracted from two cameras positioned to capture the car's front view. To reduce the training time, we used only one camera data, which reduced the number of frames to 42,949, and further, we applied a ¼ random sampling, resulting in a final training dataset size of 10,737 frames. Each training sample consists of four main entities: (1) RGB frame with a resolution of 1216x352, (2) Sparse LiDAR depth map, (3) Ground truth depth map, and (4) Instance segmentation information encoded using our encoding algorithm. It is worth noting that the sparse depth maps have about 5% valid depth information, and the ground truth depth maps have about 16% valid depth information.

The dataset creators automated the noise and artifacts removal process available in the KITTI raw dataset due to different sources like occlusions and reflection. The automated cleaning process is done by correlating the LiDAR scans to the depth maps generated using the stereo reconstruction approach. They accumulate eleven laser beams to increase the density of the generated depth maps. And finally, evaluate the generated dataset by comparing it with the manually cleaned stereo dataset [105]. Figure 31 shows an example from the KITTI depth completion dataset. Figure 31-a shows the Camera sensor image. Some work only uses the camera data for referencing purposes (non-guided depth completion) and do not use it in their pipeline.

In contrast, other works apply a guided depth completion approach that combines multiple sensor information like camera and LiDAR. Figure 31-b shows the sparse LiDAR depth map, and Figure 31-c shows the dense LiDAR depth map (the ground truth in the dataset). For both Figure

31-b and Figure 31-c, LiDAR depth information has been projected into the camera 2D plan and color-coded based on the depth range for easier understanding and better visualization.

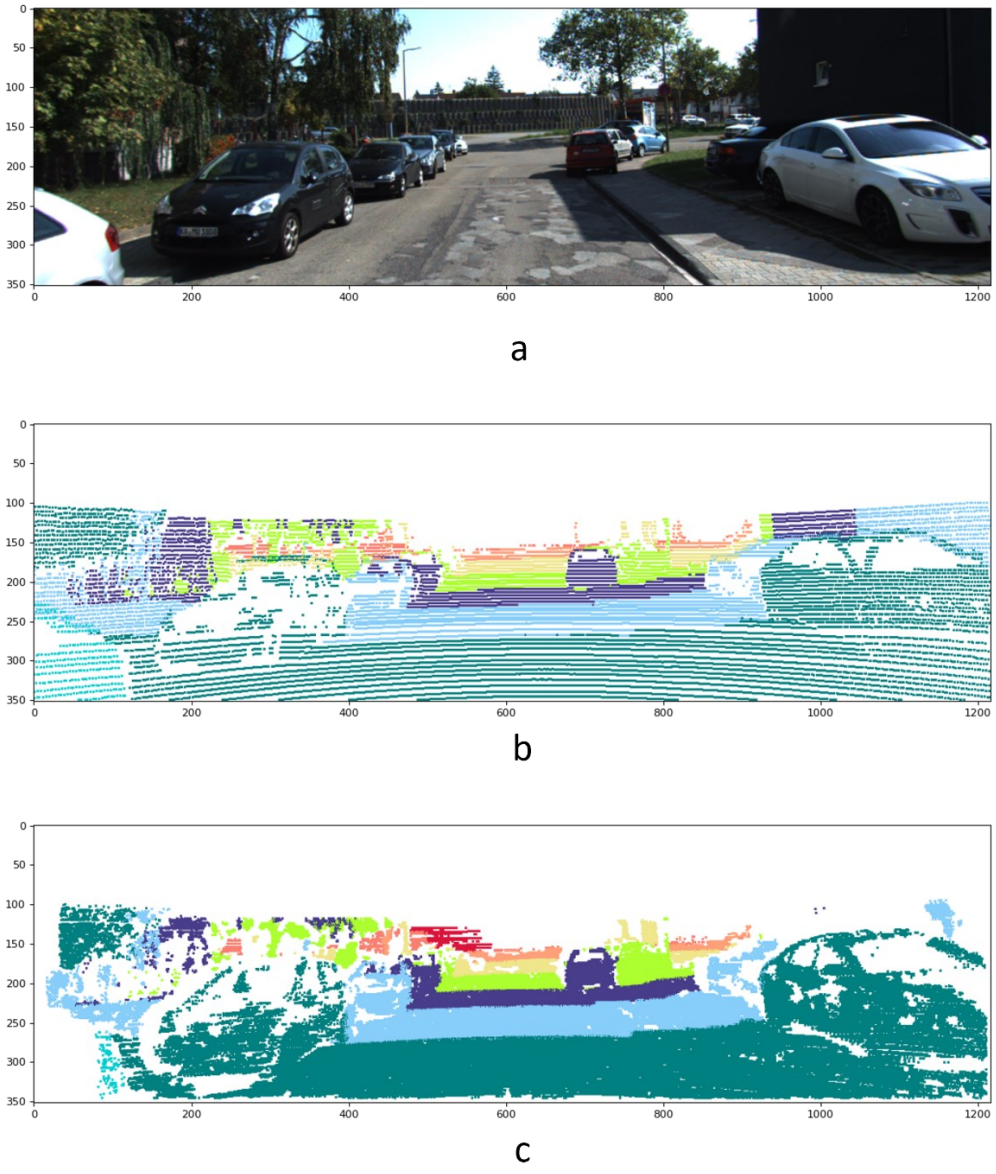

Figure 31. KITTI depth dataset a) Camera image b) Sparse LiDAR depth map in the 2D domain c) LiDAR ground truth in the 2D domain [16]

**Depth maps have been colored for easier visual understanding*

KITTI instance segmentation dataset

KITTI also provides a dataset for instance segmentation work [52]. The dataset consists of 200 training images and another 200 validation images, combining real and synthetic data. Figure 32 depicts an example of the instance segmentation dataset and shows the wide range of semantically labeled objects in a self-driving context.

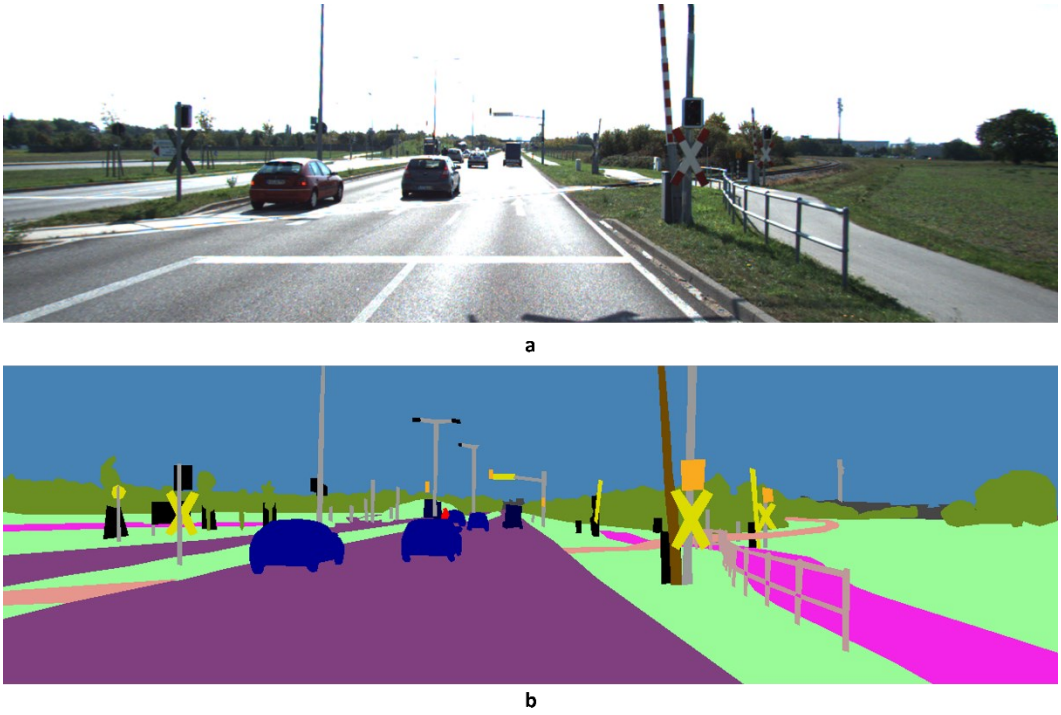

Figure 32. KITTI Instance Segmentation Dataset [52] *Example a) reference camera image b) Instance segmentation of the image objects, wherein each color is assigned to a unique object, e.g., dark blue for cars*

Experimental Setup

Environment

The simulation and analysis have been carried out in two environments; the development and prototyping have been done in a local server located in DISPLY lab. The server is Intel(R) Xeon(R) CPU E5-1620 0 @ 3.60GHz equipped with 32GiB of RAM and a GTX 1080 TI GPU. The operating system is Ubuntu, and the primary programming language is Python. Both TensorFlow and Pytorch frameworks have also been used. The long training and simulation have been done on google Colab environment with Pro membership which provides Tesla P100 or Tesla T4 GPU with 25.4 GB graphic RAM and 32GB of machine RAM and 167GB SSD storage. Table 4 provides additional specification information about the local development environment.

Table 4. DISPLY lab Development Environment Specifications

Item	Specification
Processor	Intel(R) Xeon(R) CPU E5-1620 0 @ 3.60GHz
Number of cores	8
RAM	32836MB
GPU	GTX 1080 TI
GPU RAM	11 GiB
Machine Type	Tower
Operating System	Ubuntu 20.04.2 LTS
Programming language 1	Python3 V 3.8.10

Tabel 4 - continued

Programming language 2	Pytorch V 1.2.0
Environment control	Anaconda
Nvidia toolkit	Driver Version: 470.103.01
CUDA	Version: 11.4

Instance Segmentation Based Depth Completion Framework Using Sensor Fusion

Performance metrics

In this research, for instance segmentation, the standard COCO metrics: AP, $AP^{0.5}$ and $AP^{0.75,}$ where AP stands for average precision and 0.5 and 0.75 represent the area overlap threshold is used. Instead of the general intersection over Union (5-10), the Mask IOU to evaluate the generated masks' quality is used. As for depth completion, the KITTI benchmark and other existing methods [33], [69], [70] are used as well as the five standard metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), inverse Depth Root Mean Squared Error (iRMSE), inverse depth Mean Absolute Error (iMAE) and the Mean Squared Error (MSE). These metrics are as given in equations (5-1), (5-2), (5-3), (5-4), (5-5), respectively:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(\hat{d}_i - d_i)^2} \tag{5-1}$$

$$MAE = \frac{1}{N}\sum_{i}^{N}\left|(\hat{d}_i - d_i)\right| \tag{5-2}$$

$$iRMSE = \sqrt{\frac{1}{N}\sum_{1}^{N}\left|\frac{1}{\hat{d}_i} - \frac{1}{d_i}\right|^2} \tag{5-3}$$

$$iMAE = \frac{1}{N}\sum_{1}^{N}\left|\frac{1}{\hat{d}_i} - \frac{1}{d_i}\right| \tag{5-4}$$

$$MSE = \frac{1}{N}\sum_{i}^{N}(\hat{d}_i - d_i)^2 \tag{5-5}$$

where $\hat{d}_i, d_i$ are the predicted pixel depth value and ground truth pixel depth value, respectively, and N is the number of pixels.

Instance segmentation results

The instance segmentation network has been trained on the KITTI instance segmentation dataset for 42 epochs. Figure 33, Figure 34, Figure 35, and Figure 36 show four primary loss performances over the epoch's interval: the combined loss, bounding box loss, class loss, and mask loss, respectively. Losses values were recorded for both the training subset and validation subset. All losses decrease significantly within the first 30 epochs. For example, the mask loss was around 0.06 at epoch 30 and stayed around the same value for the remaining training process. The validation loss is generally small and very close to the training loss, which means that the model is not over-fitted and can be generalized well for new unseen data. Equations (5-6), (5-7), (5-8), and (5-9) show the mathematical equations used to calculate classification loss, bounding box loss, mask loss, and combined loss [43], [95], respectively.

$$L_{cls}(p_i, \hat{p}_i) = -log\ (p_i^c) \tag{5-6}$$

where, p_i^c is the predicted probability for the proposed region belonging to class c

$$L_{box} = l_1^{smooth}(t_i - \hat{t}_i) \tag{5-7}$$

where t_i , $\hat{t}_i$ are the ground truth coordinate of the bounding box and the predicted coordinate, and l_1^{smooth} is a combination of Least Absolute Deviations and Least Square Errors.

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij}\, log\, \hat{y}_{ij}^k + (i - y_{ij}) log(1 - \hat{y}_{ij}^k) \right] \tag{5-8}$$

$where\ y_{ij}$ is the ground truth label of the pixel at i, j location $and\ \hat{y}_{ij}^k$ is the predicted label of the pixel at *i,j*.

$$Combined\ loss = L_{cls} + L_{box} + L_{mask} \tag{5-9}$$

Figure 37 shows the performance of the instance segmentation model on a randomly selected frame from the KITTI dataset. The top left image shows the frame with the instance mask placed on each object. The figure also shows the LiDAR sparse point cloud and instances examples. Interestingly, instance segmentation masks precisely identify each object's boundaries especially occluded and small objects such as those in examples 4 and 5.

Depth completion neural network can work either on each instance individually or the entire scene with preliminary information about instance masks and types.

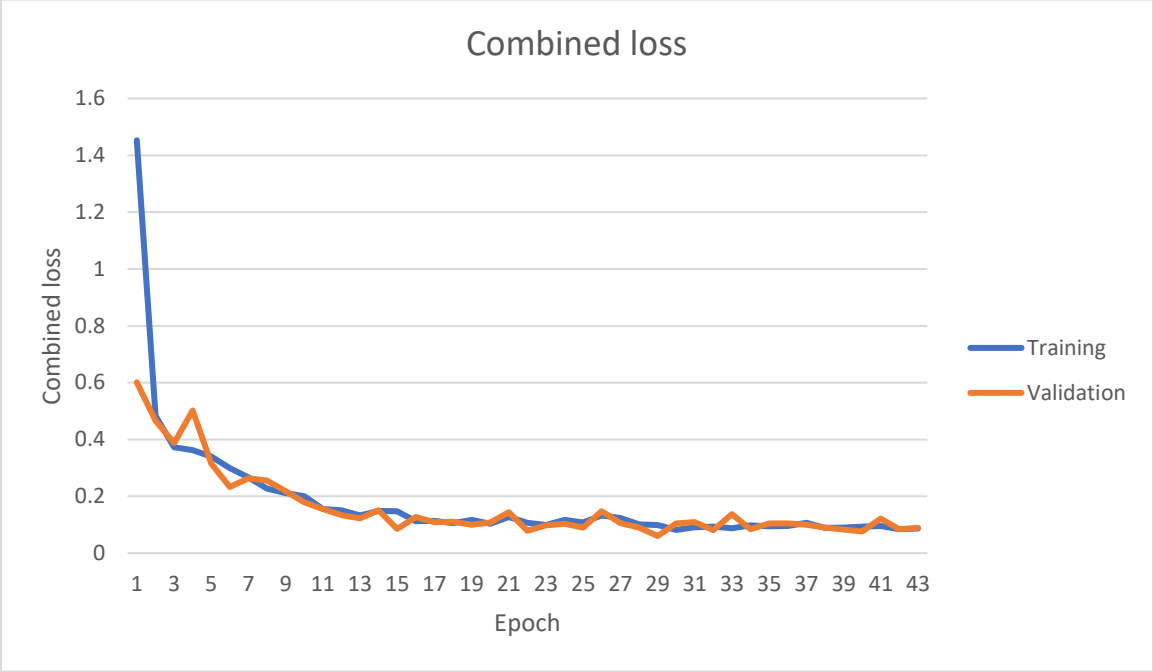

Figure 33. Instance segmentation combined loss on both training and validation subsets over different epochs

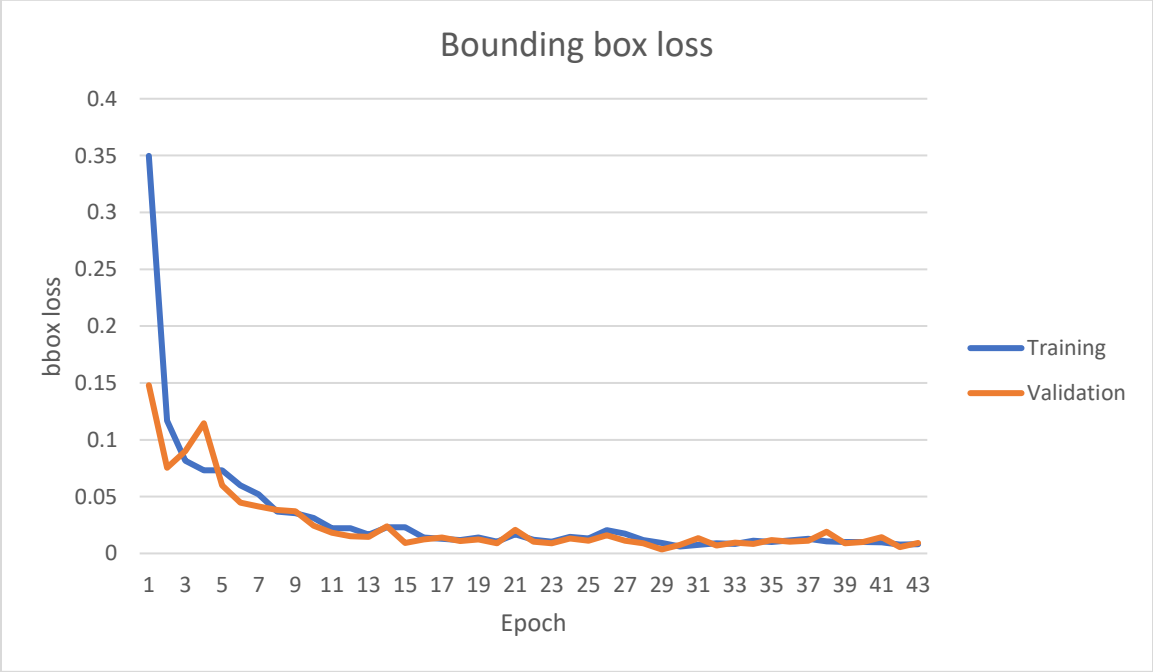

Figure 34. Instance segmentation bounding box loss on both training and validation subsets over different epochs

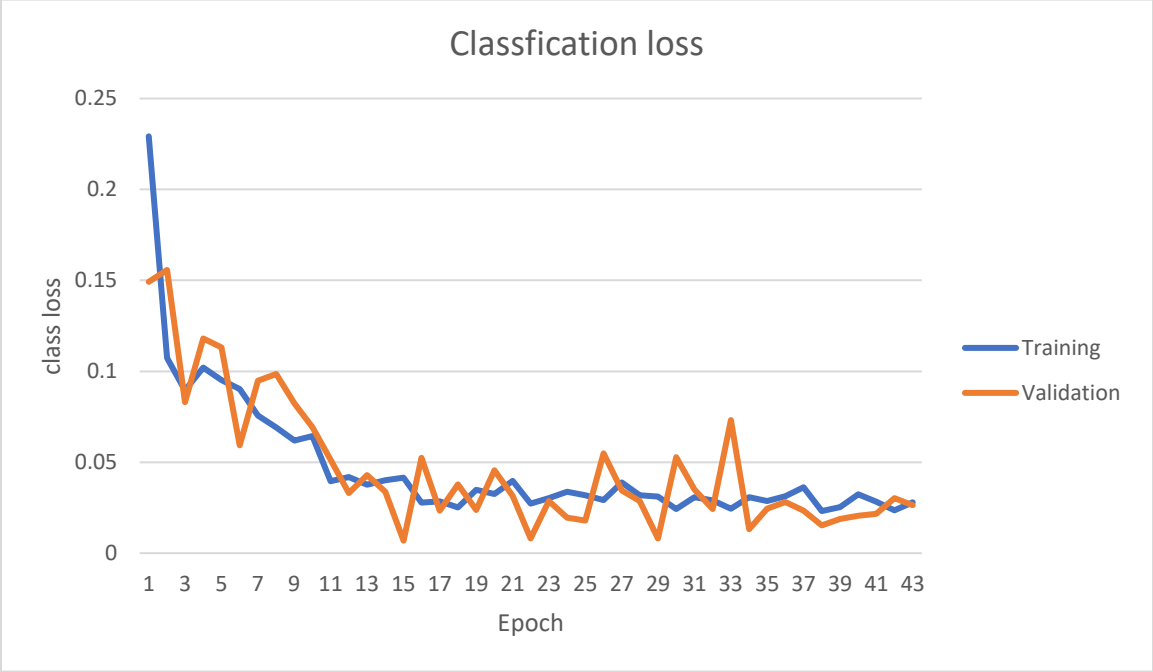

Figure 35. Instance segmentation classification loss on both training and validation subsets over different epochs

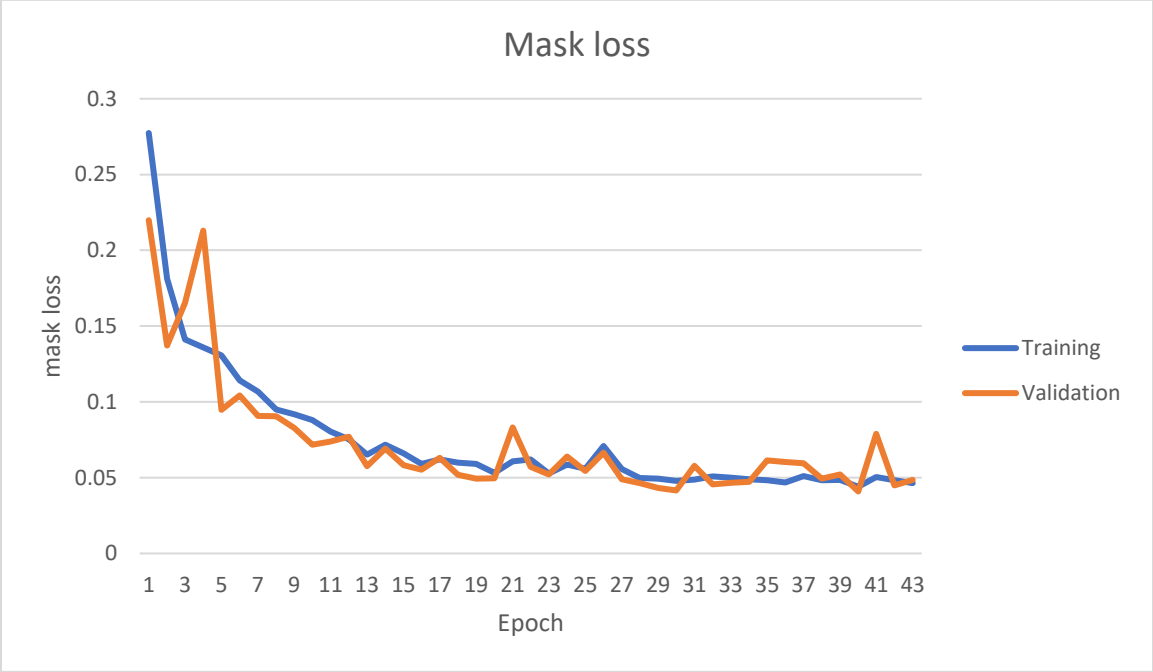

Figure 36. Instance segmentation mask loss on both training and validation subsets over different epochs

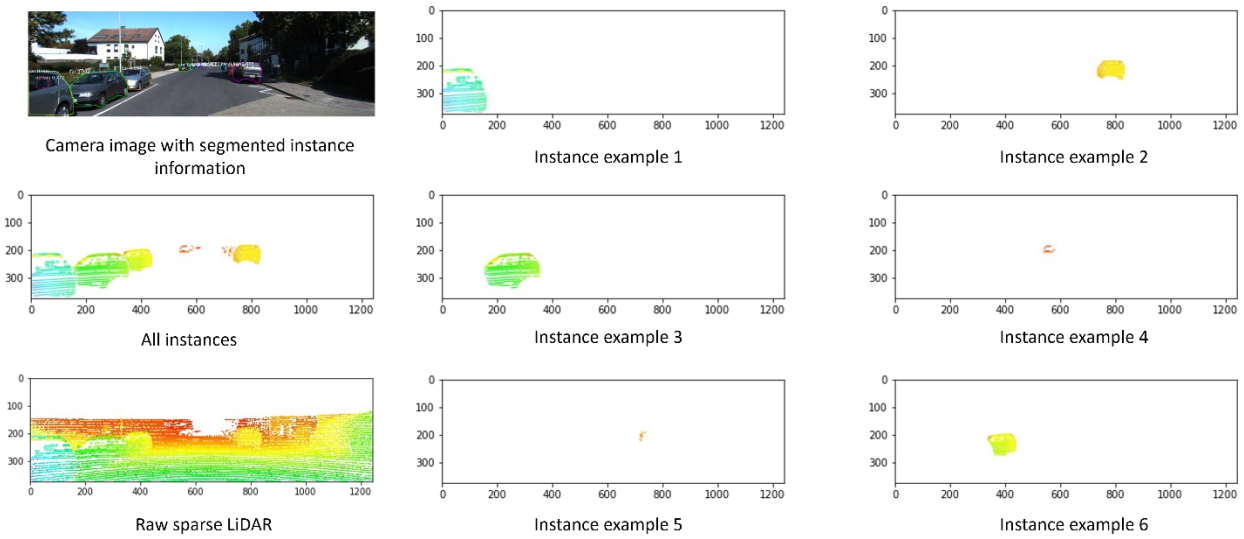

Figure 37. Performance of the trained instance segmentation model on a randomly selected frame from the KITTI dataset [16]

Although the KITTI instance segmentation dataset is small, using the transfer learning technique, a very close performance measures to those generated from the official Mask- RCNN work. Table 5 summarizes the performance measures for the custom-trained Mask-RCNN and the official trained Mask-RCNN.

Table 5. Performance metrics of the custom-trained Mask-RCNN

	AP	AP_{50}	AP_{75}
Custom trained Mask-RCNN	32.1%	51.4%	34.6%
Official Mask-RCNN	37.1%	60%	39.4%

Depth completion results

Several experiments have been done to evaluate the impact and usefulness of instance segmentation in the depth completion problem. It is worth mentioning that depth completion training is a resource and time-consuming process. Most recent works reported days or weeks of training time. Our Baseline model is the ENet [33] model. Since we are using a subset of the KITTI dataset, we first retrained the ENet on the prepared subset and used it as our baseline. The 1K validation images supplied by the KITTI dataset have been used for validation purposes. Figure 38 shows the validation RMSE for both the proposed instance segmentation-based depth completion and the baseline model; the RMSE decreases as the epoch increases, indicating that the models are improving and that the proposed method surpasses the baseline model. It is also clear that at early epochs, the proposed method starts with a lower RMSE than the baseline model, which is about 20K RMSE less than the baseline model.

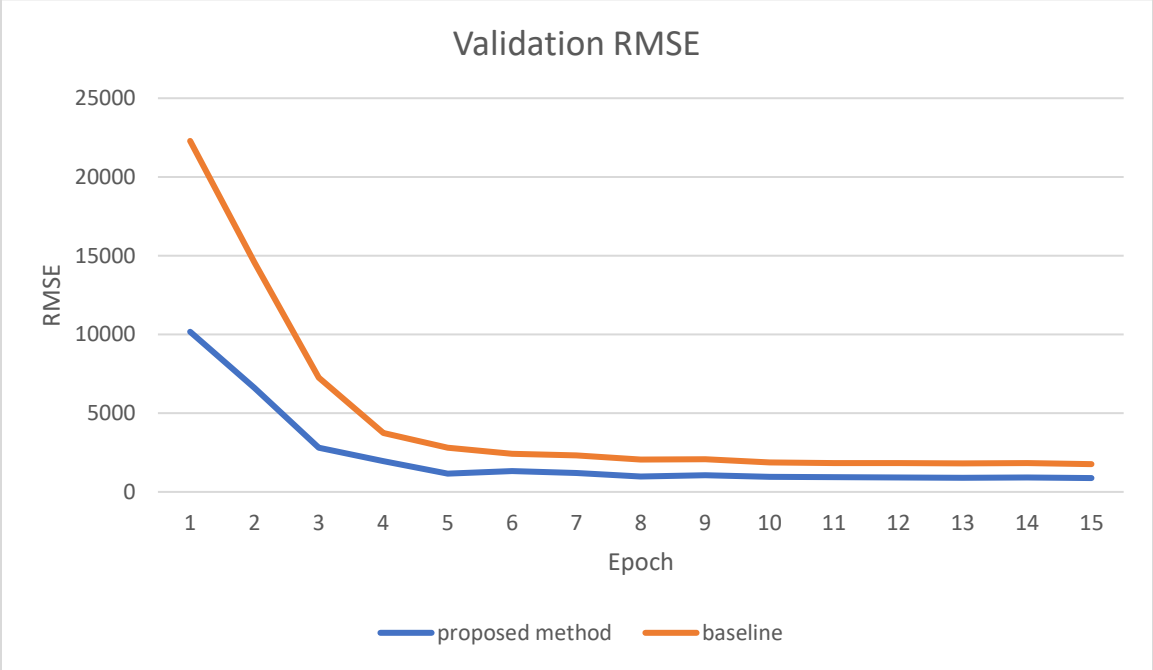

Figure 38. RMSE for both the proposed method and the baseline model over different epochs

Figure 39 shows the validation MAE for both the proposed instance segmentation-based depth completion and the baseline model; the MAE decreases as the epoch increases, indicating that the models are improving and that the proposed method surpasses the baseline model. It is also clear that at early epochs, the proposed method starts with a lower MAE than the baseline model, which is about 10K MAE less than the baseline model.

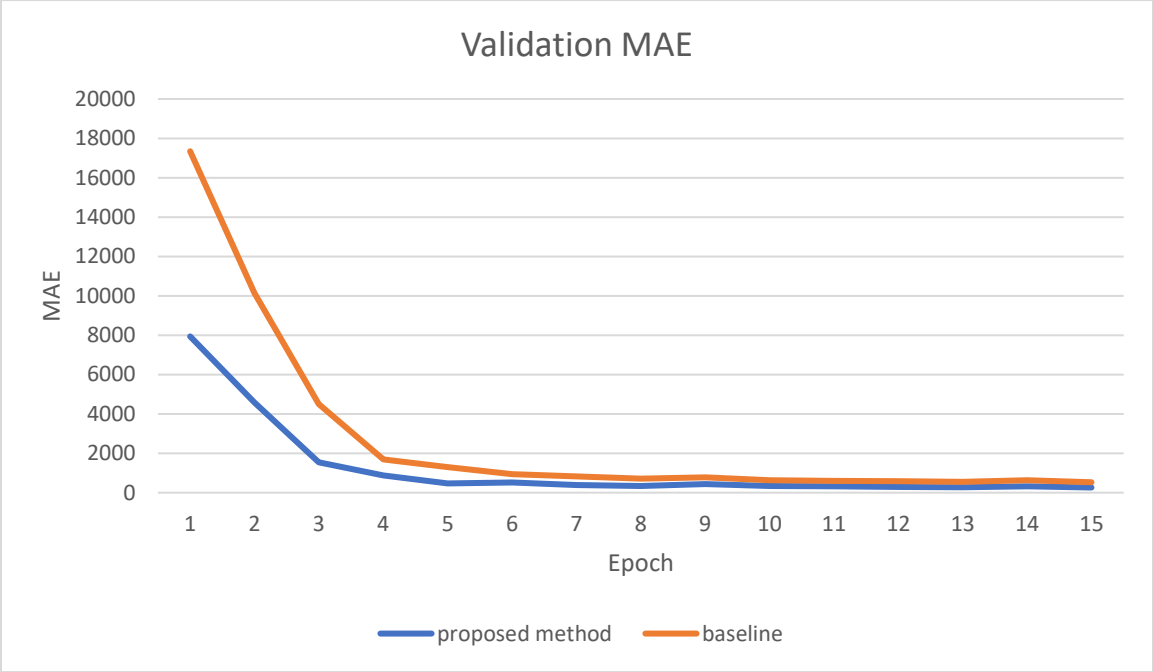

Figure 39. MAE for both the proposed method and the baseline model over different epochs

Figure 40 shows the validation MSE for both the proposed instance segmentation-based depth completion and the baseline model; the MSE decreases as the epoch increases, indicating that the models are improving and that the proposed method MSE is less than the baseline model

by a good factor. It is also clear that at early epochs, the proposed method starts with a lower MSE than the baseline model, which is about 150M MSE less than the baseline model.

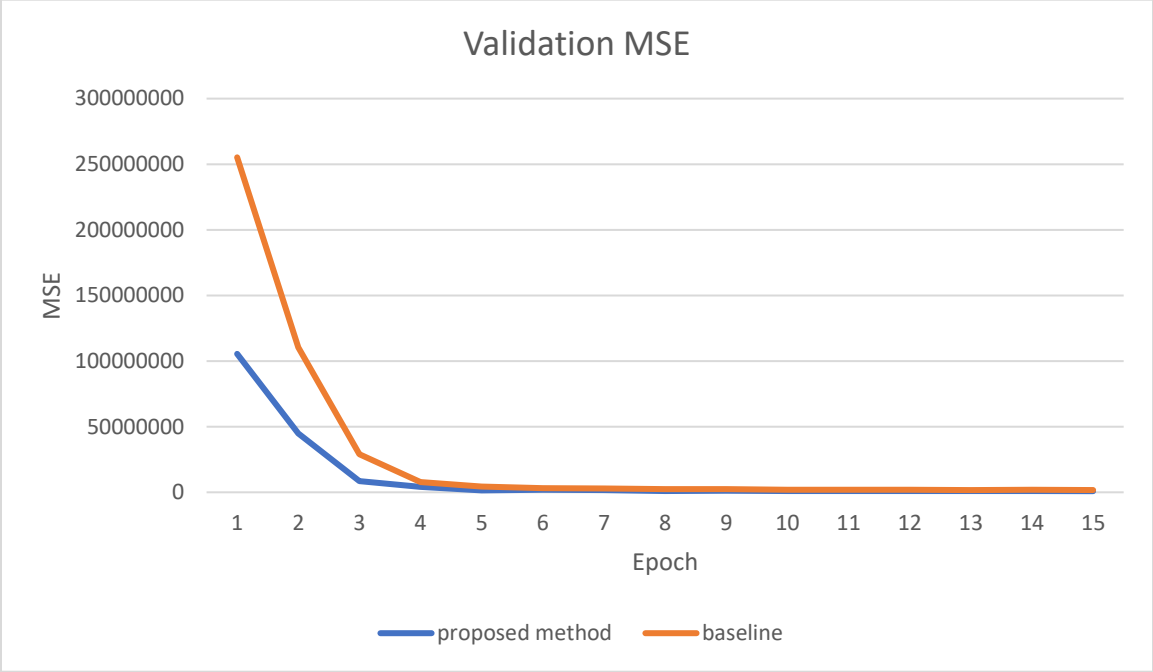

Figure 40. MSE for both the proposed method and the baseline model over different epochs

Figure 41 and Figure 42 shows RMSE and MAE but on during the training phase. The figures show that the proposed method outperforms the baseline model and starts with lower errors at the initial stages.

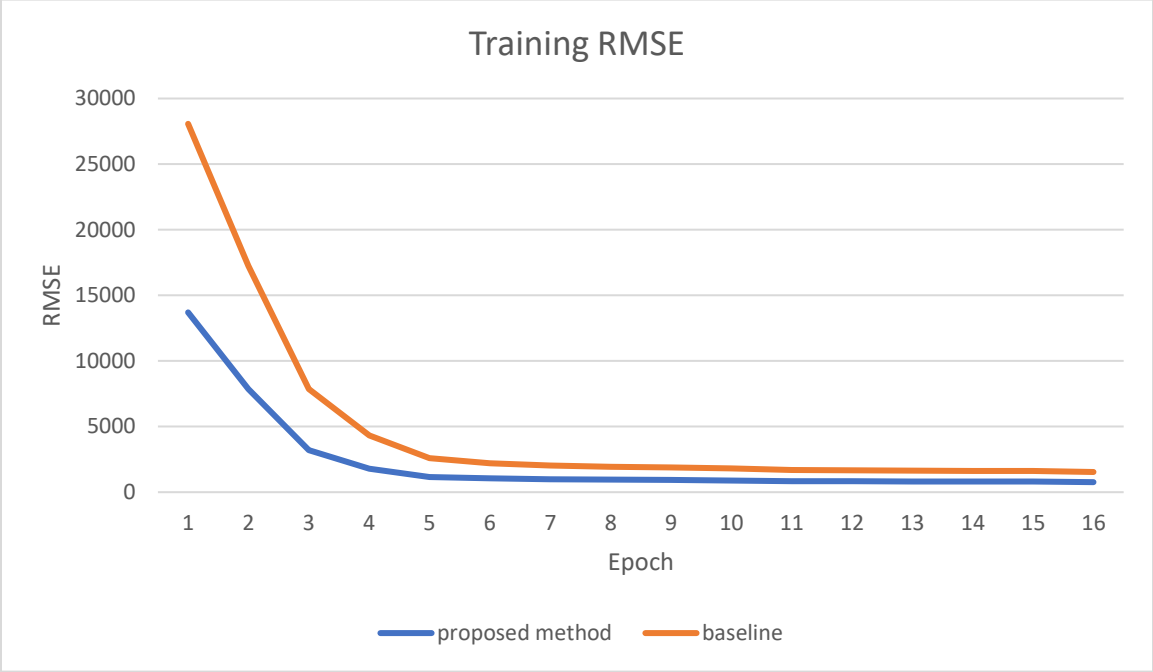

Figure 41. RMSE over different epochs for both the proposed method and the baseline model at the training phase

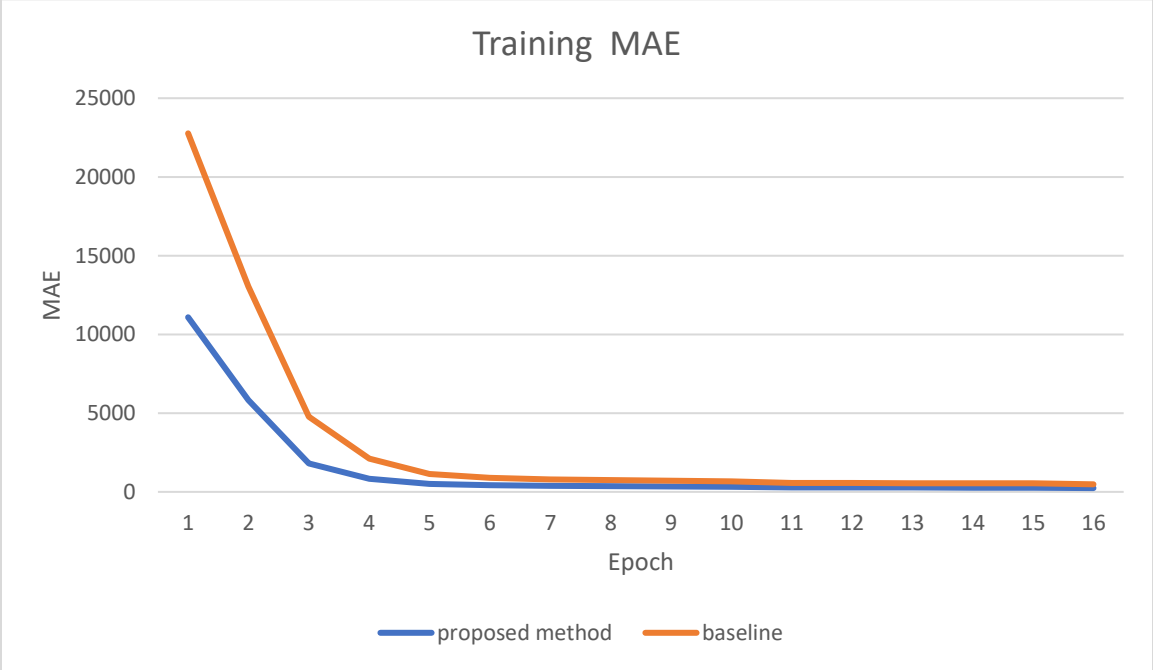

Figure 42. MAE over multiple epochs for both the proposed method and the baseline model at the training phase

Table 6 shows the validation performance measure for the proposed method and the baseline model at the last trained epoch. All listed metrics are the lower is the better. The proposed method outperforms the baseline model by 0.35 %, 0.64%, 1.85%, 1.32%, 0.37% and 4.74% for RMSE, iRMSE, MAE, iMAE and MSE, respectively.

Table 6. Performance measures on the validation dataset at the last epoch for both the proposed method and the baseline model

Metric	Baseline	Proposed method
RMSE ↓ (mm)	882.636	879.525
iRMSE ↓(1/KM)	3.178	3.1585
MAE ↓(mm)	266.933	261.991
iMAE ↓(1/KM)	1.279	1.262

Table 7. Performance measures on the training dataset at the last epoch for both the proposed method and the baseline model

Metric	Baseline	Proposed method
RMSE ↓ (mm)	773.704	764.626
iRMSE ↓(1/KM)	3.373	3.327
MAE ↓(mm)	239.695	240.602
iMAE ↓(1/KM)	1.268	1.290
MSE ↓	623970.8	613420.5

Table 7 shows the training performance measure for the proposed method and the baseline model at the last trained epoch. The proposed method outperforms the trained model in the training RMSE, and iRMSE and was very close in performance in the remaining.

Figure 43 shows three examples of the quality of the proposed method compared to the baseline model. For each example, we are providing the RGB image as a reference, the depth map of the proposed method, and the depth map from the baseline model, and the instance segmentation mask. We are also highlighting interesting areas that are easy to focus on to see the strength of the proposed method compared to the baseline model. Finally, we also provide the RMSE and MAE for each example. The proposed method in the three examples revealed an outstanding performance when focusing on objects and their boundaries.

In Figure 43, In example 1, the white rectangle highlights how the proposed method could distinguish between the two cars and the area between them. In the baseline model, the area was considered part of one of the vehicles. In example 2, the white rectangle highlights how the proposed method was able to distinguish between the right side of the cyclist while the baseline model included part of the background area to the same depth level as the right side of the cyclist. Finally, in example 3, the white rectangle highlights how the proposed method was able to distinguish between the two adjacent pedestrians while the baseline model merges them.

In summary, experimental results proved the ability of instance segmentation clues in guiding depth completion tasks. The proposed framework achieved superior performance compared to the baseline model. Both quantitative and qualitative measures have been addressed. Quantitatively the framework surpasses the baseline in all related metrics like RMSE, MAE, and

MSE. Additionally, in earlier training epochs, the proposed framework showed smaller error values than the baseline model, proving the ability of instance segmentation clues to efficiently guide the depth completion task.

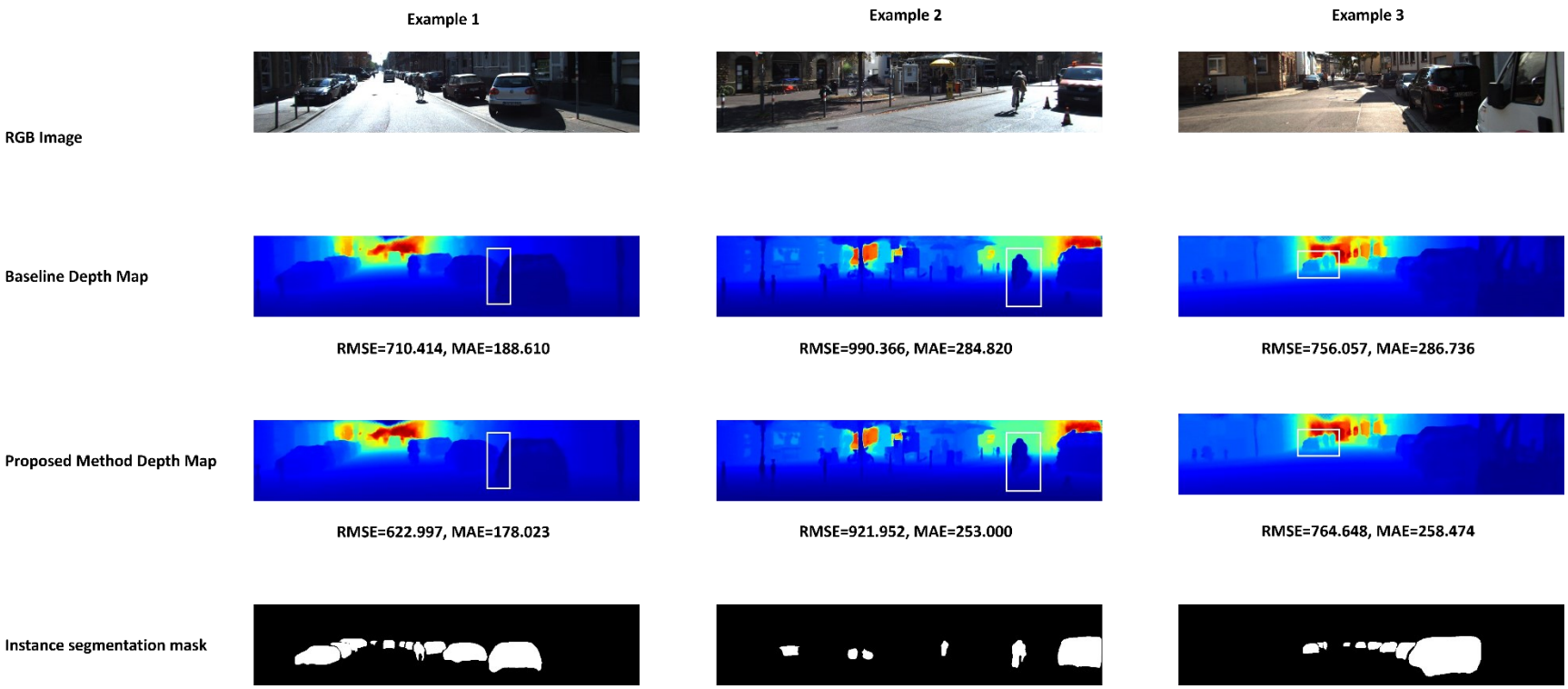

Figure 43. Qualitative comparison between the baseline model and the proposed method [16]

LiDAR Point Cloud Clustering Results

Performance metrics

An evaluation framework has been developed to evaluate the clustering performance that compares the ground truth 2D bounding boxes from the dataset with the 2D bounding boxes resulting from the clustering process. Clusters are considered correct if the intersection over union between the ground truth bounding box and the bounding box generated from the clustering process is less than a specific threshold. As a final metric, the total number of correct clusters vs. the total number of incorrect clusters is used. The flowchart in Figure 44 depicts the entire evaluation process.

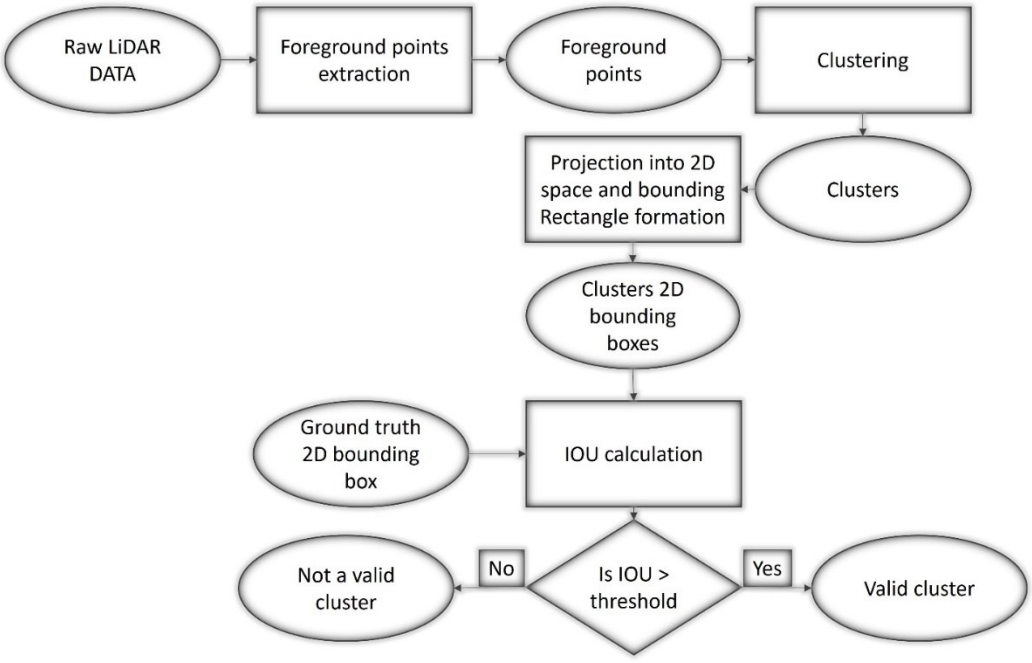

Figure 44. Proposed DBSCAN-based clustering evaluation flowchart

A graphical representation from the real dataset is also depicted in Figure 45. The 2D image is used for illustration and evaluation purposes only. LiDAR Data as shown in Figure 45-b, is processed by removing the background points. Clusters of the foreground points are projected as a 2D image as in Figure 45-c, each cluster colored with a unique color to make the visual evaluation easier. In the last step, as shown in Figure 45-e, a bounding box for the clusters is generated (green box) and compared with the ground truth 2D bounding box (white box) provided by the KITTI dataset. The intersection is calculated (the black area), and the two bounding boxes are correlated using the Intersection Over Union (IOU) as shown in equation (5-10) [25], and assessed based on a specific threshold.

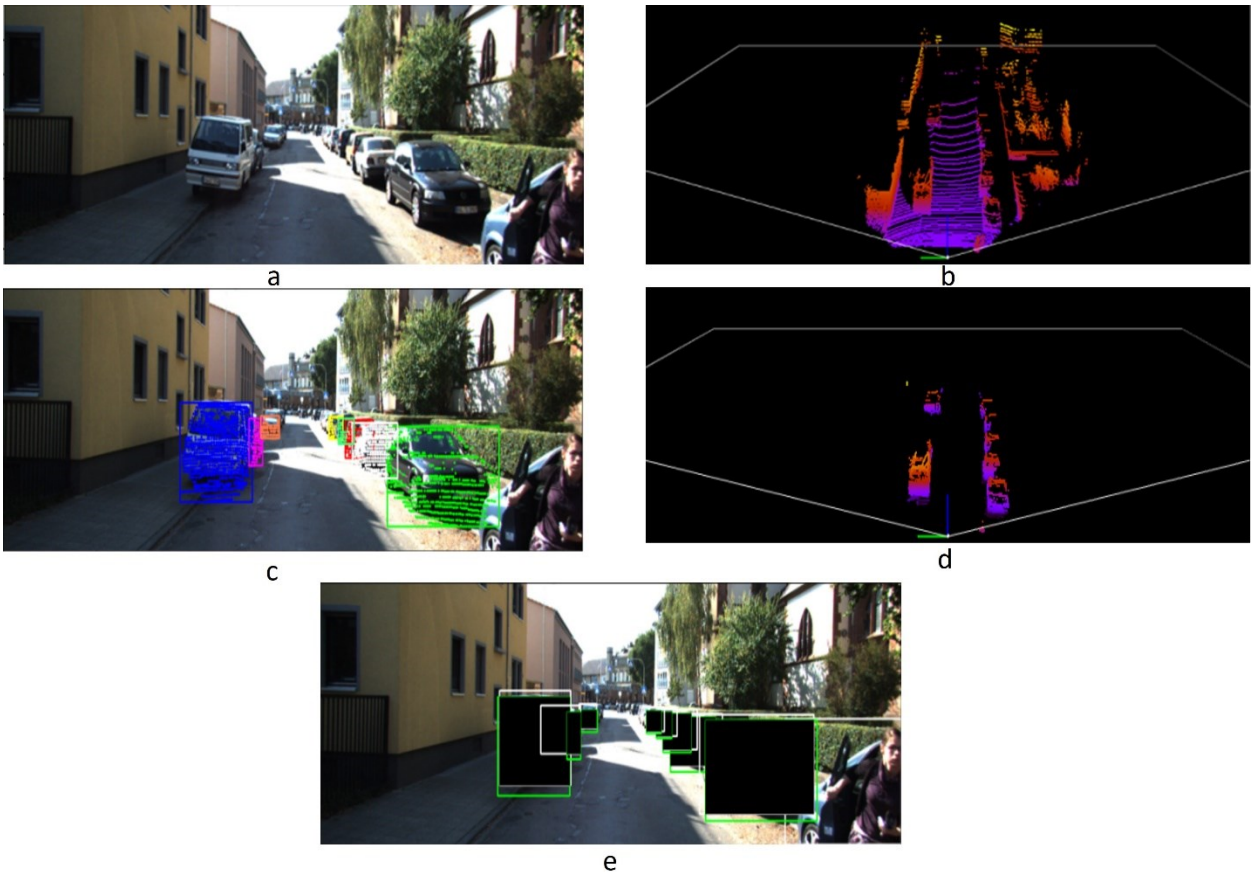

Figure 45. Clustering evaluation on a selected KITTI frame a) raw camera image, b) raw LiDAR data, c) generated clusters in a 2D domain, d) generated clusters in a 3D domain, e) ground truth and generated 2D initial bounding boxes and intersections [16]

$$IOU(b, b^g) = \frac{area\ (b\ \cap\ b^g)}{area\ (b\ \cup b^g)} \tag{5-10}$$

where b and b^g are the generated bounding box and the ground truth bounding box, respectively.

It is worth noting that the IOU has been used since the dataset does not provide cluster information explicitly. In fact, we utilize the bounding boxes to find points association; thus, IOU can provide a good indicator of accuracy for the proposed clustering method.

Clustering qualitative and quantitative results

Using the same LiDAR sweep of Figure 27, clustering simulation was repeated using the proposed method. As shown in Figure 46, all objects were successfully clustered without assigning specific ε or minPts. The results also show that all potential objects were identified without merging adjacent objects.

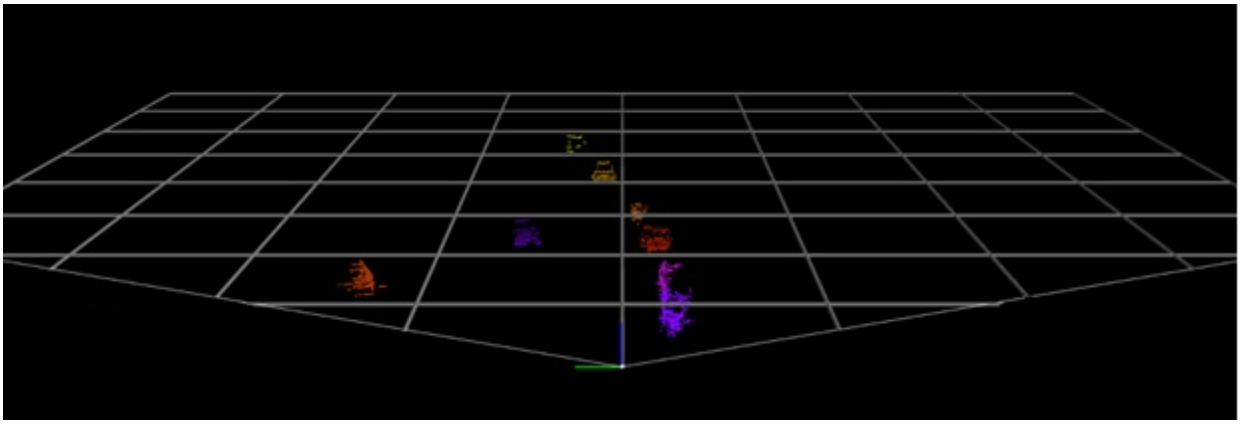

Figure 46. Qualitative clustering results of the proposed method, all objects have been appropriately clustered, especially those that are very close to each other [16]

Applying the evaluation process depicted in Figure 44 to the entire evaluation portion of KITTI dataset yields improvements when compared to DBSCAN. It is worth mentioning that for DBSCAN, extensive simulation experiments have been done to estimate the best tuning parameters. The results presented in the second column of Table 5 are the values that delivered the best performance.

In Table 8, the proposed algorithm achieved relatively similar performance to the original DBSCAN with the brute-forced parameters. The proposed equations consider LiDAR specs; thus, the proposed algorithm can be used in applications with different LiDAR specs.

Table 8. Quantitative results of the proposed clustering method compared to the original DBSCAN

Performance Indicator	DBSCAN ($\varepsilon = 1$, minPts =10)	ADEVLA-DBSCAN
Total Ground truth samples	14385	14385
Total Number of correctly detected samples	12170	12114
Total number of not detected samples	2215	2271
Accuracy	84.6%	84.2%

In summary, the experimental results using the KITTI object detection dataset indicated the ability of the automated DBSCAN parameter selection framework to achieve a similar accuracy level compared to the manual selection approach. Reveling the framework's ability to cancel the need to brute force search for the proper tuning parameter values.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Self-driving cars are a promising technology not only it will reduce fatalities of humans and wild life, curb or end emissions and fuel consumption, but also it will revolutionize public transportation by increasing traffic efficiency and better travel experiences specifically for the physically and visually impaired. However, without artificial intelligence and machine learning advancements and stringent testing, AV will not be as planned and by 2030. Hence, developing high-quality, accurate, and real-time algorithms is of utmost importance. This dissertation focuses on the perception module of self-driving vehicles, wherein two key challenges have been addressed: depth completion and LiDAR point cloud clustering. Both are critical to self-driving cars' enablement as they are the primary source of 3D information, which is an essential for many perception tasks like 3D object detection, 3D instance segmentation, localization, simultaneous localization and mapping, and 3D object tracking.

For depth completion, a guided depth completion with a multi-level sensor fusion framework has been proposed. The framework utilizes and fuses the image instance segmentation features, such as object type and objects pixel-level locations, alongside the sparse depth measurement and the color image information to guide the depth completion process. For efficient training, a data structure and encoding scheme are developed to combine the objects' location-masks and object types into a 2D 1-channel array. Experimental results on the KITTI dataset indicated the ability of the proposed framework to generate accurate depth information and exceed the baseline performance. In addition, in earlier training epochs, the proposed framework starts with smaller errors than the baseline work, which eventually allows the model to reach a faster

convergence state. We demonstrated that instance segmentation clues are very valuable in guiding the depth completion process successfully.

For LiDAR point cloud clustering, a framework that is based on the DBSCAN algorithm to cluster LiDAR point cloud is proposed, wherein DBSCAN tuning parameters are adaptively and automatically estimated. The framework uses a felid of view division scheme and points cloud statistics to calculate tuning parameters for each frame. Experimental results using the KITTI dataset revealed the framework's ability to achieve relative accuracy to DBSCAN while eliminating the need for brute force search for the best tuning parameters to make the clustering algorithm more robust in SDCs environment.

Aiming to avoid the perceptual limitations of single sensor modality and to generate 3D far-reaching global scene understanding and high redundancy, this dissertation presents a sensor fusion framework that combines information from different sensor modalities and preprocessing techniques. In this dissertation, a thorough investigation of the advantages of fusing image instance segmentation features for a better depth completion pipeline is presented. Many other promising directions can be explored in the future. For example, panoptic segmentation can be used instead of instance segmentation since it provides richer information. However, it has a higher computation complexity and hence careful computing optimization is a must. Other works can use this dissertation’s framework to include various feature fusion strategies maximizing benefits from the many AV deployed sensors to propel perception and scene understanding.

BIBLIOGRAPHY

[1] J. Kocic, N. Jovicic, and V. Drndarevic, "Sensors and Sensor Fusion in Autonomous Vehicles," *2018 26th Telecommun. Forum, TELFOR 2018 - Proc.*, no. February, 2018, doi: 10.1109/TELFOR.2018.8612054.

[2] C. Badue *et al.*, "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, no. July 2020, p. 113816, 2021, doi: 10.1016/j.eswa.2020.113816.

[3] M. Hirz and B. Walzel, "Sensor and object recognition technologies for self-driving cars," *Comput. Aided. Des. Appl.*, vol. 15, no. 4, pp. 501–508, 2018, doi: 10.1080/16864360.2017.1419638.

[4] R. Yee, E. Chan, B. Cheng, and G. Bansal, "Collaborative Perception for Automated Vehicles Leveraging Vehicle-to-Vehicle Communications," *IEEE Intell. Veh. Symp. Proc.*, vol. 2018-June, no. Iv, pp. 1099–1106, 2018, doi: 10.1109/IVS.2018.8500388.

[5] SAE, "Shuttleworth, J. SAE Standard News: J3016 Automated-Driving Graphic Update. 2019," 2019. https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic (accessed Nov. 03, 2022).

[6] J. S. Choksey and C. Wardlaw, "Levels of Autonomous Driving, Explained," *jdpower*, 2021. https://www.jdpower.com/cars/shopping-guides/levels-of-autonomous-driving-explained (accessed Nov. 03, 2022).

[7] N. Goberville *et al.*, "Analysis of LiDAR and Camera Data in Real-World Weather Conditions for Autonomous Vehicle Operations," *SAE Tech. Pap.*, vol. 2020-April, no. April, pp. 1–7, 2020, doi: 10.4271/2020-01-0093.

[8] E. Khatab, A. Onsy, M. Varley, and A. Abouelfarag, "Vulnerable objects detection for autonomous driving: A review," *Integration*, vol. 78, no. March 2020, pp. 36–48, 2021, doi: 10.1016/j.vlsi.2021.01.002.

[9] and E. K. Alexandros Gazis, Evangelos Ioannou, "Examining the sensors that enable self-driving vehicles," *IEEE Potentials 39*, no. 1, pp. 46–51, 2019.

[10] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier, "An overview of depth cameras

and range scanners based on time-of-flight technologies," *Mach. Vis. Appl.*, vol. 27, no. 7, pp. 1005–1020, 2016, doi: 10.1007/s00138-016-0784-4.

[11] R. Fan, J. Jiao, H. Ye, Y. Yu, I. Pitas, and M. Liu, "Key Ingredients of Self-Driving Cars," 2019, [Online]. Available: http://arxiv.org/abs/1906.02939.

[12] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse LiDAR data with depth-normal constraints," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 2811–2820, 2019, doi: 10.1109/ICCV.2019.00290.

[13] X. Ma, Z. Geng, and Z. Bie, "Depth Estimation from Single Image Using CNN-Residual Network,"2017, [Online]. Available: http://cs231n.stanford.edu/reports/2017/pdfs/203.pdf.

[14] N. National Center for Statistics and Analysis and H. T. S. Administration, "Early Estimates of Motor Vehicle Traffic Fatalities And Fatality Rate by Sub-Categories in 2021," no. May, 2022.

[15] J. Zhao, B. Liang, and Q. Chen, "The key technology toward the self-driving car," *Int. J. Intell. Unmanned Syst.*, vol. 6, no. 1, pp. 2–20, 2018, doi: 10.1108/IJIUS-08-2017-0008.

[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research," *Int. J. Rob. Res.*, no. October, pp. 1–6, 2013.

[17] Y. Choi *et al.*, "KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 934–948, 2018, doi: 10.1109/TITS.2018.2791533.

[18] D. J. Yeong, G. Velasco-hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, pp. 1–37, 2021, doi: 10.3390/s21062140.

[19] M. Beer, J. F. Haase, J. Ruskowski, and R. Kokozinski, "Background light rejection in SPAD-based LiDAR sensors by adaptive photon coincidence detection," *Sensors (Switzerland)*, vol. 18, no. 12, 2018, doi: 10.3390/s18124338.

[20] Mapix, "velodyne-hdl64." https://www.mapix.com/lidar-scanner sensors/velodyne/velodyne-hdl64/.

[21] H. H. Meinel and W. Bösch, “Radar Sensors in Cars,” in *Automated Driving: Safer and More Efficient Future Driving*, D. Watzenig and M. Horn, Eds. Cham: Springer International Publishing, 2017, pp. 245–261.

[22] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and M. Lienkamp, “A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection,” *2019 Symp. Sens. Data Fusion Trends, Solut. Appl. SDF 2019*, 2019, doi: 10.1109/SDF.2019.8916629.

[23] C. Badue *et al.*, “Self-driving cars: A survey,” *Expert Syst. Appl.*, vol. 165, no. August 2020, p. 113816, 2021, doi: 10.1016/j.eswa.2020.113816.

[24] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, “Deep learning for object detection and scene perception in self-driving cars : Survey , challenges , and open issues,” *Array*, vol. 10, no. February, p. 100057, 2021, doi: 10.1016/j.array.2021.100057.

[25] L. Liu *et al.*, “Deep Learning for Generic Object Detection: A Survey,” *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020, doi: 10.1007/s11263-019-01247-4.

[26] C. Premebida, L. Garrote, A. Asvadi, A. P. Ribeiro, and U. Nunes, “High-resolution LIDAR-based Depth Mapping using Bilateral Filter,” pp. 2469–2474, 2016.

[27] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, “Sparsity Invariant CNNs,” *Proc. - 2017 Int. Conf. 3D Vision, 3DV 2017*, pp. 11–20, 2018, doi: 10.1109/3DV.2017.00012.

[28] M. Felsberg and M. Persson, “Uncertainty-Aware CNNs for Depth Completion : Uncertainty from Beginning to End,” pp. 12014–12023, 2020.

[29] N. Chodosh, C. Wang, and S. Lucey, “arXiv : 1803 . 08949v1 [cs . CV] 23 Mar 2018 Deep Convolutional Compressed Sensing for LiDAR Depth Completion,” *Asian Conf. Comput. Vis.*, no. 1, pp. 1–16, 2018.

[30] M. A. U. Khan *et al.*, “A Comprehensive Survey of Depth Completion Approaches,” *Sensors*, vol. 22, no. 18, pp. 1–18, 2022, doi: 10.3390/s22186969.

[31] F. Ma and S. Karaman, “Sparse-to-Dense : Depth Prediction from Sparse Depth Samples and a Single Image,” pp. 4796–4803, 2018.

[32] F. Ma, G. V. Cavalheiro, and S. Karaman, “Self-Supervised Sparse-to-Dense : Self-

Supervised Depth Completion from LiDAR and Monocular Camera," pp. 3288–3295, 2019.

[33] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong, "PENet: Towards Precise and Efficient Image Guided Depth Completion," pp. 13656–13662, 2021, doi: 10.1109/icra48506.2021.9561035.

[34] J. Qiu *et al.*, "DeepLiDAR : Deep Surface Normal Guided Depth Prediction for Outdoor Scene from Sparse LiDAR Data and Single Color Image," *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 3313–3322, 2019.

[35] D. Neven and K. U. Leuven, "Sparse and noisy LiDAR completion with RGB guidance and uncertainty," *Int. Conf. Mach. Vis. Appl.*, pp. 1–6, 2019.

[36] X. Xiong, H. Xiong, K. Xian, C. Zhao, Z. Cao, and X. Li, "Sparse-to-Dense Depth Completion Revisited : Sampling Strategy and Graph Construction," *Comput. Vision–ECCV 2020 16th Eur. Conf. Glas.*, pp. 1–18, 2018.

[37] Z. Zou, Z. Shi, Y. Guo, J. Ye, and S. Member, "Object Detection in 20 Years : A Survey," pp. 1–39, 2019.

[38] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," 2001.

[39] N. Dalal, B. Triggs, and D. Europe, "Histograms of Oriented Gradients for Human Detection," 2005.

[40] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A Discriminatively Trained , Multiscale , Deformable Part Model," 2008.

[41] R. Girshick, J. Donahue, T. Darrell, J. Malik, U. C. Berkeley, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, p. 5000, 2014, doi: 10.1109/CVPR.2014.81.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8691 LNCS, no. PART 3, pp. 346–361, 2014, doi: 10.1007/978-3-319-10578-9_23.

[43] R. Girshick, “Fast R-CNN,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[44] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[45] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 936–944, 2017, doi: 10.1109/CVPR.2017.106.

[46] W. Liu *et al.*, “SSD: Single shot multibox detector,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[47] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *arXiv*, pp. 2980–2988, 2017.

[48] R. Padilla, S. L. Netto, and E. A. B. Da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” *Int. Conf. Syst. Signals, Image Process.*, vol. 2020-July, no. July, pp. 237–242, 2020, doi: 10.1109/IWSSIP48289.2020.9145130.

[49] A. Mueed Hafiz and G. Mohiuddin Bhat, “A Survey on Instance Segmentation,” *Int. J. Multimed. Inf. Retr.*, vol. 9, pp. 171–189, 2020.

[50] Y. Furletov, V. Willert, and J. Adamy, “Auditory scene understanding for autonomous driving,” *IEEE Intell. Veh. Symp. Proc.*, vol. 2021-July, pp. 697–702, 2021, doi: 10.1109/IV48863.2021.9575964.

[51] W. Gu, S. Bai, and L. Kong, “A review on 2D instance segmentation based on deep neural networks,” *Image Vis. Comput.*, vol. 120, p. 104401, 2022, doi: 10.1016/j.imavis.2022.104401.

[52] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, “Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes,” *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 961–972, 2018, doi: 10.1007/s11263-018-1070-x.

[53] M. F. Chang *et al.*, “Argoverse: 3D tracking and forecasting with rich maps,” *Proc. IEEE*

Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, pp. 8740–8749, 2019, doi: 10.1109/CVPR.2019.00895.

[54] F. Yu *et al.*, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2633–2642, 2020, doi: 10.1109/CVPR42600.2020.00271.

[55] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5302 LNCS, no. PART 1, pp. 44–57, 2008, doi: 10.1007/978-3-540-88682-2_5.

[56] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 3213–3223, 2016, doi: 10.1109/CVPR.2016.350.

[57] Elektra, "Elektra Datasets," 2016. http://adas.cvc.uab.es/.

[58] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior," *Proc. - 2017 IEEE Int. Conf. Comput. Vis. Work. ICCVW 2017*, vol. 2018-Janua, pp. 206–213, 2017, doi: 10.1109/ICCVW.2017.33.

[59] H. Caesar *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, no. March, pp. 11618–11628, 2020, doi: 10.1109/CVPR42600.2020.01164.

[60] Udacity, "Udacity Dataset," 2016, [Online]. Available: https://github.com/udacity/self-driving-car.

[61] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," *2009 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. CVPR Work. 2009*, vol. 2009 IEEE, pp. 304–311, 2009, doi: 10.1109/CVPRW.2009.5206631.

[62] L. Neumann *et al.*, "NightOwls: A Pedestrians at Night Dataset," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11361 LNCS, pp. 691–705, 2019, doi: 10.1007/978-3-030-20887-5_43.

[63] C. Wojek, S. Walk, and B. Schiele, “Multi-Cue onboard pedestrian detection,” *2009 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. CVPR Work. 2009*, vol. 2009 IEEE, pp. 794–801, 2009, doi: 10.1109/CVPRW.2009.5206638.

[64] D. Xu and Y. Tian, “A Comprehensive Survey of Clustering Algorithms,” *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015, doi: 10.1007/s40745-015-0040-1.

[65] W. K. Loh and Y. H. Park, “A survey on density-based clustering algorithms,” *Lect. Notes Electr. Eng.*, vol. 280 LNEE, pp. 775–780, 2014, doi: 10.1007/978-3-642-41671-2_98.

[66] M. Rapp, M. Barjenbruch, M. Hahn, J. Dickmann, and K. Dietmayer, “Probabilistic ego-motion estimation using multiple automotive radar sensors,” *Rob. Auton. Syst.*, vol. 89, pp. 136–146, 2017, doi: 10.1016/j.robot.2016.11.009.

[67] X. Ester, M., Kriegel, H. P., Sander, J., & Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.

[68] X. Cheng, P. Wang, C. Guan, and R. Yang, “CSPN++: Learning context and resource aware convolutional spatial propagation networks for depth completion,” *AAAI 2020 - 34th AAAI Conf. Artif. Intell.*, pp. 10615–10622, 2020, doi: 10.1609/aaai.v34i07.6635.

[69] M. F. F. Khan, N. D. Troncoso Aldas, A. Kumar, S. Advani, and V. Narayanan, *Sparse to Dense Depth Completion using a Generative Adversarial Network with Intelligent Sampling Strategies*, vol. 1, no. 1. Association for Computing Machinery, 2021.

[70] C. Zhang, Y. Tang, C. Zhao, Q. Sun, Z. Ye, and J. Kurths, “Multitask GANs for Semantic Segmentation and Depth Completion with Cycle Consistency,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 12, pp. 5404–5415, 2021, doi: 10.1109/TNNLS.2021.3072883.

[71] Y. Tsuji, H. Chishiro, and S. Kato, “Non-Guided Depth Completion with Adversarial Networks,” *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2018-Novem, pp. 1109–1114, 2018, doi: 10.1109/ITSC.2018.8569389.

[72] J. Ku, A. Harakeh, and S. L. Waslander, “In defense of classical image processing: Fast depth completion on the CPU,” *Proc. - 2018 15th Conf. Comput. Robot Vision, CRV 2018*, pp. 16–22, 2018, doi: 10.1109/CRV.2018.00013.

[73] M. Hornacek, C. Rhemann, M. Gelautz, and C. Rother, “Depth super resolution by rigid body self-similarity in 3D,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1123–1130, 2013, doi: 10.1109/CVPR.2013.149.

[74] H. A. Wafa, R. Aminuddin, S. Ibrahim, N. N. A. Mangshor, and N. I. F. A. Wahab, “A Data Visualization Framework during Pandemic using the Density-Based Spatial Clustering with Noise (DBSCAN) Machine Learning Model,” *2021 IEEE 11th Int. Conf. Syst. Eng. Technol. ICSET 2021 - Proc.*, no. November, pp. 1–6, 2021, doi: 10.1109/ICSET53708.2021.9612563.

[75] Z. Guo, H. Liu, L. Pang, L. Fang, and W. Dou, “DBSCAN-based point cloud extraction for Tomographic synthetic aperture radar (TomoSAR) three-dimensional (3D) building reconstruction,” *Int. J. Remote Sens.*, vol. 42, no. 6, pp. 2327–2349, 2021, doi: 10.1080/01431161.2020.1851062.

[76] D. Sheng, J. Deng, and J. Xiang, “Automatic Smoke Detection Based on SLIC-DBSCAN Enhanced Convolutional Neural Network,” *IEEE Access*, vol. 9, pp. 63933–63942, 2021, doi: 10.1109/ACCESS.2021.3075731.

[77] X. He, Y. Jiang, B. Wang, H. Ji, and Z. Huang, “An Image Reconstruction Method of Capacitively Coupled Electrical Impedance Tomography (CCEIT) Based on DBSCAN and Image Fusion,” *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, doi: 10.1109/TIM.2021.3056739.

[78] M. J. Sumnall *et al.*, “Effect of varied unmanned aerial vehicle laser scanning pulse density on accurately quantifying forest structure,” *Int. J. Remote Sens.*, vol. 43, no. 2, pp. 721–750, 2022, doi: 10.1080/01431161.2021.2023229.

[79] H. Li, X. Liu, T. Li, and R. Gan, “A novel density-based clustering algorithm using nearest neighbor graph,” *Pattern Recognit.*, vol. 102, 2020, doi: 10.1016/j.patcog.2020.107206.

[80] J. H. Kim, J. H. Choi, K. H. Yoo, and A. Nasridinov, “AA-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities,” *J. Supercomput.*, vol. 75, no. 1, pp. 142–169, 2019, doi: 10.1007/s11227-018-2380-z.

[81] M. M. R. Khan, M. A. B. Siddique, R. B. Arif, and M. R. Oishe, “ADBSCAN: Adaptive

density-based spatial clustering of applications with noise for identifying clusters with varying densities," *4th Int. Conf. Electr. Eng. Inf. Commun. Technol. iCEEiCT 2018*, pp. 107–111, 2019, doi: 10.1109/CEEICT.2018.8628138.

[82] K. Sawant, "Adaptive Methods for Determining DBSCAN Parameters," *IJISET - International J. Innov. Sci. Eng. Technol.*, vol. 1, no. 4, pp. 329–334, 2014, [Online]. Available: www.ijiset.com.

[83] C. Wang, M. Ji, J. Wang, W. Wen, T. Li, and Y. Sun, "An improved DBSCAN method for LiDAR data segmentation with automatic Eps estimation," *Sensors (Switzerland)*, vol. 19, no. 1, 2019, doi: 10.3390/s19010172.

[84] M. Cao and J. Wang, "Obstacle Detection for Autonomous Driving Vehicles With Multi-LiDAR Sensor Fusion," *J. Dyn. Syst. Meas. Control*, vol. 142, no. 2, 2020, doi: 10.1115/1.4045361.

[85] J. Zhao, H. Xu, H. Liu, J. Wu, Y. Zheng, and D. Wu, "Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors," *Transp. Res. Part C Emerg. Technol.*, vol. 100, no. January, pp. 68–87, 2019, doi: 10.1016/j.trc.2019.01.007.

[86] M. Elhousni and X. Huang, "A Survey on 3D LiDAR Localization for Autonomous Vehicles," *IEEE Intell. Veh. Symp. Proc.*, no. Iv, pp. 1879–1884, 2020, doi: 10.1109/IV47402.2020.9304812.

[87] A. Atapour-Abarghouei and T. P. Breckon, "A comparative review of plausible hole filling strategies in the context of scene depth image completion," *Comput. Graph.*, vol. 72, pp. 39–58, 2018, doi: 10.1016/j.cag.2018.02.001.

[88] T. M. Nguyen and M. Yoo, "Wasserstein Generative Adversarial Network for Depth Completion with Anisotropic Diffusion Depth Enhancement," *IEEE Access*, vol. 10, pp. 1–1, 2022, doi: 10.1109/access.2022.3142916.

[89] J. Lambert, E. Takeuchi, and K. Takeda, "Optimizing learned object detection on point clouds from 3D lidars through range and sparsity information," *2019 Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. APSIPA ASC 2019*, no. November, pp. 407–413, 2019, doi: 10.1109/APSIPAASC47483.2019.9023145.

[90] G. T. U. A. Colleges *et al.*, “Microsoft COCO,” *Eccv*, no. June, pp. 740–755, 2014.

[91] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.

[92] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv Prepr. arXiv*, 2014.

[93] C. Szegedy *et al.*, “Going deeper with convolutions,” 2015.

[94] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[95] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, 2020, doi: 10.1109/TPAMI.2018.2844175.

[96] F. Zhuang *et al.*, “A Comprehensive Survey on Transfer Learning,” *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2021, doi: 10.1109/JPROC.2020.3004555.

[97] M. P. Hosseini, S. Lu, K. Kamaraj, A. Slowikowski, and H. C. Venkatesh, *Deep Learning Architectures*, vol. 866. 2020.

[98] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, “3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes,” *Rob. Auton. Syst.*, vol. 83, pp. 299–311, 2016, doi: 10.1016/j.robot.2016.06.007.

[99] C. Jiang *et al.*, “A practical method for employing multi-spectral LiDAR intensities in points cloud classification,” *Int. J. Remote Sens.*, vol. 41, no. 21, pp. 8366–8379, 2020, doi: 10.1080/01431161.2020.1775323.

[100] M. El Yabroudi, K. Awedat, R. C. Chabaan, O. Abudayyeh, and I. Abdel-Qader, “Adaptive DBSCAN LiDAR Point Cloud Clustering For Autonomous Driving Applications,” *IEEE Int. Conf. Electro Inf. Technol.*, vol. 2022-May, pp. 221–224, 2022, doi: 10.1109/eIT53891.2022.9814025.

[101] B. Xiang, J. Yao, X. Lu, L. Li, R. Xie, and J. Li, “Segmentation-based classification for 3D point clouds in the road environment,” *Int. J. Remote Sens.*, vol. 39, no. 19, pp. 6182–6212, 2018, doi: 10.1080/01431161.2018.1455235.

[102] S. Shi, X. Wang, and H. Li, “PointRCNN: 3D object proposal generation and detection from point cloud,” *arXiv*, pp. 770–779, 2018.

[103] X. Chang, H. Li, J. Rong, X. Chen, and Y. Wang, “Determining the appropriate lane width at urban signalised intersections - A case study in Beijing,” *IET Intell. Transp. Syst.*, vol. 13, no. 12, pp. 1785–1791, 2019, doi: 10.1049/iet-its.2018.5401.

[104] D. Hawley, “Dimensions and Weights of Common SUVs,” 2021. https://www.jdpower.com/cars/shopping-guides/dimensions-and-weights-of-common-suvs.

[105] M. Menze, C. Heipke, and A. Geiger, “Joint 3D estimation of vehicles and scene flow,” *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 2, no. 3W5, pp. 427–434, 2015, doi: 10.5194/isprsannals-II-3-W5-427-2015.