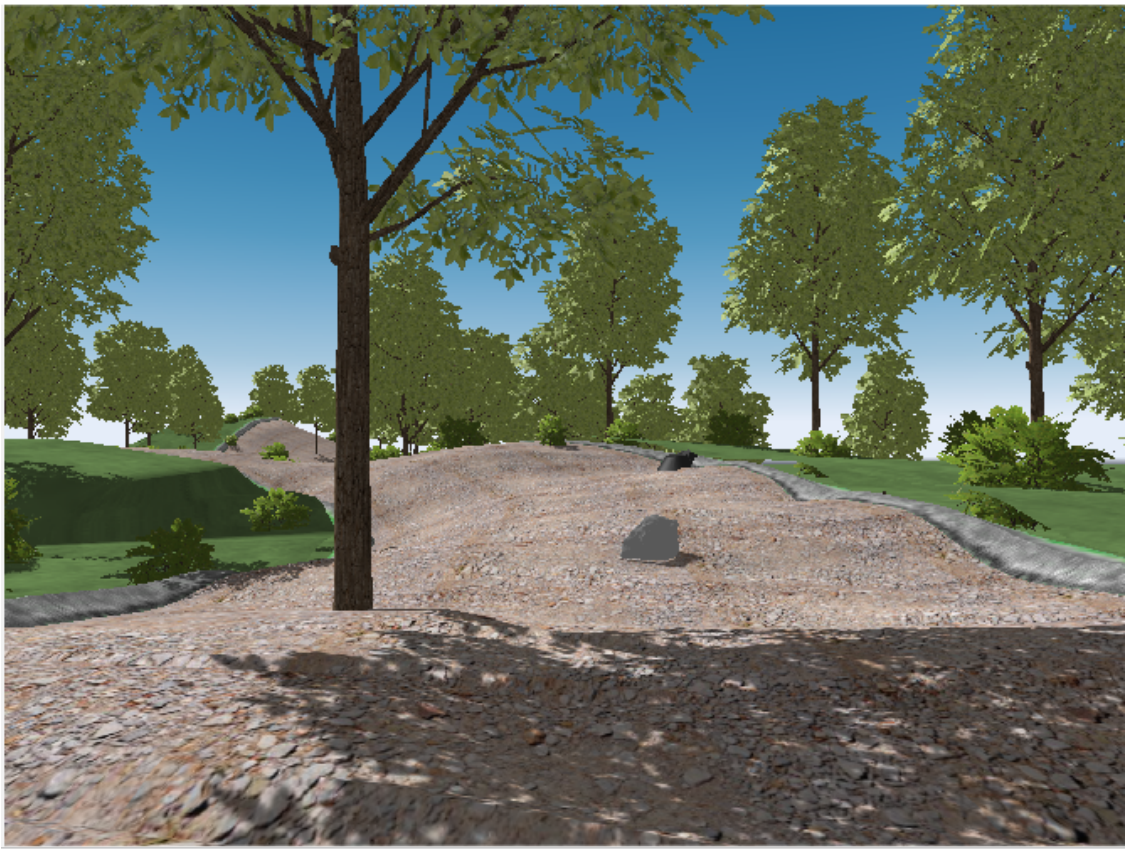
CHALMERS
UNIVERSITY OF TECHNOLOGY



Rough Road Ahead Off-road Terrain Detection Through Enhanced Sensor Fusion

Master Thesis Report

Mostafa Husseini and Poornesh Velineni

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Rough Road Ahead Off-road Terrain Detection Through Enhanced Sensor Fusion

Enhanced sensor fusion detects rough road terrain and obstacles, providing a detailed and reliable terrain illustration.

Mostafa Husseini and Poornesh Velineni

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2024

Rough Road Ahead Off-road Terrain Detection Through Enhanced Sensor Fusion
Enhanced sensor fusion detects rough road terrain and obstacles, providing a detailed and reliable terrain illustration.
Mostafa Husseini Poornesh Velineni

Supervisor: Karthik Prasad, Chief Product Owner, Zeekr Technology Europe
Examiner: Lars Hammarstrand, Department of Electrical Engineering, Chalmers

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image of off-road scenario created in Carmaker IPG Simulation software.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Abstract

This thesis examines the development of the Automatic Terrain Detection and Adaptation (ATDA) feature for off-road vehicles. The ATDA system is designed to improve vehicle safety, efficiency, and comfort in demanding off-road environments characterized by uneven terrain, diverse obstacles, and unpredictable road types and conditions.

Traditional methods for detecting off-road terrain and obstacles, which rely on either image-based or point cloud-based models, have proven insufficient for accurate predictions. To overcome these limitations, this research integrates camera-based and point LiDAR-based methodologies through a sensor fusion approach. This integrated method is anticipated to yield more accurate results, significantly improving the vehicle's navigation capabilities in complex off-road terrains.

Aligned with the latest trends in sensor fusion for environmental perception in autonomous vehicles, this project prioritizes evaluating path traversability using static vehicle features rather than the traditional focus on obstacle density. This innovative approach enhances the vehicle's ability to detect and assess off-road terrain and obstacles, thereby improving navigation safety and efficiency. With an emphasis on the integration of camera and LiDAR technologies for enhanced terrain and obstacle detection and classification.

Keywords: ATDA, Off-Road, Camera, Image data, LiDAR, Point cloud, Sensor Fusion, path traversability.

Acknowledgements

We wish to extend our heartfelt gratitude to our esteemed tutors, Karthik Prasad (supervisor) and Utsav Khan at Zeeker Technology Europe. Their unwavering guidance, support, and collaborative approach were indispensable throughout our academic journey. Their continuous assistance and invaluable insights played a crucial role in shaping the outcome of our research endeavors.

We also express our deep appreciation to Prof. Lars Hammarstrand at Chalmers University of Technology, our esteemed examiner. His invaluable guidance and scholarly input significantly enriched the quality of our project. We are profoundly grateful for his steadfast support.

In addition, we acknowledge the IPG Automotive GmbH Corporation (Carmaker Simulation Tool) team for their valuable support and time during the initial phase of our thesis work.

Lastly, we convey our sincere gratitude to our families and friends. Their unwavering support and encouragement have been a constant source of strength throughout our academic journey. Without their steadfast belief in our endeavors, this work would not have been possible.

Mostafa Husseini and Poornesh Velineni, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ATDA	Automatic Terrain Detection and Adaption
mIoU	Mean Intersection Over Union
PMF	Perception-aware Multi-sensor Fusion
RGB	Red Green Blue
CNN	Convolutional Neural Network
PSPNet	Pyramid Scene Parsing Network
ResNet	Residual Network
VGG16	Visual Geometry Group with 16 layers
AWV	accuracy-based weighted voting
RRT	Rapidly-exploring Random Tree
FOV	Field Of View
RSI	Raw Signal Interface
LOF	Length OF Flight
FOVH	Field Of View Horizontal
FOVV	Field Of View Vertical
GUI	Graphical User Interface
CCA	Connected Component Analysis

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

c	Index for the camera
l	Index for lidar
o	Index for the origin
w	Index for Width

Parameters

u_0	Principal-axis of the camera in x direction
v_0	Principal-axis of the camera in x direction
Img_w	Camera image width
Img_h	Camera image height
Fov_v	Camera or lidar vertical field of view
Fov_h	Camera or lidar horizontal field of view
f_x	Camera focal lenght in x direction
f_y	Camera focal length in y direction
$\mathbf{K}$	Camera calibration matrix
T_l	Translation vector for the lidar coordination frame with respect to origin coordinate frame
T_c	Translation vector for the camera coordination frame with respect to origin coordinate frame
T_c^l	Translation vector for the lidar coordination frame with respect to camera coordinate frame

H_o^c	Homogeneous transformation matrix of the camera with respect the origin coordinate frame
H_o^l	Homogeneous transformation matrix of the lidar with respect the origin coordinate frame
H_c^o	Homogeneous transformation matrix of the origin with respect the camera coordinate frame
H_c^l	Homogeneous transformation matrix of the lidar with respect the camera coordinate frame
R_c	Rotation matrix of the camera with respect the origin coordinate frame
R_l	Rotation matrix of the lidar with respect the origin coordinate frame
R_c^l	Rotation matrix of the camera with respect the lidar coordinate frame
T_2	Prediction weight of the 2D probabilities used in the fusion prediction
T_3	Prediction weight of the 3D probabilities used in the fusion prediction

Variables

x	Cartesian coordination of the lidar point in x direction[m]
y	Cartesian coordination of the lidar point in y direction[m]
z	Cartesian coordination of the lidar point in z direction[m]
r	range of a lidar point
θ	Azimuth angle of a lidar point
ϕ	Elevation angle of a lidar point
$\mathbf{P}$	Cartesian coordination of the lidar point
$\mathbf{p}$	2D coordination of the projected lidar point to the image plane
$\mathbf{P}'$	Cartesian coordination of the lidar point with respect to camera coordinate system
P_2	Predicted image of the 2D model consisting class probabilities
P_3	Predicted image of the 3D model consisting class probabilities
A	2D coordination of the left down corner of obstacle box
B	2D coordination of the right down corner of obstacle box
C	2D coordination of the right top corner of obstacle box
D	2D coordination of the left top corner of obstacle box

E	2D coordination of the center of obstacle box
A'	2D coordination of the left down corner of obstacle box translated to the origin
B'	2D coordination of the right down corner of obstacle box translated to the origin
C'	2D coordination of the right top corner of obstacle box translated to the origin
D'	2D coordination of the left top corner of obstacle box translated to the origin
E'	2D coordination of the center of obstacle box translated to the origin
A''	2D coordination of the left down corner of obstacle box translated to the origin and rotated
B''	2D coordination of the right down corner of obstacle box translated to the origin and rotated
C''	2D coordination of the right top corner of obstacle box translated to the origin and rotated
D''	2D coordination of the left top corner of obstacle box translated to the origin and rotated
E''	2D coordination of the center of obstacle box translated to the origin and rotated
A'''	2D coordination of the left down corner of obstacle box rotated and translated back
B'''	2D coordination of the right down corner of obstacle box rotated and translated back
C'''	2D coordination of the right top corner of obstacle box rotated and translated back
D'''	2D coordination of the left top corner of obstacle box rotated and translated back
E'''	2D coordination of the center of obstacle box rotated and translated back
$\overrightarrow{V_1}$	Vector from nearest point to the random point
$\overrightarrow{V_2}$	Vector from nearest point to the its parent point

Contents

List of Figures

List of Tables

1
Introduction

The Automatic Terrain Detection and Adaptation (ATDA) system is an advanced Vehicle feature designed specifically for off-road driving. Detecting off-road terrain types and obstacles [1] is essential for both autonomous and manual vehicles, ensuring significantly safer and more efficient operations in challenging environments. ATDA enables vehicles to automatically identify various terrain types, such as mud, sand, gravel, or snow, and calculates the safest path for navigation, helping to avoid unforeseen obstacles.

The research and development efforts are ongoing. Initially, the approach focused on using either image-based or point cloud-based models for terrain detection, but these methods did not provide sufficiently accurate predictions. To improve accuracy, current research is now integrating both camera-based and LIDAR-based techniques. This fusion of methodologies is expected to produce significantly more precise and reliable results in terrain detection and navigation.

1.1 Background

This research work holds significant potential in addressing the intricate challenges presented by off-road terrain, characterized by its uneven surfaces, terrain type, and numerous obstacles such as positive or hanging obstacles. Developing technology that can accurately identify and assess various terrain type and obstacles is essential for enhancing the safety and efficiency of off-road navigation. This advancement is closely linked with the progress in autonomous vehicle technology and off-road capabilities, representing a cutting-edge approach in the field. Such innovations are crucial for improving vehicle performance in challenging environments.
The thesis capitalizes on the fusion of camera and LiDAR technologies to detect and classify terrain and obstacles lying ahead of the vehicle. which aligns with research trends in sensor fusion for environmental perception in autonomous vehicles [2]. Furthermore, this technology's emphasis on evaluating path traversability using static vehicle features instead of traditional methods that prioritize obstacle density in path planning demonstrates its innovative nature.

1.2 Related Work

Navigating off-road terrain has become increasingly challenging in contemporary times due to factors such as uneven surfaces, terrain conditions, obstacles, and the

complexities of vehicle dynamics.

The research article [3] offers a comprehensive overview of terrain detection methods for autonomous ground vehicles (AGVs) that employ various sensor combinations. The article emphasizes that detection is a crucial capability for AGVs, particularly in off-road environments where challenges are more pronounced compared to on-road settings. The authors categorize detection into three major components: drivable ground, positive obstacles, and negative obstacles. Despite numerous studies focusing on individual aspects of detection—such as sensing technologies, learning algorithms, and obstacle identification—the article highlights that no existing approach fully integrates all these critical elements into a cohesive solution for off-road environments. The article discusses the use of various sensors, including LiDAR, cameras, radar, infrared, lasers, and stereo vision, along with advanced algorithms like CNNs, supervised and unsupervised learning, SVMs, and deep learning. Although significant advancements have been made, challenges remain, including issues like sensor alignment, false detection, the complexity of real-time analysis, and environmental difficulties. The article points out the potential for improving detection performance through sensor fusion, artificial intelligence, remote sensing, and new algorithms, while also addressing computational complexity. The authors call for further research to overcome these challenges, emphasizing the importance of real-time analysis, 3D object detection, and V2X connectivity in enhancing off-road detection systems. They believe their work will serve as a valuable resource for researchers pursuing similar advancements in AGV detection technologies.

The article [4] introduces an efficient method for obstacle detection and simultaneous identification of traversable regions in off-road terrains, addressing challenges posed by unpredictable surfaces and vehicle dynamics. The approach relies on high-definition 3D Lidar points, incorporating both radial and transverse features.The methodology involves systematic processing of Lidar points through simultaneous scanning lines and sectors within the polar system. Radial features enable rapid obstacle identification, while transverse features are used to validate detections and reduce false positives. The resulting delineation of constrained regions, comprising the nearest obstacle points within each sector, defines the traversable area around the vehicle. Notably, the method excels in real-time detection of positive, negative, and hanging obstacles. Experimental results highlight the robustness and precision of the proposed method across diverse off-road environments. This approach signifies a significant advancement in enhancing the efficiency and reliability of obstacle detection and traversable region determination, demonstrating its practical applicability in real-world scenarios.

The article [5] explain about the development of Off-road detection algorithm for autonomous navigation. Two pieces make up the environment detection algorithm: 1) Detection by eyesight; 2) Detection through LIDAR. The detection of speed bumps, pedestrian crossings, and driving lanes have all been accomplished with vision-based detection systems. The VELD module has evaluated the information the color camera has detected regarding the driving lane. To estimate the current lane without

misidentification or loss, the lane filter was created. Obstacles have been identified by LIDAR detection. The inability to use precise vehicle position data led to the algorithm's decision to create a local obstacle map rather than a global one.

The article [6] explains about multi-sensor fusion technology. This innovative approach enhances navigation and localization in challenging off-road environments, such as rough terrains, by amalgamating data from various sensors. The focal point is the utilization of a deep learning architecture for the identification of drivable and obstacle regions in images, employing semantic segmentation to simultaneously classify and cluster regions. To further improve the effectiveness of drivable region classification, a LiDAR-based ground segmentation method is introduced. This method involves splitting regions into smaller bins and applying a ground fitting technique with adaptive likelihood estimation. A late fusion method is then proposed to seamlessly combine the results from both semantic segmentation and LiDAR-based ground segmentation, offering a more comprehensive classification of the drivable region.

The article [7], describes the the PointPainting method is a novel sequential fusion technique is described. It paints lidar point clouds with semantics based on images. On the KITTI and nuScenes competitions, PointPainting achieves state-of-the-art results using several different lidar networks. Because of its adaptability, the PointPainting framework can integrate the outputs of any lidar network and segmentation network. When combining picture and lidar data for 3D object detection, PointPainting is the best architecture, as shown by the strength of these results and their broad applicability.

The research article [8] investigates LiDAR and camera sensor fusion for semantic segmentation using deep learning transformers DeiT. It explores various fusion levels and finds slight performance improvement over LiDAR-only models in mean Intersection over Union (mIoU) across 19 classes. Despite potential overfitting to small datasets like SemanticKitti, fusion models demonstrate benefits in feature extraction. Challenges such as class imbalance, particularly with the motorcyclist class, are noted. Future work entails larger dataset training and different fusion architectures to enhance performance. Overall, the research underscores the potential of deep learning fusion models, especially middle-level fusion, in advancing LiDAR point cloud segmentation, indicating the importance of image-based transformers in fusion frameworks.

The reserach article [9] has focused on employing fusion-based strategies to improve semantic segmentation. One noteworthy study explores 3D LiDAR semantic segmentation using a perception-aware multi-sensor fusion (PMF) method. This method, in contrast to previous approaches, projects point clouds onto the camera coordinate system in order to combine perceptual data from RGB images. By using residual-based fusion modules and a two-stream network, PMF effectively combines data from both modalities, improving segmentation accuracy. The efficiency of PMF is validated by empirical results on benchmark datasets, which also suggest

intriguing areas for future research in difficult tasks like as object identification in auto-driving scenarios.

1.3 Aim

Vision: The primary objective of this thesis is to develop the Automatic Terrain Detection and Adaptation (ATDA) feature to enhance vehicle performance in off-road conditions. This involves designing a robust system that can accurately detect and assess various off-road terrain types, surface conditions, and variant obstacles over the terrain, ensuring safer and more efficient navigation in challenging environments.

Acceptable Outcome (Minimum Requirement): At a minimum, the project aims to deliver a functional prototype of the ATDA feature that demonstrates the capability to detect and classify off-road terrain and obstacles in front of the vehicle using camera and Lidar technologies. The prototype should be able to provide binary assessments of path traversability based on static vehicle characteristics, indicating whether a given path is suitable for straight-line navigation without directional changes. Additionally, the acceptable outcome includes the development of a simulation environment for testing and validation purposes, ensuring the reliability and accuracy of the ATDA feature under various off-road conditions.

1.4 Objective

1.4.1 Research Questions

- What are the strengths and limitations of existing off-road detection algorithms, specifically those utilizing vision-based detection systems and Lidar technology?

- How can a multi-sensor fusion approach, incorporating deep learning and semantic segmentation and Lidar technologies contribute to the identification of irregular surfaces, and categorization of obstacles for off-road traversability?

- In the context of off-road conditions, how can a binary assessment model be developed to determine the traversability of a path, based on constraints and vehicle properties?

1.5 Limitation

In this research, we will focus exclusively on the navigability of a given path without addressing the dynamic aspects of the vehicle. The primary goal is to determine how well a vehicle can traverse on uneven terrain with numerous obstacles. In this scenario, the vehicle will operate in a static mode, conducting a brief scan of the environment using a camera and a LiDAR sensor mounted on top. Based on this scan,

the vehicle will make decisions considering its physical attributes, such as height, width, and length.

A significant limitation of this work is, the unavailability of a dataset that combines both camera and LiDAR sensor data for off-road environments. To overcome this challenge, an off-road scenario was simulated using the IPG-Carmaker tool to generate the necessary dataset for training, validating, and testing the model. In this simulation, a vehicle equipped with a camera and LiDAR sensor was driven through created off-road environment scenario to collect the essential data from both sensors.

The IPG simulation tool offers a robust platform for creating customized scenarios with all the necessary sensor configurations. However, it has certain limitations, such as the inability to replicate the real-time texture of off-road terrain, it has limited off-road terrain objects and the data transfer rate, which prevents it from achieving a completely authentic off-road experience. The simulation tool allows only the attachment of an image representing the road type to the surface, which means that changing the road type does not alter the complete physical properties of the road, such as unevenness, reflection rate, friction, etc. Consequently, while the camera can detect different surface types, the lidar perceives all these different road types with the same behaviour and properties. To address this issue, road type classification relies solely on camera observations. Instead of creating a single model that takes both camera and lidar data as input and outputs object class predictions, two separate models were trained. The first model uses camera images as input and classifies the object classes, especially the road type, producing an output image. The second model uses lidar data as input and provides a classified object image as output, excluding road type and sky. Finally, these two predictions are fused to generate a more robust and accurate predicted image. It is important to note that this research will not address the real properties of the objects, such as their shape and friction.

2

Theory

This chapter delves into the foundational concepts crucial to our thesis, particularly focusing on off-road terrain and obstacle detection. We elucidate key methodologies employed to address these challenges, including the formulation of our dataset, the theoretical underpinnings of 2D and 3D semantic segmentation, and our approach to prediction analysis. While our discussion may be concise, we offer references to pertinent literature for those keen on further exploration.

2.1 Off-road Terrain and Obstacle Detection

Off-road environments encompass regions that extend beyond suburban or non-urban boundaries, typically lacking structured roadways and infrastructural elements. These areas are generally devoid of well-defined routes, road signs, and traffic signals, which are standard features in developed areas. Examples of off-road environments include forests, rural roads, muddy or sandy tracks, and terrains covered by dense vegetation as shown in 2.1

Figure 2.1: Off-Road Scenarios

In essence, off-road environments are settings that lack essential driving facilities and clear instructions, presenting conditions that are significantly more challeng-

ing than conventional driving environments. These areas do not provide the basic amenities such as paved roads, signage, or predictable pathways, thereby demanding advanced navigation and maneuvering skills.

Biao Gao et al.[10] Off-road environments often require vehicles to traverse uneven, unstable, or obstructed surfaces, necessitating robust and adaptable navigation systems . The lack of predefined routes and traffic management infrastructure in these areas means that vehicles must be capable of interpreting and responding to dynamic and unpredictable conditions. Consequently, navigating off-road environments involves a higher level of complexity and adaptability, making it a critical area of study for developing ATDA.

2.2 Dataset Development

Dataset development is a crucial step in the creation and validation of the ATDA model. In our research, we utilize the IPG Carmaker simulation tool to generate the necessary datasets. Based on the methodologies detailed by Peng Jiang et al. [11], we have created a new dataset comprising image data and LiDAR point clouds. This methodology employs semantic segmentation techniques to enhance the volume and quality of off-road datasets, thus promoting further research and development in this field. Furthermore, Suvash Sharma et al. [12] provide an in-depth description of the off-road environment, highlighting the numerous uncertainties inherent in off-road driving. These uncertainties include uneven terrain, positive and negative obstacles, ditches, quagmires, and hidden objects, all contributing to the highly unstructured nature of such environments. The ability to traverse these areas is dependent on the vehicle's type and capabilities.

In accordance with the insights from these articles, we have designed various off-road scenarios within the IPG Carmaker simulation tool. Using these scenarios, we collected, streamed, and pipelined image data from camera sensors and point clouds from LiDAR sensors, forming the basis for our dataset development efforts.

2.3 2D semantic segmentation

2D semantic segmentation is a fundamental technique in the field of computer vision that focuses on the detailed interpretation of an image by partitioning it into distinct regions, each corresponding to different objects or classes. This method contrasts with traditional image classification, which assigns a single label to an entire image, offering only a broad understanding of the scene. In semantic segmentation, every pixel in the image is classified into a specific category, resulting in a comprehensive pixel-level understanding of the scene.

The process begins with the analysis of the image, where advanced algorithms and models, often based on deep learning architectures such as Convolutional Neural Networks (CNNs), are employed to identify and categorize each pixel. These mod-

els are trained on large datasets containing images with annotated pixel-level labels. The training process involves learning the intricate features and patterns that distinguish different classes, enabling the model to accurately segment new, unseen images.

2.3.1 2D-Segmentation: Network Architecture

In off-road environments, semantic segmentation encounters formidable challenges owing to the intricate and dynamic nature of such settings. The presence of diverse natural elements, coupled with variations in terrain and lighting conditions, poses significant hurdles to achieving accurate segmentation. As noted by Youngsaeng Jin et al. [13], the evolution of semantic segmentation methods has been closely tied to the advancement of Convolutional Neural Networks (CNNs), particularly leveraging deep architectures like ResNet and ResNext as encoders for robust feature extraction.

In pursuit of precise segmentation masks, researchers have introduced various sophisticated decoder modules. While early methods laid the groundwork, their simplicity constrained their efficacy. Modern approaches, exemplified by PSPNet and Deeplab, have elevated performance by integrating multi-scale contextual information using spatial pyramid modules. Additionally, techniques employing dense layers facilitate feature reuse, while attention mechanisms enable capturing global context.

To delve deeper into these techniques, our proposed methodology involves the development of a novel architecture of U-NET [14] and PSPNet (Pyramid Scene Parsing Network) [15], [16]. In which ,U-Net utilizes a CNN architecture consisting of a contracting path (encoder) and an expansive path (decoder). The contracting path captures context information through convolutional and pooling layers, while the expansive path enables precise localization through upsampling operations. where as, PSPNet captures global context information at multiple scales by using a pyramid pooling module. This helps in better understanding the scene and improves segmentation accuracy. All those network architecture models are trained with the categorical cross-entropy loss and Adam Optimizer

Through rigorous analysis and performance evaluation, we aim to glean insights into the effectiveness of these approaches. By assessing metrics such as accuracy, precision, recall, and computational efficiency, we endeavor to draw conclusive findings regarding the efficacy of our proposed architecture compared to existing methodologies. This research endeavors to contribute to the advancement of semantic segmentation in challenging off-road environments, thereby enhancing the applicability of computer vision systems in real-world scenarios.

2.3.2 Semantic segmentation: Backbone

In the context of semantic segmentation, the backbone is a fundamental component of the neural network architecture, primarily responsible for extracting features from

the input image. Utilizing an effective backbone is crucial for enhancing both accuracy and performance of the segmentation model. To achieve improved accuracy and performance, we intend to implement widely-used backbones such as Residual Network (ResNet) and Visual Geometry Group with 16 layers (VGG16) in our segmentation model. These backbones have been extensively validated in the literature [17] and are known for their robustness in feature extraction and contribution to the overall effectiveness of the model.

2.4 3D Semantic segmentation

3D semantic segmentation is a fundamental technique focuses on the detailed interpretation of a 3D scene by partitioning it into distinct regions, each corresponding to different objects or classes. This method contrasts with traditional 3D object classification, which assigns a single label to an entire object or scene, offering only a broad understanding of the environment. In 3D semantic segmentation, every point in the 3D space is classified into a specific category, resulting in a comprehensive point-level understanding of the scene.

The process begins with the analysis of the 3D data, where advanced algorithms and models, often based on deep learning architectures such as Convolutional Neural Networks (CNNs) and more specialized networks like PointPainting, Squeezeseg V2, Point-Net and its variants, are employed to identify and categorize each point. These models are trained on large datasets containing 3D data with annotated point-level labels. The training process involves learning the intricate features and patterns that distinguish different classes, enabling the model to accurately segment new, unseen 3D scenes.

2.4.1 3D-Segmentation: Network architecture

The most commonly used 3D semantic segmentation technique has been PointPainting [7] ,[18], which enhances 3D object detection by integrating semantic information from images into point clouds. This method uses semantic segmentation masks derived from RGB images and applies them to point clouds. The semantic labels from the 2D image are projected onto the 3D point cloud, effectively "painting" each point with its corresponding label.

However, due to constraints related to our data sources, computation unit and simulation environment. we have opted to implement an alternative technique known as SqueezeSegV2. This technique processes point clouds as input and generates an output image with five channels. SqueezeSegV2 is designed to improve the efficiency and accuracy of semantic segmentation in 3D point clouds by leveraging this multi-channel output format.

SqueezeSegV2 [19] is an advanced technique for 3D semantic segmentation that builds upon its predecessor, SqueezeSeg, with enhancements in model accuracy and robustness. Specifically designed to process LiDAR point cloud data, SqueezeSegV2

uses a convolutional neural network (CNN) architecture to efficiently handle 3D data by transforming it into a 2D representation. The model was trained with the focal cross-entropy loss and Adam Optimizer.

The process begins with an input transformation: the raw point cloud data is projected onto a spherical surface to create a range image. This range image includes five channels: range, intensity, and three Cartesian coordinates (x, y, z). The five-channel range image is processed using a CNN that extracts features through a series of convolutional layers, designed to capture spatial relationships and patterns within the data. The network refines these features and outputs a segmented image, where each pixel is classified into specific categories corresponding to different object classes or environmental features.

2.5 Fusion technique

Advancements in Convolutional Neural Networks (CNNs) and deep ensemble learning techniques have greatly enhanced image recognition tasks, offering essential aid to millions of individuals globally who experience hearing loss. A prominent illustration of this is the ASL Recognition System utilizing multiple deep CNNs and accuracy-based weighted voting (AWV). This system includes stages such as data preprocessing, feature extraction, and classification. It employs ensemble learning with various CNN architectures—LeNet, AlexNet, VGGNet, GoogleNet, and ResNet—to extract features. The extracted features are then used to generate new datasets for classification. AWV algorithm used in this system gives precedence to models with higher accuracy, thereby improving overall performance. Remarkably, ARS-MA achieved accuracies of 98.83% and 98.79% on the ASL Alphabet and ASLA datasets, respectively[20].
Inspired by the ARS-MA methodology, a comparable approach can be employed to fuse semantic segmentation predictions from two models through accuracy-based weighting. This technique ensures that the predictions from both models are taken into account, resulting in a more robust and accurate final prediction.

2.6 Rapidly-exploring Random Tree Star(RRT*)

The path planning algorithm for moving objects is highly complex and lacks sufficient automatic path planning capabilities to handle intricate real-world environments. To overcome these limitations, a rapid path planning algorithm based on RRT-Star is required [21]. RRT is a path planning method that utilizes random sampling [22]. The algorithm efficiently avoids obstacles during movement and aims to optimize for criteria such as shortest distance, minimal time, or lowest energy consumption. Due to its high search speed and lack of preprocessing requirements, it is widely used. Rather than modeling the environment, the algorithm generates a random tree by exploring the entire space through random sampling[23].
The first step of the algorithm is to predefine certain parameters:

- **Start point**: The current position of the vehicle.

- **Goal point**: The target destination of the vehicle.
- **Step size**: The distance between each node.
- **Search radius**: Defines the circular area around a random point within which all nodes are considered for potential connections. The center of this circle is the random point.
- **Goal radius**: The circular area around the goal point.
- **Maximum number of iterations**: The limit at which the algorithm ceases exploration.

After defining the start point for the algorithm, the model begins generating random points within all navigable areas. The next step is to find all points within the search radius of each random point. The model then evaluates these nearby points to determine which connection results in the lowest cost. The cost is defined as the number of connections between each point and the start point. The model selects the nearby point with the lowest cost.
Subsequently, a collision check function is utilized to ensure that a straight line between the selected point and the random point does not intersect any obstacles. If a collision is detected, the model skips the current point and evaluates the next nearest point with a lower cost. This step differentiates RRT* from RRT. In RRT, the nearest point to the random point is selected without considering the cost.
Next, the algorithm draws a straight vector between the chosen point and the random point, with a length equal to the step size and in the direction from the nearest point to the random point. A new point is created at the end of this vector, and the random point is discarded. The new point will be added to the nodes vector and the its nearest point will be added to the parents vector.This process continues until one of the newly generated points is located within the goal radius and the number of iterations is less than the maximum number of iterations. If the number of iterations exceeds the maximum without any new point reaching the goal circle, the exploration stops and the goal is not reached.

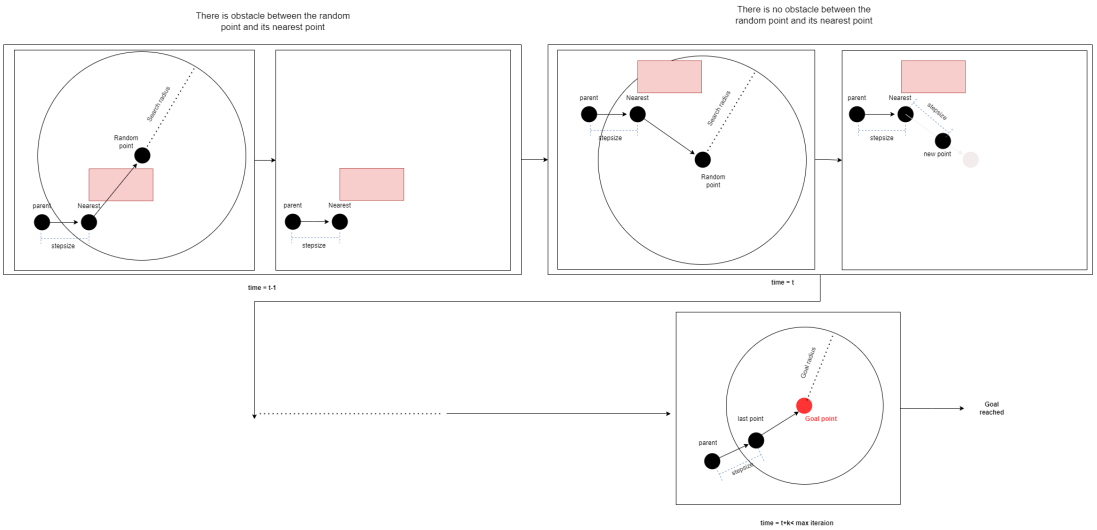


Figure 2.2: RRT* algorithm in different steps if the goal is reached

When the goal is reached, the final point will be added to the path array, followed

by its parent, then its parent's parent, and so on. This process continues until the start point is reached. Finally, the path vector will be reversed to start from the initial point and end at the final point.

3
Methods

In our pursuit to improve navigation safety through off-road terrain and object detection, we will concentrate on resolving a series of critical sub-problems. It's essential to emphasize that our reliance on simulation software such as IPG for model evaluation precludes us from utilizing real-world data. The sub-problems we aim to address in order to attain our objective are as follows:

- Scenario creation and Sensor configuration
- Data Acquisition and Preprocessing
- Implementation of Semantic Segmentation Fusion
- Development of Mathematical Algorithms
- Evaluation and Performance Assessment of the Implementation
- Designing an Analytical Model for Enhance Decision-Making

3.1 Scenario creation

Scenario creation is crucial to achieving our objectives, as it forms the basis for configuring sensors and streaming essential data during the research process. The goal is to design a highly accurate and specific dataset for off-road environments, with a strong emphasis on precision in scenario design and sensor setup. This ensures that the collected data aligns closely with the thesis objectives, providing a solid foundation for the development and testing phases.

3.1.1 Scenario Design in IPG

In the IPG Carmaker simulation software [24], we meticulously construct off-road terrains. These terrains are characterized by diverse and irregular surface conditions, simulating a range of challenges encountered in off-road environments. Within the simulation environment, we engineer scenarios featuring varied surface profiles and obstacles such as trees, bushes, rocks, stones, and logs scattered across the terrain. Our approach involves creating off-road scenarios tailored to three distinct road

conditions [25]: gravel, mud, and snow. Each scenario is carefully crafted to achieve our thesis objective, ensuring that the simulated environments accurately represent the challenges posed by different off-road terrains.
Given that the obstacles described earlier are not considered solid objects within the IPG environment, the vehicle can traverse them without encountering collisions. However, to facilitate collision testing, a solid object representation is required. Therefore, cones have been introduced as solid objects in our scenario.

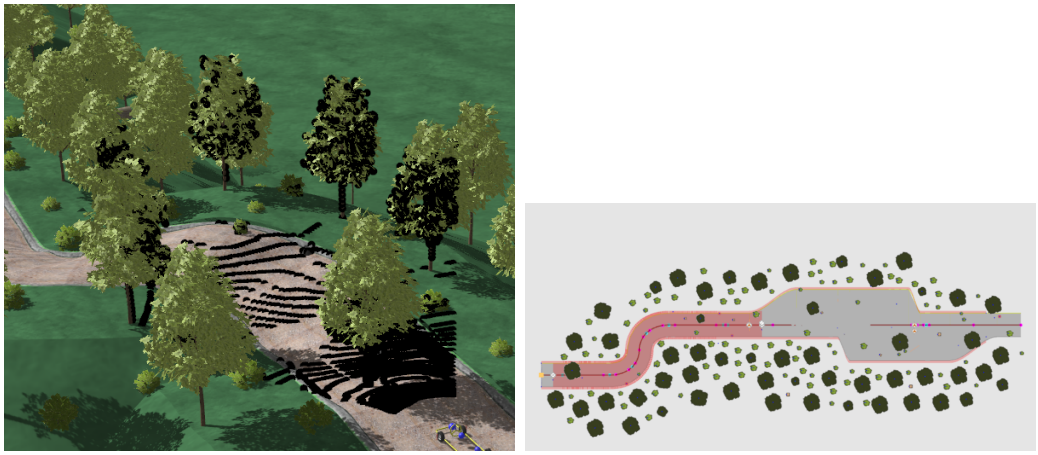


Figure 3.1: Created scenario in IPG CarMaker

3.1.2 Sensor mounting and Configurations

Our research focuses primarily on harnessing image and point cloud data, necessitating the deployment of camera and LiDAR sensors on our test vehicle. The placement of these sensors is of paramount importance as it directly influences the coverage of the field of view (FOV) within the processing regions. It's crucial to highlight that our chosen sensors are Raw Signal Interfaces (RSIs), specifically tailored for component development and testing. RSIs prioritize a meticulous examination of physical effects through signal propagation in the environment. Unlike conventional sensor types, RSIs do not incorporate processing or tracking components, granting users the flexibility and responsibility to model all electronic and software elements. This unique attribute empowers users to delve into the intricate details of real components under development or testing while safeguarding proprietary knowledge.

The primary objective of physical sensor models is to furnish input data for the sensor's perception algorithms. These inputs typically comprise image data for camera simulation or LiDAR point clouds. Generating these raw signals necessitates meticulous consideration of detailed physical effects, as well as the geometry and material properties of objects. Physical sensor models [26] represent the pinnacle of realism, albeit being the most computationally demanding sensor models. In our open integration and test platform, CarMaker, physical sensor models are referred to as Raw Signal Interfaces (RSIs) 3.2.

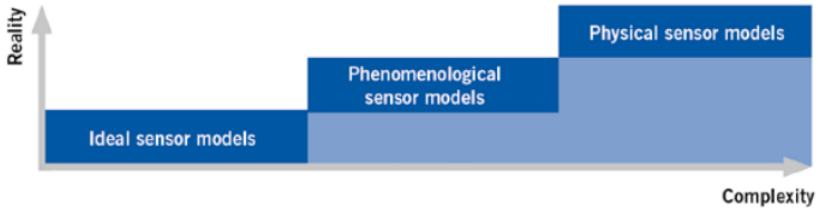


Figure 3.2: Physical sensor models are referred to as Raw Signal Interfaces (RSIs)

3.1.2.1 Camera Sensor

In our simulation software, the utilization of camera sensors is integral for capturing the environmental dynamics, providing essential RGB data outputs. The process involves meticulous configuration, which is structured into two distinct phases: Sensor Configuration and Sensor Assembly.

In the initial phase, Sensor Configuration 3.3, critical parameters such as field of view (FOV), resolution, lens types, and Coloration mode are fine-tuned to align with the specific requirements of the simulation scenario. This step ensures the precise capture of visual data essential for accurate analysis and simulation outcomes.

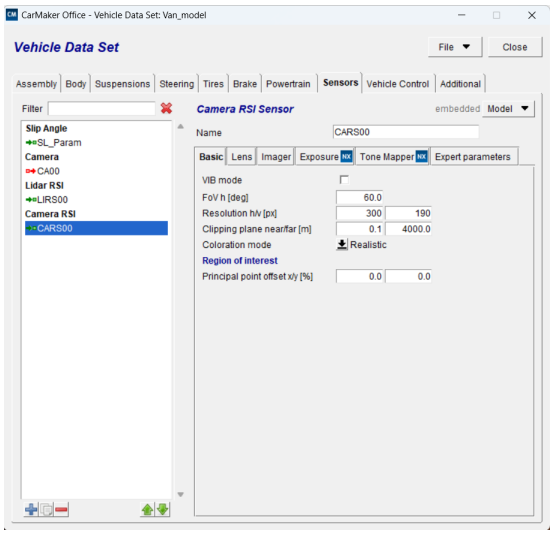


Figure 3.3: Camera Parameterized Window

Following Sensor Configuration, the next phase is Sensor Assembly 3.4, where parameters related to name of the sensor, Configured Parameterization, positioning in meters(m) and orientation in degree(deg) in the x, y, and z dimensions are meticulously configured. Adhering to Zeekr's standard criteria, sensors are strategically positioned to optimize data collection efficiency and ensure compliance with project specifications. This step is crucial for achieving reliable simulation results by accurately replicating real-world scenarios.

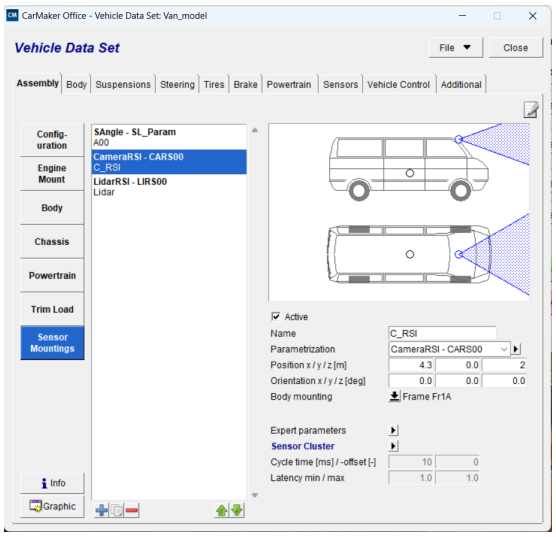


Figure 3.4: Camera Sensor Mounting Window

Moreover, an advanced feature known as the sensor cluster 3.5 is available, particularly for RSI sensors. This feature enables users to configure cycle time, cycle offset, and latency values, enhancing the functionality and performance of the sensors within the simulation environment.

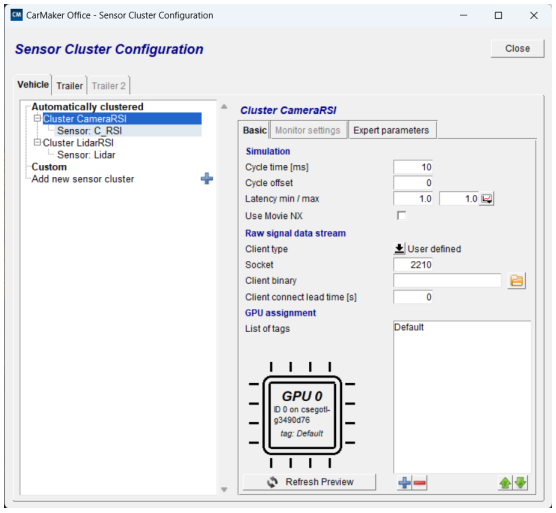


Figure 3.5: Camera Sensor Cluster Window

Through the systematic configuration of sensors and adherence to established standards, our simulation software empowers users to generate highly accurate and reliable simulations, facilitating insightful analysis and decision-making processes.

3.1.2.2 Lidar Sensor

In our simulation framework, the integration of LiDAR sensors plays a pivotal role in capturing the intricate 3D point cloud representation of the environment. This process involves the meticulous collection of crucial data elements, including BeamID,

LOF (Length of Flight), and intensity. To ensure alignment with our scenario specifications, we undertake a comprehensive sensor configuration approach.

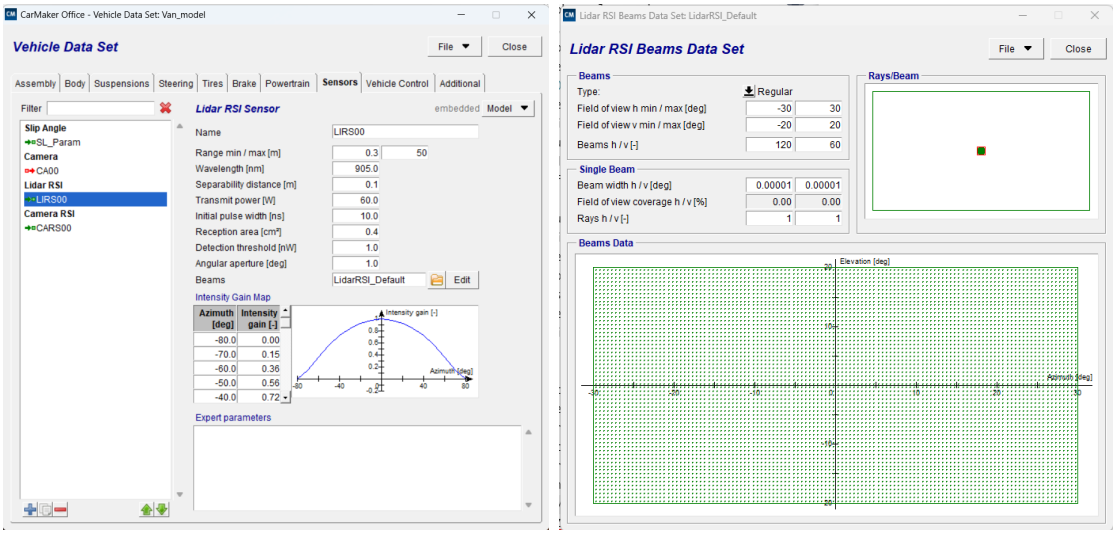


Figure 3.6: Lidar Sensor Parameterized Window

During the sensor configuration phase 3.6, we meticulously fine-tune various parameters to optimize sensor performance. These parameters encompass critical aspects such as range, the number of cloud points, and the field of view (FOV) in both vertical and horizontal dimensions. Additionally, settings related to Beam configurations (FOVH, FOVV, Beams h/V), as well as the intensity gain map (default Value), are carefully adjusted to achieve the desired simulation outcomes.

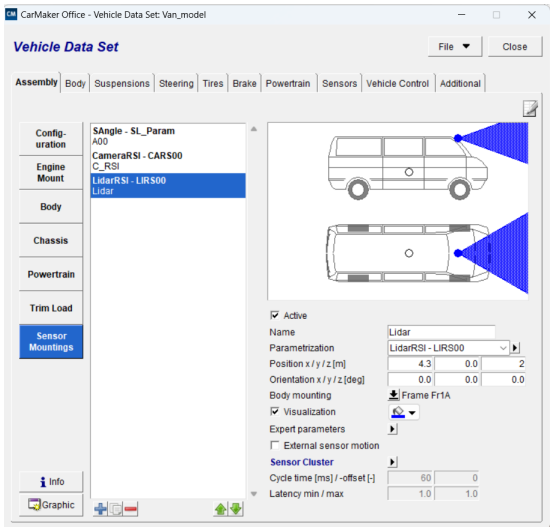


Figure 3.7: Lidar Sensor Mounting Window

Furthermore, in the next phase is Sensor Assembly where parameters related to name of the sensor, Configured Parameterization, positioning in meters(m) and orientation in degree(deg) in the x, y, and z dimensions are meticulously configured.

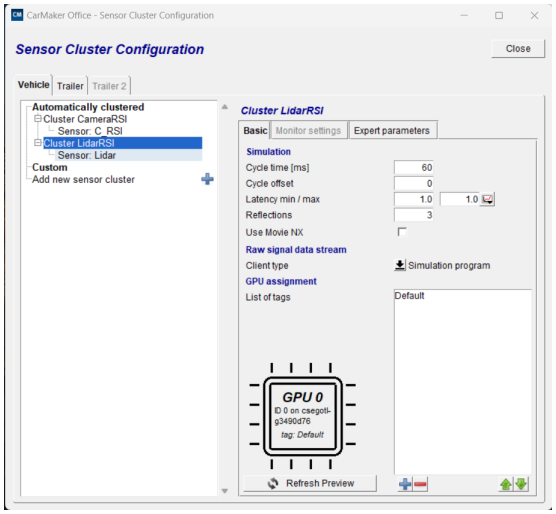


Figure 3.8: Lidar Sensor Cluster Window

Moreover, for enhanced functionality and performance, our simulation framework offers the Sensor Cluster 3.8 feature tailored specifically for LiDAR sensors. Within this feature, users can configure parameters such as cycle time, cycle offset, latency (both minimum and maximum), and reflections. These settings empower users to optimize sensor behavior, enabling comprehensive data capture and analysis within the simulation environment.

By meticulously configuring LiDAR sensors and adhering to established standards, our simulation framework facilitates precise and reliable 3D environment capture, empowering users with invaluable insights for decision-making and analysis.

3.2 Data Acquisition

In our project focused on terrain classification and object detection, data acquisition stands as a pivotal consideration, especially within the simulation environment. We rely on two primary sensors, namely cameras and Lidar, to gather essential data for training and testing our model. These sensors play a crucial role in capturing diverse environmental features and nuances, enabling our model to accurately classify and detect various terrains and objects.

3.2.1 Acquisition of camera data

To facilitate our 2D model's development, we employ a camera to capture environment images, which are then streamed to local storage for model implementation, training, validation, and testing. This streaming process utilizes the TCP/IP protocol 3.9. To execute this streaming process seamlessly, we have meticulously modified the C-language script "RSDS-client-camera-standalone.c" to align with our specific requirements. This tailored script facilitates the smooth transfer of image data from the simulation environment's memory to designated local drive storage, ensuring the preservation and accessibility of critical simulation data.

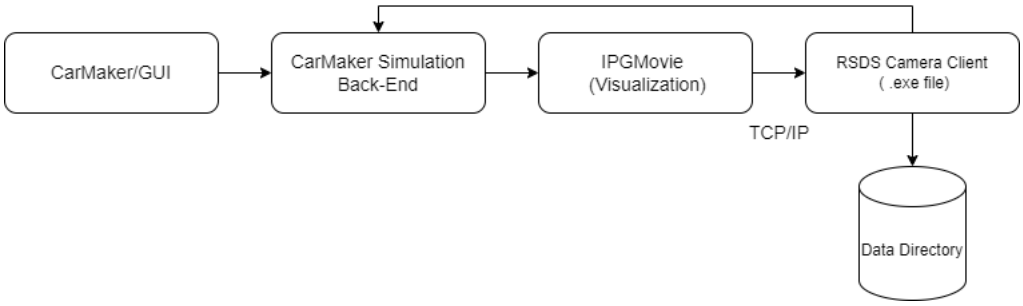


Figure 3.9: Process Flow for Image Data Streaming

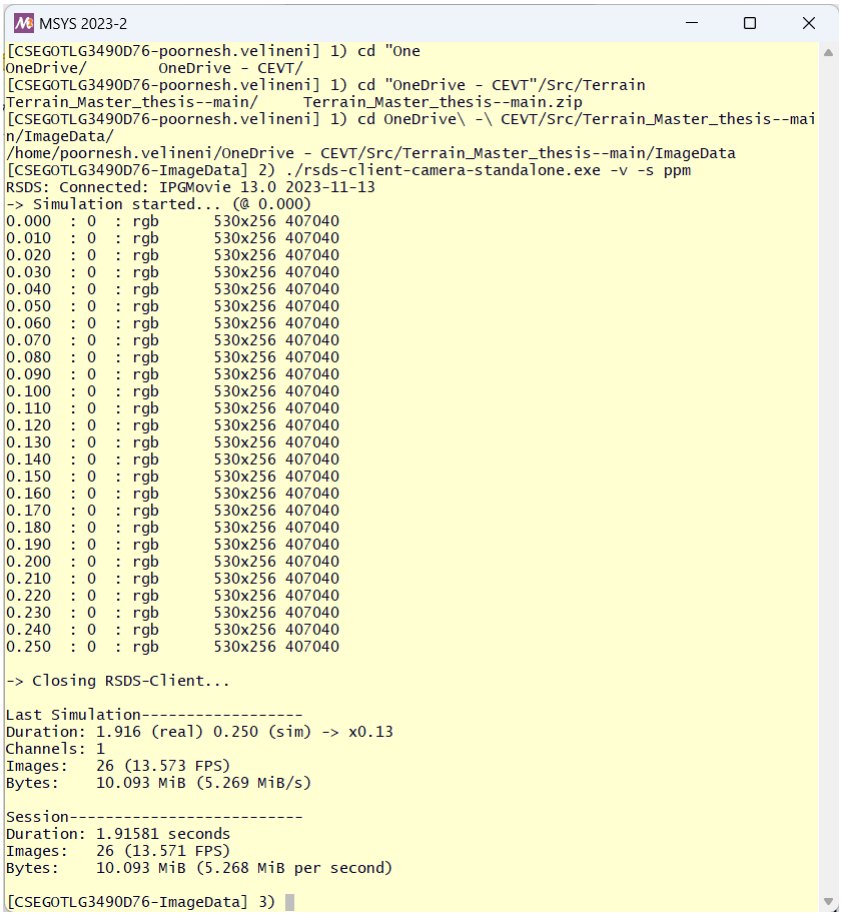


Figure 3.10: Streaming Output

To Initiate the image data streaming necessitates meticulous configuration within the sensor cluster configuration window (3.5). Here, meticulous attention is paid to configuring the communication port, a pivotal step in ensuring the successful establishment of the streaming process. With the TCP/IP protocol chosen for data transfer, setting the Socket (channel port number) as 2210 is imperative for seamless communication. Additionally, since our communication relies on the custom C-language script, the client type must be designated as user-defined, deviating from the default configuration intended for simulation programs.

Upon completing the configuration process, the execution of the "RSDS-client-camera-standalone.exe" executable file in the command prompt marks the commencement of the streaming process. As the RSDS TCP/IP connection is established and the simulation environment springs into action, a reassuring confirmation message (3.10) signals the readiness to receive data from the simulation. With the simulation in motion, the streaming process unfolds, diligently storing the captured data in the designated directory.

This pivotal action must be repeated for each terrain condition, ensuring comprehensive data capture, and whenever simulation data is required to be streamed for analysis or further utilization. Through meticulous attention to detail and seamless execution, our methodology ensures the efficient and reliable streaming of simulation data, laying the groundwork for robust modeling and analysis endeavors.

3.2.2 Acquisition of lidar data

In our endeavor to develop a 3D Semantic Segmentation model, streaming Lidar data plays another pivotal role. Upon the completion of a project within the IPG, an executable Simulink file comprising various blocks will be produced. Furthermore, the option to access an IPG Graphical User Interface (GUI) via the Simulink file is also facilitated. To accomplish this, we've developed a Simulink script to extract data from the simulation environment and stream it to the Matlab workspace. Subsequently, this data is saved as a struct file, ready for further preprocessing steps.The Lidar sensor within the simulation environment provides raw data, necessitating a conversion process to get the Cartesian coordination of all of the point clouds. The output quantities from the Lidar sensor, as depicted in the accompanying Figure 3.11, require careful analysis and processing to ensure accuracy and reliability in subsequent stages of model development and evaluation.

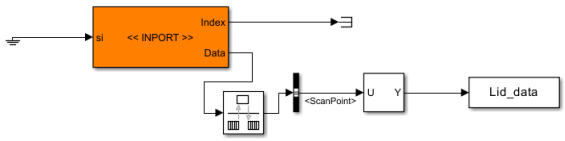


Figure 3.11: Lidar block added to the Simulink code for data acquisition

3.3 Preprocessing the dataset

After obtaining the dataset, the immediate subsequent step is data preprocessing, a crucial procedure to guarantee the accuracy and reliability of the data for subsequent implementation and evaluation phases. Given that we are handling data from two sensors, precision becomes particularly vital to mitigate any potential data loss or scarcity.

3.3.1 Data alignments

Data alignment involves structuring the data into a standardized format suitable for tasks like data augmentation and splitting for model training, validation, and testing. In the context of sensor data fusion, it's crucial to ensure that both image and point cloud data are timestamped synchronously.For instance, we've devised an algorithm tailored for Lidar sensor data. This algorithm converts the raw data from the sensor's Length of Flight (LOF) format into a more manageable x, y, and z format. This conversion process entails utilizing azimuth and elevation angles to accurately represent the spatial coordinates of the points in the point cloud. By aligning the data and transforming it into a consistent format, we ensure that our models receive synchronized input, facilitating effective fusion of image and point cloud data for enhanced analysis and decision-making.

3.3.2 Data Annotation

Semantic segmentation involves analyzing and categorizing each pixel or point in an image or point cloud, distinguishing between different classes or categories. Before implementing the model, data annotation is essential to provide ground truth labels for training and evaluating the model's performance.In our case, where we're developing both 2D and 3D semantic segmentation models, annotation is required for both image and point cloud data. This entails labeling each pixel in images and each point in point clouds with the corresponding semantic class or category. Annotation ensures that the model learns to accurately classify and segment objects or regions of interest within the data. It provides the necessary supervision for training the model to understand the semantic context of the scene, enabling it to make informed decisions during inference.

3.3.2.1 Image data Annotation

In dealing with simulated data using the IPG Carmaker simulation tool, there's a convenient feature for obtaining annotated images. To utilize this feature, you'll need to configure several parameters and create an object class configuration file. This file defines all the objects present in your simulations scenario along with their corresponding masked colors.Before initiating the simulations, it's essential to bind the "ObjectClass.cfg" file within the sensors configuration box. Once the configuration is complete, you can stream the data and store it in the respective directory.

3.3.2.2 Cloud-Point data Annotation

Annotating point cloud data isn't as straightforward as annotating image data. To annotate point clouds, we first need to project them onto an encoded image. The detailed implementation is explained in 3D semantic segmentation section 3.2

3.4 Segmentation Model

Our main task now revolves around implementing a semantic segmentation model. To accomplish this objective, we propose a solution that enhances sensor fusion techniques. Initially, we'll develop a 2D model for image prediction, while simultaneously creating a 3D model for cloud-points prediction. Subsequently, we fuse the predicted outputs from both the 2D and 3D models to improve the accuracy of our predictions. This fusion process aims to leverage the strengths of each model, ultimately leading to more precise segmentation results.

3.4.1 2D Image Segmentation

Based on surveys and existing research [27], we have chosen to employ the U-Net and Pyramid Scene Parsing Network (PSPNet) architectures for training 2D semantic segmentation models. This decision is driven by the specific challenges posed by outdoor environments, particularly uneven terrains. The U-Net architecture is well-suited for capturing fine details and nuances in complex scenes, making it an ideal choice for detecting subtle variations in outdoor terrain. On the other hand, PSPNet's ability to aggregate global contextual information from different regions.

3.4.1.1 Model Creation and Training

In the initial phase of our project, we embarked on constructing a 2D semantic segmentation model utilizing the U-Net architecture, bolstered by a ResNet backbone. This foundational step involved meticulous fine-tuning of hyperparameters and selecting an appropriate loss function to optimize precision. Concurrently, we replicated this architecture using the VGG16 backbone for comparison purposes. Subsequently, in the second stage, we transitioned to employing the PSPNet architecture, also leveraging a ResNet backbone, and tailored a specific loss function to enhance model performance. Simultaneously, we repeated the training process using the PSPNet architecture with the VGG16 backbone.
Throughout these stages, we meticulously monitored and analyzed various metrics including computation time, memory usage, and precision. This rigorous comparative analysis allowed us to discern the strengths and weaknesses of each model variant.Upon thorough evaluation, we concluded that the U-Net architecture with the ResNet backbone exhibited superior performance compared to the other model configurations. This determination was based not only on precision metrics but also on considerations such as computation time and memory efficiency.

3.4.1.2 Dataset pre-processing

Prior to commencing the model training process, it is imperative to preprocess the dataset. The input data for the U-net model is required to be an RGB image with dimensions of $256 \times 256 \times 3$. Conversely, the model's output data shall consist of an encoded grayscale image with dimensions of 256×256. In the case of PSPNet, the input dimensions are specified as $240 \times 240 \times 3$ for RGB images, with corresponding output dimensions of 240×240 for encoded grayscale images. Consequently, resizing

of real images is necessitated to align with the model's input requirements. Moreover, the masked images are to be resized, converted to grayscale, and encoded prior to initiating the model training phase. An essential consideration when resizing images is to utilize the "INTER NEAREST" command in Python and the "nearest" method in MATLAB. Failure to do so may result in the generation of new pixels with altered colors during the resizing process. These new colors could potentially introduce confusion during encoding and decoding, consequently inflating the number of classes.
Given the camera's sampling rate of 10ms and the vehicle's relatively low velocity, many images exhibit similarity with minor variations between adjacent frames. To mitigate the occurrence of duplicated images and to enhance the dataset quality, dataset augmentation techniques have been employed. This augmentation strategy includes horizontal and vertical flips, as well as rotation by increments of 45 degrees.The preferred loss function for multi-class classification, notably utilized in the training of both PSPnet and U-net models, is categorical cross-entropy. Recognizing the inherent imbalance within the dataset, specific weights have been assigned to individual classes to address this disparity during training. Assigning weights allows the model to appropriately adjust for this class imbalance during training. By assigning higher weights to underrepresented classes, the model can pay more attention to these classes and prevent them from being overshadowed by dominant classes. The table illustrating the existing classes and their respective encoded colors and weights is provided in Table 3.1.

Class	**RGB color**	**Gray-scale color**	**Encoded color**	**Training Weights**
Unlabled	0,0,0	0	0	1.63
Log	102,0,0	30	1	23.53
Stone	255,0,0	76	2	22.16
Sideroad	69,89,99	84	3	5.65
Sky	33,148,240	124	4	0.26
Terrain	74,173,79	133	5	1.53
Small cone	25,219,56	142	6	258.94
Gravel Road	255,255,0	226	7	0.63
Vegetation	138,194,74	164	8	0.37
Cone	252,150,212	188	9	51.58
Snowy road	0,255,255	152	10	0.74
Muddy road	255,128,0	179	11	0.86

Table 3.1: Object classes and their colors and training weights

3.4.2 3D Cloud-Point Segmentation

Based on research [28], we have opted to utilize the SqueezeSeg V2 architecture for training our 3D semantic segmentation models, primarily due to the unique demands imposed by outdoor environments, particularly in the context of uneven terrains. SqueezeSeg V2's architecture is well-primed to address the intricacies of outdoor

scenes, capturing fine details and nuances. While acknowledging the existence of other architectures such as SalsaNet, PointNet, PointNet++, and RangeNet++, our decision to prioritize SqueezeSeg V2 stems from its demonstrated efficacy in handling the challenges specific to outdoor environments. By leveraging this architecture, we aim to develop robust 3D semantic segmentation models capable of accurately delineating objects and terrain features amidst the complexities of outdoor terrains. The network structure of SqueezeSegV2 is shown in Figure 3.12.

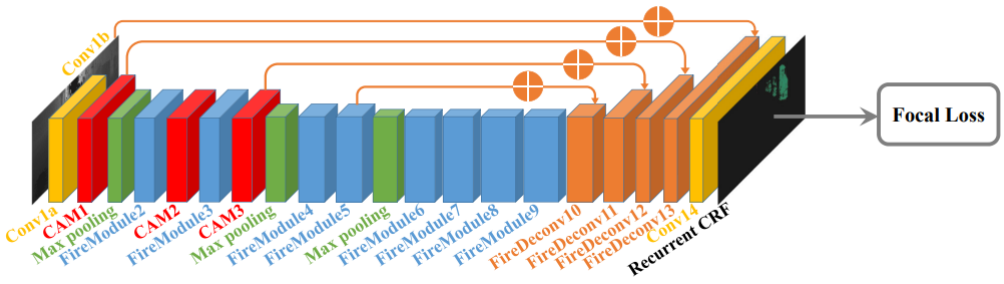


Figure 3.12: Squeezseg V2 network structure [19]

3.4.2.1 Model Creation and Training

SqueezeSegV2 is a Convolutional Neural Network (CNN) meticulously crafted to perform end-to-end semantic segmentation on organized LiDAR point clouds. It's tailored to handle the intricacies of LiDAR data, offering a comprehensive solution for semantic segmentation tasks within this domain. Notably, the training protocol outlined in this framework relies on 2-D spherical projected images as crucial inputs for the deep learning network. It's noteworthy that the model architecture is structured to accommodate inputs with five channels. Furthermore, leveraging pre-trained SqueezeSeg architectures plays a pivotal role in training and fine-tuning the model, ensuring efficient learning and robust performance in various real-world scenarios.

The initial stage preceding model training involved preparing and adjusting the point cloud data for input into the model. As previously noted, the input for the SqueezeSeg model necessitates an image comprising five channels: x, y, z, intensity, and range. Additionally, the dimensions of the input image must be divisible by 8. Therefore, in our scenario, the input dimensions are set at 64×128. The total count of point clouds should correspond to the image size. Given that the maximum number of point clouds in the IPG simulation was 10,000, we opted for an input size of 64×128, resulting in 8,192 point clouds. An additional critical consideration is ensuring that both the camera and Lidar share the same field of view. This alignment is essential as we intend to utilize the annotated camera images as labels for the corresponding point clouds.
The streamed point cloud data from IPG arrives in raw form, encompassing attributes such as flight length and intensity. Following the sensor's installation onto

the vehicle and the establishment of its configuration, a text file named "LidarRSI Default" is generated. This file contains the beamIDs alongside their respective azimuth and elevation angles. Subsequently, upon streaming the raw lidar data and acquiring their associated beamIDs and flight lengths, the task at hand involves deriving the Cartesian coordinates for each point cloud(Azimuth angle=θ,Elevation angle=θ, and r = length of flight/2).

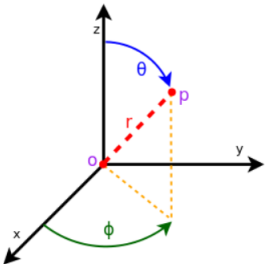


Figure 3.13: Spherical to Cartesian coordination conversion

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\sin(\phi)\cos(\theta) \\ r\sin(\phi)\sin(\theta) \\ r\cos(\phi) \end{bmatrix} \tag{3.1}$$

The annotated labels of the input data comprise the annotated images captured by the camera. These annotated camera images should match the input image size, set at 64×128. Furthermore, it's necessary to convert these annotated images to gray-scale and then encode them prior to training. Existing classes with their corresponding encoded colors have been shown in Table 3.2.

Class	**RGB color**	**Gray-scale color**	**Encoded color**
Terrain	74,173,79	133	1
Road	255,255,0	226	2
Vegetation	138,194,74	164	3
Sideroad	69,89,99	84	4
Log	102,0,0	30	5
Stone	255,0,0	76	6
Cone	252,150,212	188	7
Small cone	25,219,56	142	8
Unlabled	0,0,0	0	9

Table 3.2: Object classes and their colors

For simplifying the training process, both the camera and Lidar sensor have been positioned in identical locations with matching orientations and fields of view. Consequently, when projecting the point cloud onto the image, each point cloud corresponds to a pixel within the image.

Upon obtaining the Cartesian coordinates for all point clouds, the array size of the Cartesian positions will be 8192×3. Subsequently, the projection of these point clouds onto the camera image plane is necessary to determine their positions on the annotated camera image. This projection process requires the utilization of the camera calibration matrix. The process of camera calibration involves converting distances measured in meters to pixel units and adjusting the image plane to align with the center of the image. This calibration ensures accurate spatial measurements and proper alignment of the captured images. (principal-axis$_x$=u_0, principal-axis$_y$=v_0, image width = Img_w, image height = Img_h, field of view horizontal = Fov_h, field of view vertical = Fov_v)

$$u_0 = Img_w/2,\ v_0 = Img_h/2,\ f_x = \frac{Img_w/2}{\tan(Fov_h/2)},\ f_y = \frac{Img_h/2}{\tan(Fov_v/2)} \tag{3.2}$$

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Upon performing matrix multiplication between the Cartesian coordinate array and the camera calibration matrix, adjustments are necessary to account for the fact that the camera image plane is situated where z equals 1. Consequently, to obtain a 2D image, scaling of the x and y coordinates by $1/z$ is required.

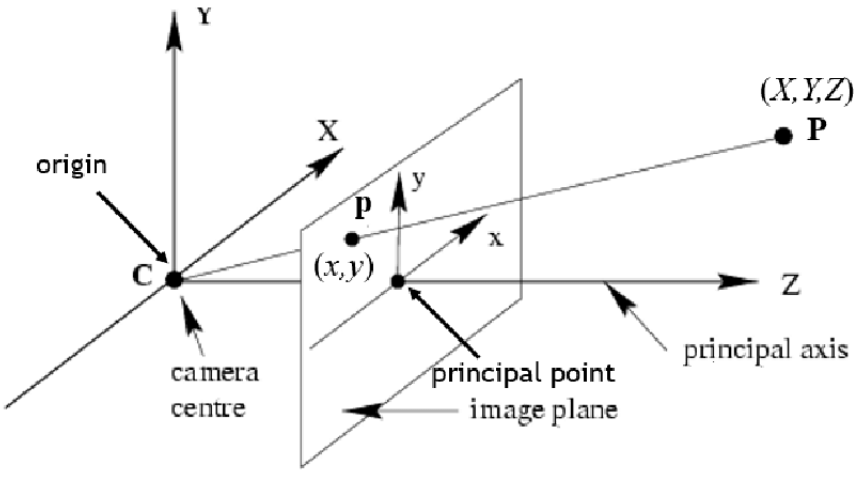


Figure 3.14: Pinhole camera model

$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \Rightarrow \mathbf{P}_{calibrated} = \mathbf{K}P = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \Rightarrow \mathbf{p} = \begin{bmatrix} x'/z' \\ y'/z' \\ z'/z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.4}$$

Following projection, the array size of positions will transition from 8192×3 to $64 \times 128 \times 3$. Similarly, for intensity and range data, the array size will transform from 8192×1 to $64 \times 128 \times 1$. Upon concatenating these arrays, a combined array of dimensions $64 \times 128 \times 5$ will be obtained, corresponding to $img_w \times img_h \times (x, y, z, Intensity, range)$.

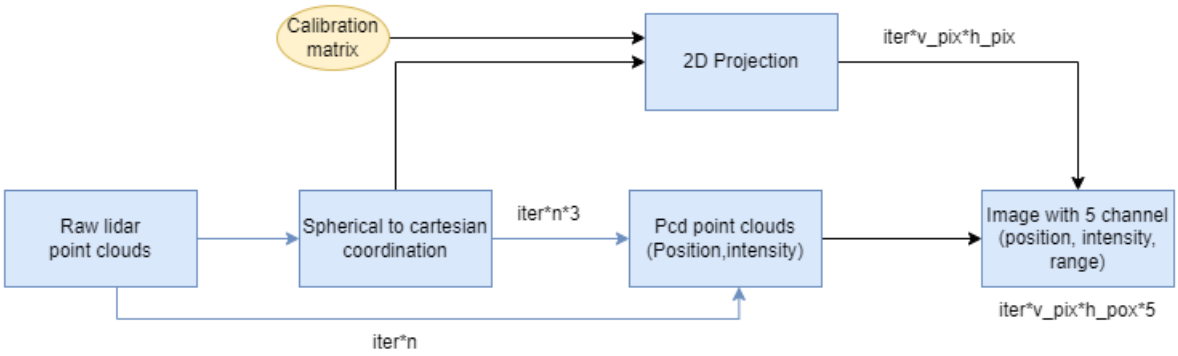


Figure 3.15: Lidar point clouds pre-processing procedure

Following data preprocessing, the model is ready for training. The dataset has been divided into two subsets: 70% for training and 30% for validation and testing. As previously stated, the model's input comprises five-channel point cloud data. The model generates an encoded 64×128 image as output. The final phase of the prediction involves decoding this image based on the table provided in Section 3.2 to produce an RGB representation of the scene.

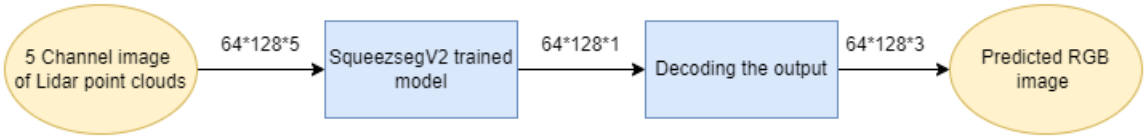


Figure 3.16: 3D semantic segmentation label prediction of the Lidar point clouds

An important consideration is that the camera and Lidar may have distinct positions and orientations. Thus, prior to projecting the Lidar point cloud onto the camera image plane, it's essential to recognize that the origin for the Lidar point clouds lies within the Lidar's coordinate frame. To obtain the Cartesian coordinates of the point clouds relative to the Camera coordinate frame, a transformation matrix between the camera and Lidar coordinate frames is required. By multiplying this transformation matrix with the extended Lidar Cartesian coordinates, the points will be transformed into the Camera coordinate frame, thereby establishing the Camera coordinate frame as the new origin.

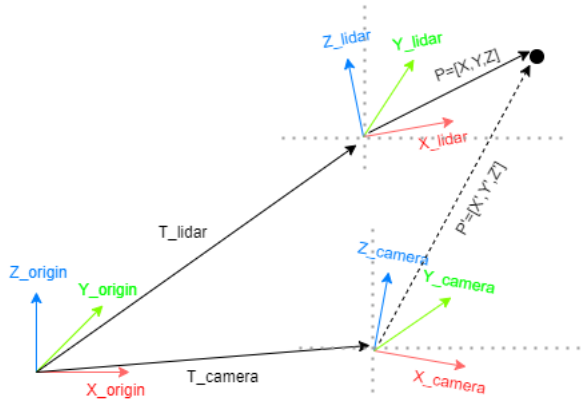


Figure 3.17: Relations between camera and lidar coordinate system

As illustrated in Figure 3.17, point P resides in the lidar coordinate frame. The orientation and position of each sensor relative to the origin can be obtained from sensor configurations in IPG, as depicted in Figures 3.4 and 3.7. To determine the coordinates of point P in the camera coordinate system (P'), the following tasks need to be performed(c:camera, l=lidar, o:origin, H_o^c: Homogenous transformation matrix of camera with respect to the origin, H_o^l: Homogenous transformation matrix of lidar with respect to the origin, H_c^l: Homogenous transformation matrix of lidar with respect to the camera, T_c: Translation vector of camera with respect to the origin, T_l: Translation vector of lidar with respect to the origin, R_c: Rotation matrix of camera with respect to the origin, R_l: Rotation matrix of lidar with respect to the origin):

$$H_o^c = \begin{bmatrix} R_c & T_c \\ \mathbf{0} & 1 \end{bmatrix}, \quad H_o^l = \begin{bmatrix} R_l & T_l \\ \mathbf{0} & 1 \end{bmatrix} \Rightarrow H_c^l = H_o^l H_c^o = H_o^l \left(H_o^c\right)^{-1} \tag{3.5}$$

$$\begin{bmatrix} P' \\ 1 \end{bmatrix} = H_c^l \begin{bmatrix} P \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_c^l & T_c^l \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.6}$$

3.5 Model Evaluation

The evaluation of the model involves separate assessments for both 2D and 3D semantic segmentation to comprehensively analyze the performance of each model in their respective domains. This approach allows for a detailed examination of how well each model performs in segmenting and classifying objects within their specific contexts.

3.5.1 Model Performance

The performance evaluation of both the 2D and 3D models is conducted using key metrics such as Mean Intersection over Union (MeanIOU) value for each class, precision, and loss values. These metrics provide insights into the accuracy, consistency,

and effectiveness of the models in segmenting objects across different classes within their respective dimensions. By leveraging these metrics and visualizations plots, we gain valuable insights into the overall performance of the trained models and can make informed decisions regarding further optimization or deployment.

3.5.1.1 2D model performance:

As mentioned in the Methods section, two distinct semantic segmentation models with different backbones (ResNet and VGG16) have been trained. The dataset was shuffled and divided, allocating 70% for training and 30% for validation before training each model.
The chosen loss function is the 'Categorical Cross Entropy' loss function, widely used for multi-class classification models.
Recognizing the dataset's imbalance, specific weights have been assigned to each class to ensure the model treats all classes equally (See Table 3.1).

U-net model with Resnet Backbone

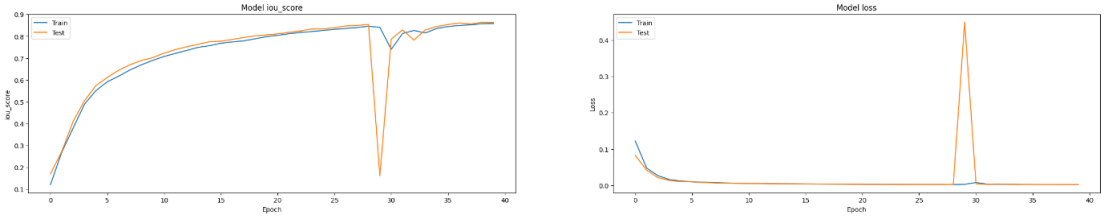


Figure 3.18: The loss and accuracy diagrams for both training and validation dataset after training 40 epochs of a U-net model with Resnet as backbone.

```
Epoch 29/40
188/188 [==============================] - 545s 3s/step - loss: 0.0024 - iou_score: 0.8460 - val_loss: 0.0026 - val_iou_score: 0.8538
Epoch 30/40
188/188 [==============================] - 545s 3s/step - loss: 0.0029 - iou_score: 0.8404 - val_loss: 0.4491 - val_iou_score: 0.1616
Epoch 31/40
188/188 [==============================] - 544s 3s/step - loss: 0.0079 - iou_score: 0.7401 - val_loss: 0.0041 - val_iou_score: 0.7870
Epoch 32/40
188/188 [==============================] - 545s 3s/step - loss: 0.0034 - iou_score: 0.8128 - val_loss: 0.0030 - val_iou_score: 0.8294
Epoch 33/40
188/188 [==============================] - 544s 3s/step - loss: 0.0031 - iou_score: 0.8264 - val_loss: 0.0039 - val_iou_score: 0.7823
Epoch 34/40
188/188 [==============================] - 544s 3s/step - loss: 0.0033 - iou_score: 0.8153 - val_loss: 0.0030 - val_iou_score: 0.8288
Epoch 35/40
188/188 [==============================] - 544s 3s/step - loss: 0.0028 - iou_score: 0.8354 - val_loss: 0.0026 - val_iou_score: 0.8444
Epoch 36/40
188/188 [==============================] - 545s 3s/step - loss: 0.0026 - iou_score: 0.8446 - val_loss: 0.0025 - val_iou_score: 0.8539
Epoch 37/40
188/188 [==============================] - 545s 3s/step - loss: 0.0024 - iou_score: 0.8495 - val_loss: 0.0025 - val_iou_score: 0.8609
Epoch 38/40
188/188 [==============================] - 545s 3s/step - loss: 0.0024 - iou_score: 0.8522 - val_loss: 0.0025 - val_iou_score: 0.8560
Epoch 39/40
188/188 [==============================] - 545s 3s/step - loss: 0.0023 - iou_score: 0.8573 - val_loss: 0.0023 - val_iou_score: 0.8642
Epoch 40/40
188/188 [==============================] - 546s 3s/step - loss: 0.0022 - iou_score: 0.8584 - val_loss: 0.0023 - val_iou_score: 0.8624
```

Figure 3.19: The loss and accuracy results for both training and validation dataset after training 40 epochs of a U-net model with Resnet as backbone.

PSP-net model with Resnet Backbone

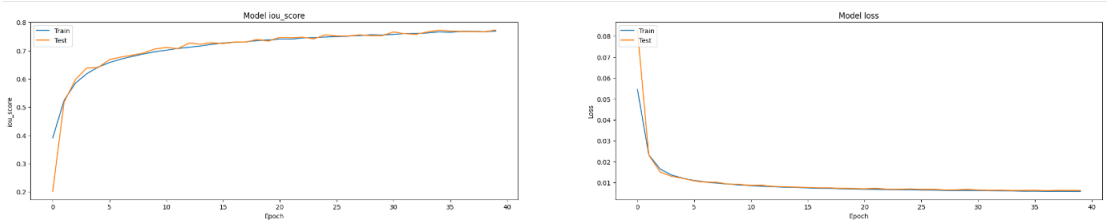


Figure 3.20: The loss and accuracy diagrams for both training and validation dataset after training 40 epochs of a PSP-net model with Resnet as backbone.

```
Epoch 29/40
188/188 [==============================] - 201s 1s/step - loss: 0.0063 - iou_score: 0.7557 - val_loss: 0.0066 - val_iou_score: 0.7527
Epoch 30/40
188/188 [==============================] - 200s 1s/step - loss: 0.0063 - iou_score: 0.7547 - val_loss: 0.0068 - val_iou_score: 0.7527
Epoch 31/40
188/188 [==============================] - 201s 1s/step - loss: 0.0062 - iou_score: 0.7572 - val_loss: 0.0065 - val_iou_score: 0.7659
Epoch 32/40
188/188 [==============================] - 200s 1s/step - loss: 0.0061 - iou_score: 0.7599 - val_loss: 0.0064 - val_iou_score: 0.7594
Epoch 33/40
188/188 [==============================] - 200s 1s/step - loss: 0.0061 - iou_score: 0.7603 - val_loss: 0.0064 - val_iou_score: 0.7573
Epoch 34/40
188/188 [==============================] - 200s 1s/step - loss: 0.0061 - iou_score: 0.7621 - val_loss: 0.0063 - val_iou_score: 0.7662
Epoch 35/40
188/188 [==============================] - 200s 1s/step - loss: 0.0059 - iou_score: 0.7661 - val_loss: 0.0063 - val_iou_score: 0.7717
Epoch 36/40
188/188 [==============================] - 200s 1s/step - loss: 0.0060 - iou_score: 0.7647 - val_loss: 0.0063 - val_iou_score: 0.7694
Epoch 37/40
188/188 [==============================] - 200s 1s/step - loss: 0.0059 - iou_score: 0.7676 - val_loss: 0.0062 - val_iou_score: 0.7668
Epoch 38/40
188/188 [==============================] - 200s 1s/step - loss: 0.0059 - iou_score: 0.7675 - val_loss: 0.0064 - val_iou_score: 0.7671
Epoch 39/40
188/188 [==============================] - 200s 1s/step - loss: 0.0059 - iou_score: 0.7663 - val_loss: 0.0063 - val_iou_score: 0.7663
Epoch 40/40
188/188 [==============================] - 201s 1s/step - loss: 0.0058 - iou_score: 0.7688 - val_loss: 0.0063 - val_iou_score: 0.7722
```

Figure 3.21: The loss and accuracy results for both training and validation dataset after training 40 epochs of a PSP-net model with Resnet as backbone.

U-net model with Vgg16 Backbone

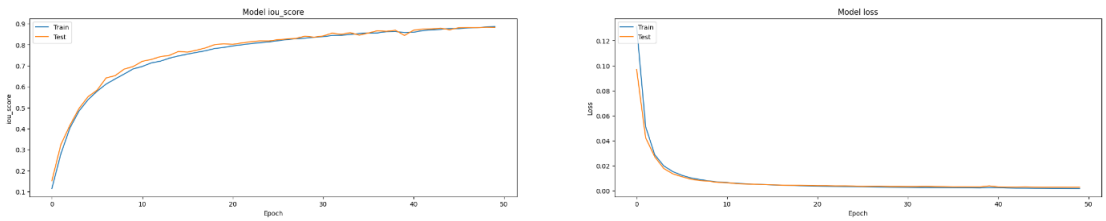


Figure 3.22: The loss and accuracy diagrams for both training and validation dataset after training 50 epochs of a U-net model with Vgg16 as backbone.

```
Epoch 39/50
188/188 [==============================] - 1020s 5s/step - loss: 0.0023 - iou_score: 0.8628 - val_loss: 0.0031 - val_iou_score: 0.8687
Epoch 40/50
188/188 [==============================] - 1020s 5s/step - loss: 0.0026 - iou_score: 0.8576 - val_loss: 0.0040 - val_iou_score: 0.8441
Epoch 41/50
188/188 [==============================] - 1021s 5s/step - loss: 0.0024 - iou_score: 0.8590 - val_loss: 0.0031 - val_iou_score: 0.8687
Epoch 42/50
188/188 [==============================] - 1021s 5s/step - loss: 0.0022 - iou_score: 0.8670 - val_loss: 0.0030 - val_iou_score: 0.8740
Epoch 43/50
188/188 [==============================] - 1021s 5s/step - loss: 0.0021 - iou_score: 0.8708 - val_loss: 0.0030 - val_iou_score: 0.8749
Epoch 44/50
188/188 [==============================] - 1019s 5s/step - loss: 0.0021 - iou_score: 0.8727 - val_loss: 0.0031 - val_iou_score: 0.8785
Epoch 45/50
188/188 [==============================] - 1021s 5s/step - loss: 0.0020 - iou_score: 0.8770 - val_loss: 0.0030 - val_iou_score: 0.8709
Epoch 46/50
188/188 [==============================] - 1024s 5s/step - loss: 0.0020 - iou_score: 0.8764 - val_loss: 0.0029 - val_iou_score: 0.8807
Epoch 47/50
188/188 [==============================] - 1021s 5s/step - loss: 0.0020 - iou_score: 0.8795 - val_loss: 0.0029 - val_iou_score: 0.8820
Epoch 48/50
188/188 [==============================] - 1023s 5s/step - loss: 0.0020 - iou_score: 0.8809 - val_loss: 0.0029 - val_iou_score: 0.8813
Epoch 49/50
188/188 [==============================] - 1020s 5s/step - loss: 0.0019 - iou_score: 0.8842 - val_loss: 0.0029 - val_iou_score: 0.8830
Epoch 50/50
188/188 [==============================] - 1020s 5s/step - loss: 0.0019 - iou_score: 0.8863 - val_loss: 0.0029 - val_iou_score: 0.8824
```

Figure 3.23: The loss and accuracy results for both training and validation dataset after training 50 epochs of a U-net model with Vgg16 as backbone.

PSP-net model with Vgg16 Backbone

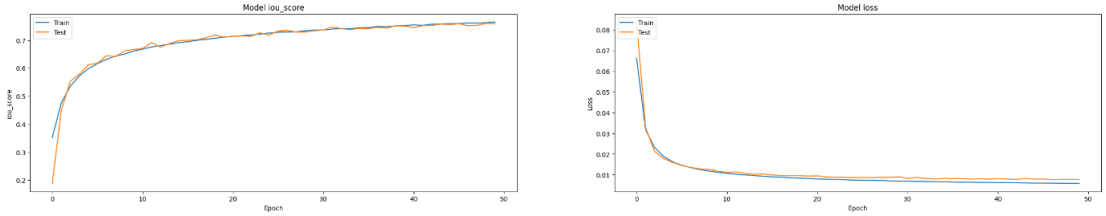


Figure 3.24: The loss and accuracy diagrams for both training and validation dataset after training 50 epochs of a PSP-net model with Vgg16 as backbone.

```
Epoch 39/50
188/188 [==============================] - 205s 1s/step - loss: 0.0063 - iou_score: 0.7511 - val_loss: 0.0080 - val_iou_score: 0.7515
Epoch 40/50
188/188 [==============================] - 205s 1s/step - loss: 0.0062 - iou_score: 0.7521 - val_loss: 0.0079 - val_iou_score: 0.7505
Epoch 41/50
188/188 [==============================] - 205s 1s/step - loss: 0.0062 - iou_score: 0.7549 - val_loss: 0.0081 - val_iou_score: 0.7447
Epoch 42/50
188/188 [==============================] - 204s 1s/step - loss: 0.0062 - iou_score: 0.7532 - val_loss: 0.0079 - val_iou_score: 0.7519
Epoch 43/50
188/188 [==============================] - 204s 1s/step - loss: 0.0061 - iou_score: 0.7572 - val_loss: 0.0077 - val_iou_score: 0.7520
Epoch 44/50
188/188 [==============================] - 203s 1s/step - loss: 0.0060 - iou_score: 0.7583 - val_loss: 0.0082 - val_iou_score: 0.7579
Epoch 45/50
188/188 [==============================] - 204s 1s/step - loss: 0.0060 - iou_score: 0.7592 - val_loss: 0.0079 - val_iou_score: 0.7556
Epoch 46/50
188/188 [==============================] - 204s 1s/step - loss: 0.0060 - iou_score: 0.7590 - val_loss: 0.0079 - val_iou_score: 0.7591
Epoch 47/50
188/188 [==============================] - 205s 1s/step - loss: 0.0059 - iou_score: 0.7616 - val_loss: 0.0076 - val_iou_score: 0.7515
Epoch 48/50
188/188 [==============================] - 205s 1s/step - loss: 0.0059 - iou_score: 0.7613 - val_loss: 0.0077 - val_iou_score: 0.7529
Epoch 49/50
188/188 [==============================] - 204s 1s/step - loss: 0.0058 - iou_score: 0.7626 - val_loss: 0.0078 - val_iou_score: 0.7601
Epoch 50/50
188/188 [==============================] - 205s 1s/step - loss: 0.0058 - iou_score: 0.7648 - val_loss: 0.0077 - val_iou_score: 0.7587
```

Figure 3.25: The loss and accuracy results for both training and validation dataset after training 50 epochs of a PSP-net model with Vgg16 as backbone.

After comparing the four models, it was observed that the U-net with VGG16 as the backbone achieved the highest training and validation accuracy among them. Consequently, this model is selected as the final 2D model. Additionally, by computing

the Intersection over Union (IoU) values for the models with VGG16 as the backbone, we can determine whether the U-net with VGG16 backbone exhibits higher IoU values for each class or if certain classes demonstrate higher IoU values in PSP-net with VGG16. If the latter is true, it implies that a fusion of these two models is warranted. Specifically, for certain classes, we would rely on the predictions from PSP-net, while for others, we would trust the U-net with VGG16 backbone.

Object class	Grayscale color	U-net IoU	PSP-net IoU
unlabled	0	0.99	0.98
log	30	0.82	0.68
stone	76	0.89	0.79
sideroad	84	0.89	0.82
sky	124	0.90	0.72
terrain	133	0.93	0.86
small cone	142	0.45	0.22
muddy road	151	0.97	0.96
vegetation	164	0.86	0.66
cone	188	0.81	0.75
gravel road	226	0.98	0.97
snowy road	240	0.98	0.98

Table 3.3: IoU of the each class

As Table 3.3 illustrates, the U-net model with Vgg16 backbone has high IoUs for all of the object classes and that's why no model fusion is needed in this stage.

3.5.1.2 3D model performance:

The pre-processing of the Lidar point clouds before training the SqueezeSegV2 model has already been mentioned in Methods section. After dataset preparation and processing, it is time to train the model. The dataset has been split to 75% for training and 25% for validation. The loss function which has been used is "Focal cross entropy" loss function. Focal loss focuses on the examples that the model gets wrong rather than the ones that it can confidently predict, ensuring that predictions on hard examples improve over time rather than becoming overly confident with easy ones[29]. After training the model for 50 epochs and batch size 32, the following results have been achieved:

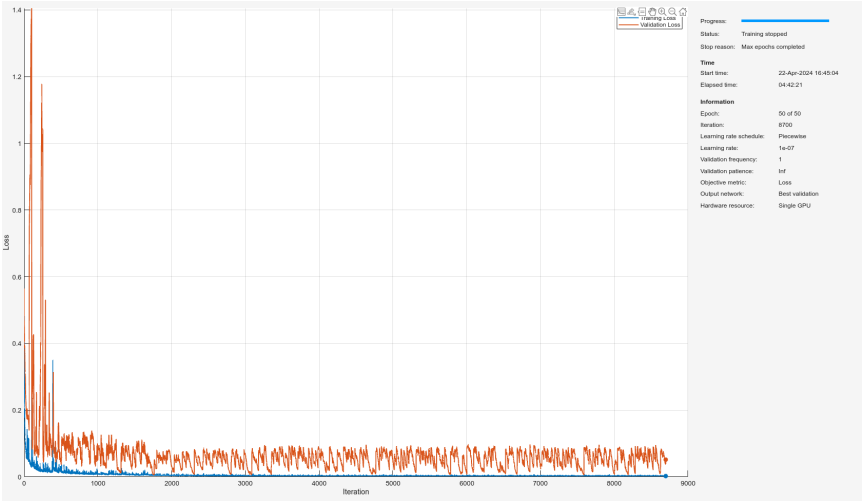


Figure 3.26: The loss diagram for both training and validation dataset after training 50 epochs

Object class	Grayscale color	Accuracy	IoU
unlabled	0	0.99	0.99
log	30	0.73	0.71
stone	76	0.84	0.78
sideroad	84	0.88	0.82
sky	124	NaN	NaN
terrain	133	0.96	0.89
small cone	142	0.09	0.09
road	151	0.98	0.96
vegetation	164	0.97	0.95
cone	188	0.56	0.42
Total	**-**	**0.98**	**0.97**

Table 3.4: Accuracy and IoU of each class and the total Accuracy and IoU after 50 epochs training.

According to Table 3.4, the model performs well and achieves a high accuracy. However, including the "sky" class in this model seems unnecessary, as there is no point cloud reflection from the sky. Additionally, while the surface condition (gravel, muddy, snowy) in the IPG simulator can be modified by selecting different materials, these changes do not alter the underlying surface properties, such as reflection, slope, irregularity, and friction. Instead, they merely replace the surface image while retaining its original properties. As a result, the 3D trained model is unable to differentiate between road surface conditions (snowy, muddy, gravel), as it receives the same point clouds for all three road surfaces. Therefore, for accurate road surface prediction, reliance on the 2D model is necessary.

3.5.2 Fusion of 2D and 3D models

As previously mentioned, due to some limitations in the IPG Carmaker simulator, two different semantic segmentation models have been trained. To achieve more accurate and robust predictions, these two models need to be integrated. Our group's innovative method for fusing these models involves using the training accuracy of each model as the prediction fusion weight for each class. This means that after generating the predictions from the 2D and 3D models, and before taking the 'argmax' of the images consisting of the probability of each class, each class will be multiplied by its corresponding weight. Then, these weighted probabilities for the same classes in both the 2D and 3D model outputs will be added together. The total probability of each class will then be normalized by dividing it by the sum of its 2D and 3D weights.
After obtaining the 'argmax' of the resultant output, the fused prediction image will be generated. This method takes into account both the 2D and 3D model predictions, leading to more accurate results.

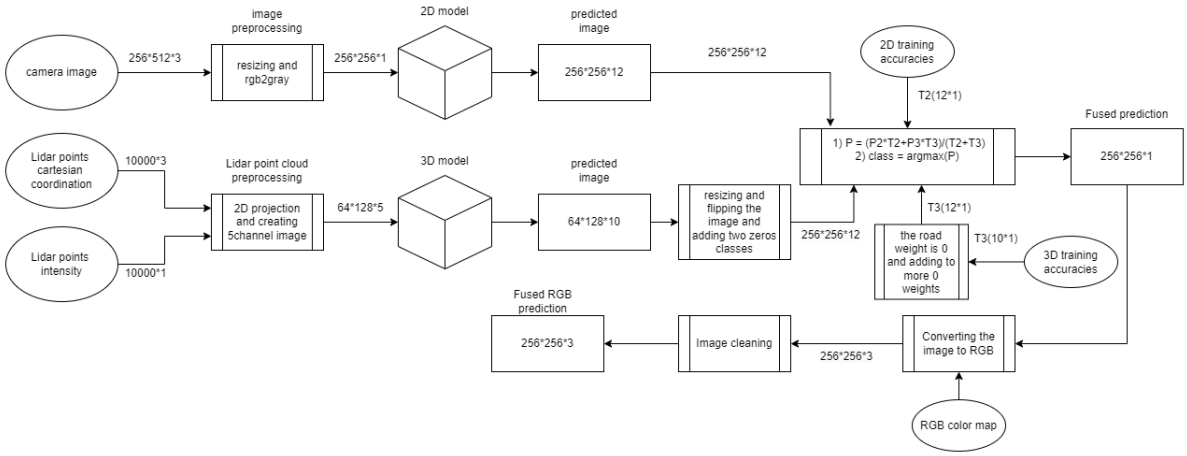


Figure 3.27: Overview of the fusion of the two models' outputs

As shown in Figure 3.27, some pre-processing steps are required before fusing the two predictions. Since the output size of the 2D model is larger than that of the 3D model, the 3D model's output needs to be resized and flipped. Due to limitations in IPG CarMaker, only the 2D model can determine the type of road (gravel, mud, snow). Consequently, the 3D model consists of 10 classes and predicts all road types as gravel. To facilitate the fusion, two extra classes (mud and snow) with zero probability have been added to the end of the 3D model output. The same adjustment has been made for the weights: two zero weights have been added at the end of the 3D prediction weights, and the weight of the gravel road has been set to zero. This ensures that the prediction of terrain types relies on the 2D model. The prediction of the sky also relies on the 2D model. Since the 3D model predicts the sky as unlabeled, it is advisable to assign zero prediction weights to the classes 'unlabeled' and 'sky' in the 3D prediction weight array.

Object class	2D prediction weight(T_2)	3D prediction weight(T_3)
unlabeled	0.99	0.00
log	0.82	0.71
stone	0.89	0.78
side road	0.89	0.82
sky	0.90	0.00
terrain	0.93	0.89
small cone	0.45	0.09
gravel road	0.98	0.00
vegetation	0.86	0.95
cone	0.81	0.42
muddy road	0.97	0.00
snowy road	0.98	0.00

Table 3.5: 2D and 3D prediction weights of each class

After obtaining the 'argmax' of the fused prediction and converting it to an RGB image using an RGB color map, an image cleaning process is performed. During this process, spots with a size of 5×5 pixels or smaller are eliminated and replaced by their neighboring RGB pixels.

3.5.3 Prediction Performance

During the testing phase, the trained models are evaluated using test scenarios alongside ground truth data. This process enables an assessment of the prediction performance of the models in practical settings. By comparing the predicted results with the ground truth data, we can quantify the accuracy of the predictions. Prediction accuracy is the percentage of correctly classified instances, which reflects the proportion of predictions that match the ground truth labels.
The test scenario encompasses various elements, including stones, logs of varying sizes and orientations, cones of different sizes, as well as background objects like vegetation, sky, terrain, and gravel road. Notably, side roads and small cones have been omitted from the test scenario. The test scenario has been illustrated in Figure 3.28.

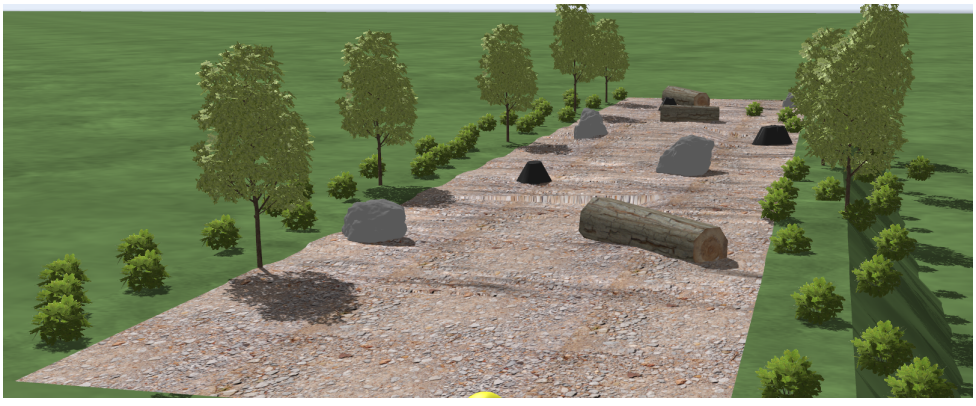


Figure 3.28: The test scenario created in IPG

Object class	3D mIoU (%)	2D mIoU (%)	Fused mIoU (%)	Δ= Fused - 2D (%)
unlabled	0.00	NaN	0.00	NaN
log	60.37	73.55	80.36	6.81
stone	52.49	58.39	63.13	4.74
side road	0.00	0.00	0.00	0.00
sky	0.00	89.78	89.83	0.05
terrain	24.42	40.97	40.85	-0.12
small cone	0.00	0.00	0.00	0.00
gravel road	75.54	89.82	89.82	0.00
vegetation	47.05	81.34	79.74	-1.6
cone	9.20	61.37	63.76	2.39
muddy road	NaN	0.00	0.00	0.00
snowy road	NaN	0.00	0.00	0.00

Table 3.6: The prediction performance of the 2D, 3D and the fused model. Δ IoU is the differenc between the fused IoU and 2D IoU.

As depicted in Table 3.6, the fused image demonstrates a Δ IoU of 0 for gravel road, indicating that it has been accurately extracted from the 2D prediction image. The table shows significant improvement in predicting small objects, with the fused image achieving higher mIoUs (mean intersection over union) for log (6.81%), stone (4.74%), and cone (2.39%) compared to the 2D predicted images. While the fused mIoU for some objects, like vegetation, has been reduced, this is less significant since small objects are considered critical traffic elements in off-road scenarios, whereas trees and bushes are mostly located outside the road.

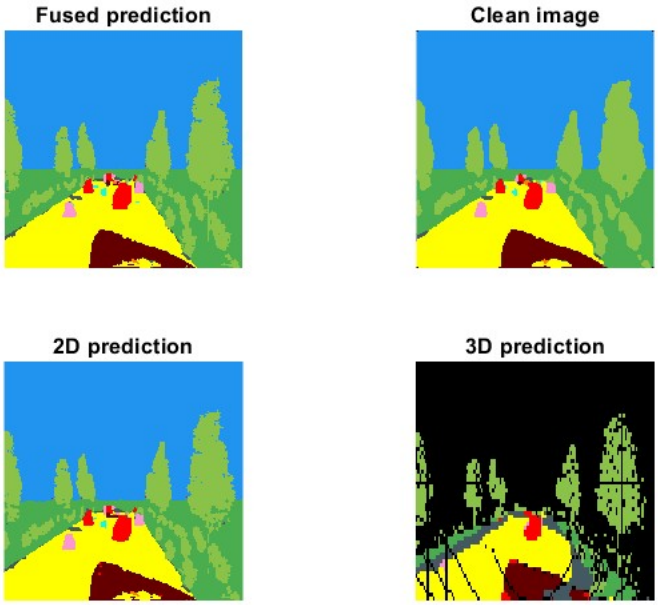


Figure 3.29: Comparison of the 2D, 3D and fused predicted images

3.6 Mathematical model

In the conclusive stage, we process invaluable information, ranging from predicted classes to the spatial distribution of objects within the scene. The subsequent phase involves a meticulous delineation of boundary boxes around the identified classes, a task achieved through a multifaceted array of methodologies 3.30. These techniques are designed to refine the raw data, effectively enhancing its utility and facilitating comprehensive analysis.

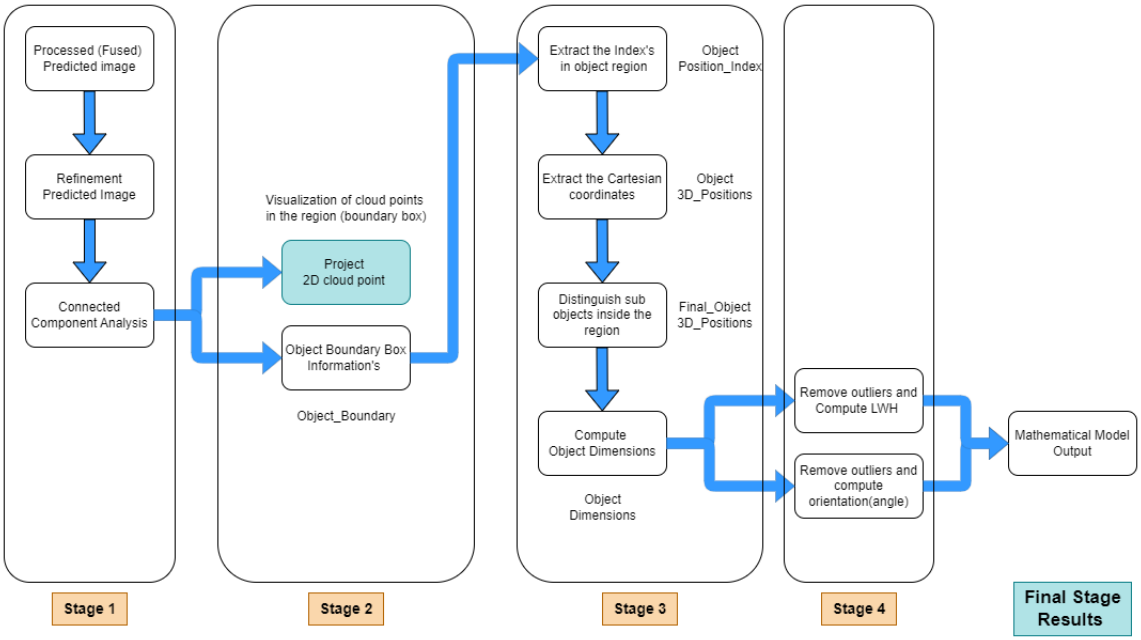


Figure 3.30: Process Flow of Mathematical Model

Stage I: The initial phase of our methodology revolves around enhancing the accuracy of the input predicted Fusion data. This process begins with the ingestion of the image input, followed by the creation of a mask image tailored to our defined classes and their respective LabelIDs 3.2. Subsequently, the mask image undergoes a transformation into a binary representation, wherein pixels not aligning with the specified criteria (LabelIDs) are rendered as white (255).

This binary rendition is then reconverted to the RGB format to preserve color information pertinent to matching pixels. Employing an element-wise multiplication technique in conjunction with the complement of the binary image ensures the retention of colors for matching pixels, while overlaying the binary image guarantees non-matching pixels are appropriately set to white, culminating in a refined and discernible composite image. The outcome is shown in Fig: 3.31.

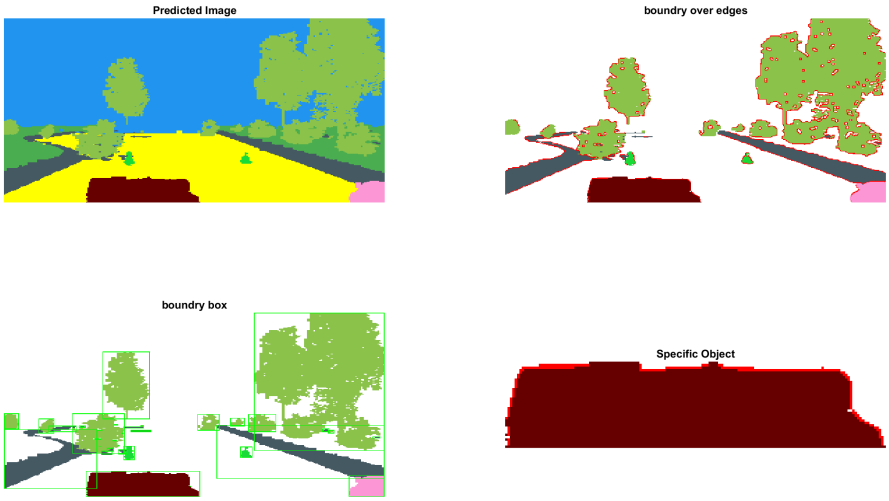


Figure 3.31: Distinguish the Object on an Image. Top-left is the predicted fused image. Top-right image is the refined edge detection. Bottom-Left image is the boundary box over an object. Bottom-Right image represents some specific object

Our subsequent aim is to accurately outline boundary boxes around diverse predicted classes, employing a diverse range of methodologies. This includes the utilization of edge detection techniques followed by Connected Component Analysis (CCA). The process begins with edge detection, a method that identifies significant changes in pixel intensity, effectively delineating object boundaries. Following this, CCA is applied to group adjacent pixels into connected components, thereby forming distinct regions corresponding to individual objects.To ensure precision, a threshold value is meticulously configured to eliminate noise within these regions, enhancing the accuracy of object delineation. Subsequently, the CCA algorithm is employed to delineate precise boundary boxes around the detected regions, providing comprehensive information regarding the spatial extent of each object within the scene.

This meticulous approach ensures the extraction of reliable boundary box information for all identified objects and saved as a list with m*5, where 'm' (row) is defined as number_of_objects and 5 (columns) defines the object_ID (object Id is a Label_IDs of each classes) and boundary_box_info (xmin, ymin, xmax, and ymax). Visual representations of the processed data, complete with bounding boxes encapsulating individual objects, provides comprehensive overview of the scene. This intuitive visualization 3.31 facilitates qualitative assessments and aids in the identification of potential anomalies or irregularities within an image.

Stage II: Following the boundary box delineation process, the next crucial step involves projecting cloud points onto a 2D plane for visualization 3.4 for verification purposes, ensuring the preservation of valuable cloud point data 3.32. Simultaneously, the extraction of object positions emerges as a pivotal juncture in our analyt-

ical framework. This iterative process entails the meticulous analysis of bounding boxes, where the x and y dimensions of each box are leveraged to filter and catalog cloud point indices corresponding to individual objects.

This meticulous approach yields an output two matrix. one matrix with dimensions n*2, where 'n' represents the number of objects identified within the scene multiplied by number of cloud points for each object. Each row of the matrix encapsulates vital information, including the object_ID and the associated cloud point indices. Another matrix with the dimensions m*1, where 'm' represents Object_Number and number of cloud points belongs to each object.

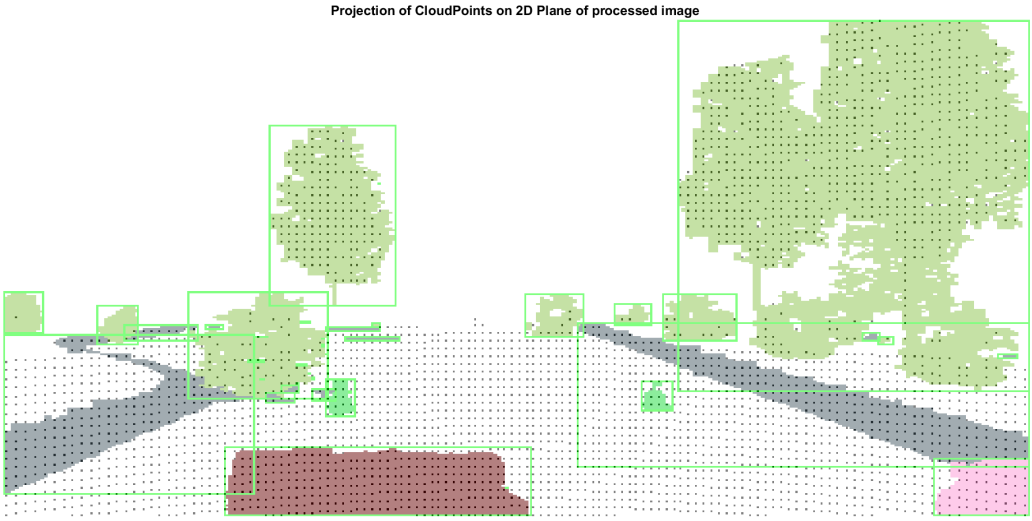


Figure 3.32: Projection of cloud points on the 2D plane on an Processed image for the visualization

Stage III: Utilizing the previously obtained matrices—one containing object_IDs and their associated cloud point indices, and the other detailing the number of cloud points for each object, we proceed to correlate these indices with the processed 3D cloud points. By comparing the indices, we extract the Cartesian coordinates for each object, adhering to the conversion formula (3.1). These coordinates, comprising the object number, ID, and respective x, y, and z coordinates, are then meticulously saved into an array.

The resultant array possesses dimensions s*5, where 's' denotes the total number of objects identified within the scene multiplied by the number of cloud points associated with each object. Each row of the array encapsulates vital information, including the object number, ID, and the corresponding 3D Cartesian coordinates. This comprehensive array facilitates a detailed understanding of the spatial distribution of objects within the scene, providing crucial insights for further analysis and interpretation.

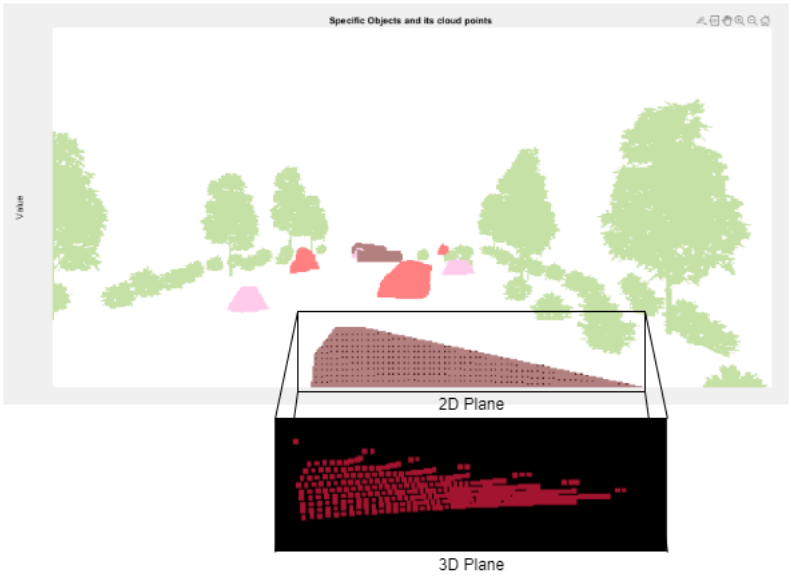


Figure 3.33: Extracting 3D Cartesian coordinates utilizing 2D coordinates indices

In pursuit of heightened computational efficiency and the refinement of data significance, a function has been developed to discern sub-objects within a given region. Through this implementation, computational resources are streamlined to focus on pertinent data exclusively. This function operates by conducting comparisons across x and y axes to determine whether an object pertains to its corresponding object or to another entity. Such categorization facilitates a more precise information about an object 3.34.

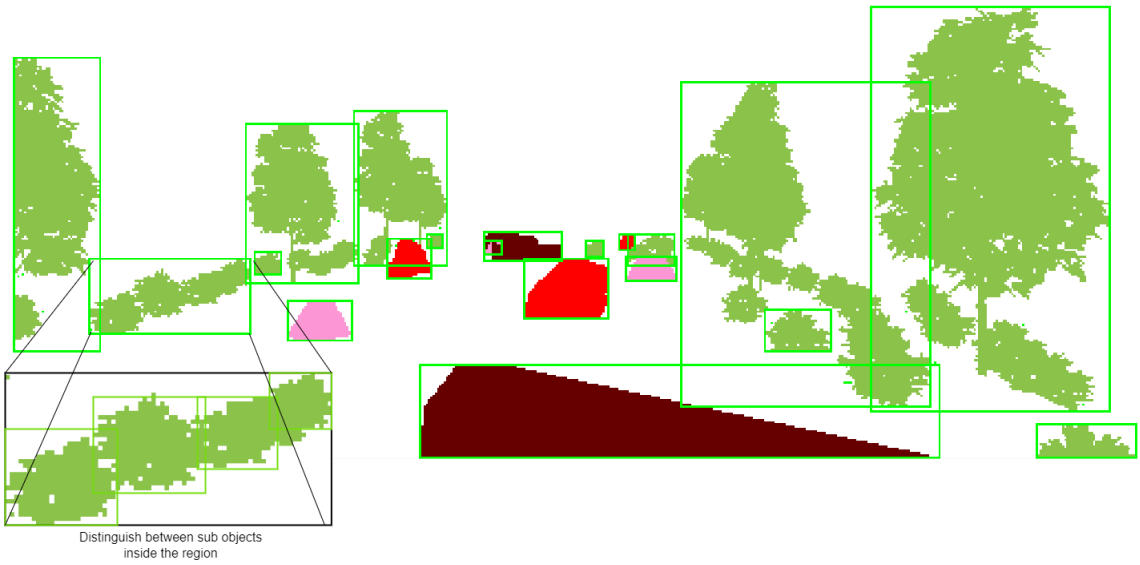


Figure 3.34: Distinguish sub-objects within the region

Moreover, additional threshold value is used which serves as a filter, systematically removing unnecessary data points from our analysis pipeline. Specifically, objects smaller than the threshold value are automatically excluded from further processing. Additionally, objects lacking any cloud point information are deemed outside the required range and consequently omitted from consideration. This meticulous filtering process ensures that only pertinent and informative data are retained for subsequent analysis. By eliminating extraneous data points, we streamline computations and enhance the overall quality of our results. The end result is a dataset enriched with meaningful information, conducive to more accurate and efficient analysis processes.

Stage IV: In the penultimate phase of our analysis, we focus on refining the dataset by removing outliers associated with each object. These outliers, characterized by data points that deviate significantly from the surrounding neighborhood, have the potential to distort calculations and compromise the accuracy of our results. To address this, outlier removal is performed across all three dimensions of the Cartesian coordinates. By systematically identifying and eliminating outliers, we safeguard our analysis from erroneous conclusions and ensure the integrity of our findings. This meticulous process serves to enhance the robustness of our dataset, fortifying it against the influence of noisy or irrelevant data points. The refined dataset underscores the improved quality and reliability of our analysis.

In the culmination of our mathematical model, we embark on the final step: computing the dimensions of each object. This pivotal process begins by determining the minimum and maximum coordinates along each axis—X, Y, and Z—relative to the lidar positions. By identifying these extremes, we establish the boundaries within which each object exists.

Subsequently, we leverage these min-max coordinates to calculate the dimensions of the object. The width is derived by subtracting the minimum X-coordinate from the maximum X-coordinate, encapsulating the object's span along the X-axis. Similarly, the length is determined by the difference between the maximum and minimum Y-coordinates, representing the object's extent along the Y-axis. Finally, the height is computed as the difference between the maximum and minimum Z-coordinates, capturing the object's vertical dimension.

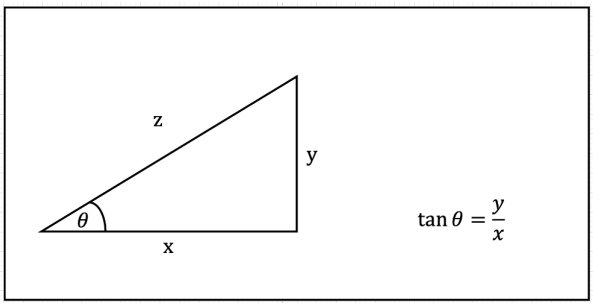


Figure 3.35: Trigonometric formula to find angle

One of the challenging aspects of the mathematical model lies in determining the

angles of objects relative to the LiDAR, a crucial piece of information for decision-making within an analytical framework. This involves leveraging data from the x and y axes, which provide a top-down view. By applying linear regression techniques to these coordinates, a linear line can be drawn to represent the cloud points 3.36. Subsequently, the task is to construct a parallel line originating from the LiDAR's point of reference. Utilizing trigonometric formulas, the angle can then be computed based on the opposite divide by adjacent lengths **??**. This process is essential for accurately assessing the orientation of objects in relation to the LiDAR sensor, thereby enhancing the analytical capabilities of the model.

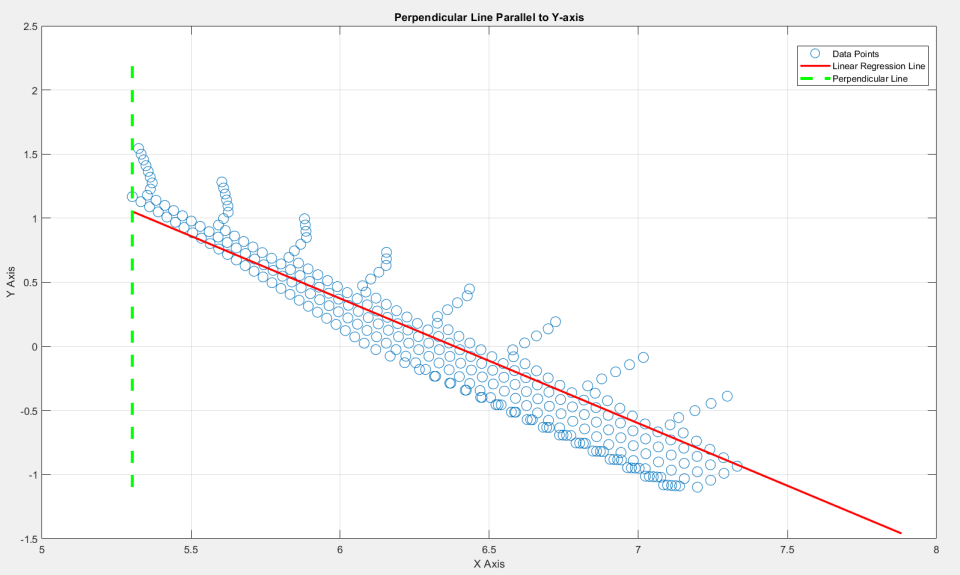


Figure 3.36: Computing object angle wit x and y axis data of a cloud points

This meticulous process ensures that each dimension is accurately quantified with respect to the lidar positions, providing crucial insights into the spatial characteristics of the objects under analysis. The final mathematical model return data With dimensions meticulously calculated, we gain a comprehensive understanding of the size and spatial arrangement of objects within the scene, facilitating informed decision-making and further scientific inquiry.

3.7 Analitical Model

3.7.1 Boolean Map

The output of the mathematical model is a matrix consisting of the objects' IDs (grayscale color) that determine the class of the object, objects' positions $(X_{min}, Y_{min}, Z_{min})$, objects' dimensions (width, length, height), and objects' orientations.
The next step is to create a boolean map consisting of drivable and non-drivable areas. This will be accomplished through the following steps:

1. Draw the road area.

2. Eliminate all objects outside the road area.
3. Eliminate objects whose height is lower than the chassis height(Marking them as derivable).
4. Obtain the four corners of each object from a top view.
5. Determine the center point of each object.
6. Translate the corner points using the center point coordinates, making the center point the origin.
7. Rotate the corners according to the object's orientation.
8. Translate the corners back to their real positions.

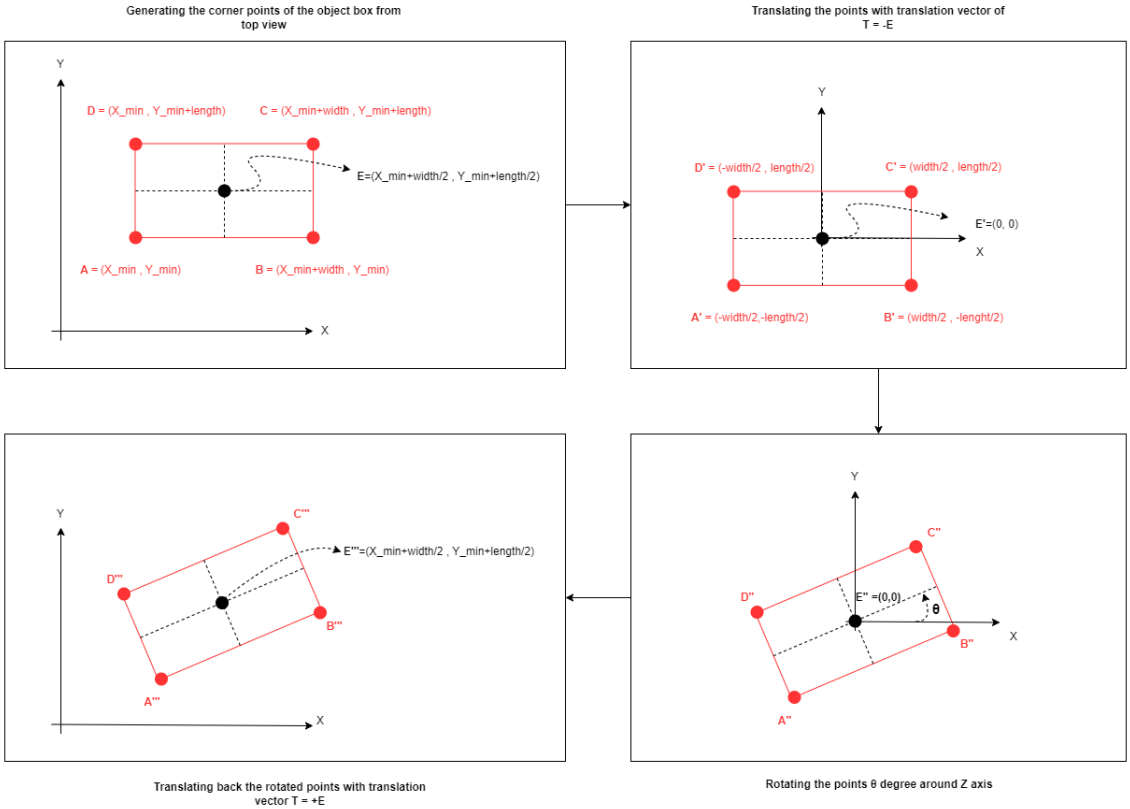


Figure 3.37: The illustration of steps above

After implementing the aforementioned steps, the next stage involves further filtering. Figure 3.38 illustrates three types of filters that have been utilized. The first filter corresponds to point 3 above. Width filtering has been conducted in two phases. Initially, the distance between the nearest corner of the object box and the side road is measured. If this distance is less than the vehicle width, it indicates a potential risk of the vehicle getting stuck, necessitating the prevention of collision by drawing a straight line between the object and the side road. The subsequent width filtering occurs between two objects. If the nearest corners of two objects have a distance lower than the vehicle width, a straight line will be drawn between them to prevent the vehicle from getting stuck or colliding with the objects.

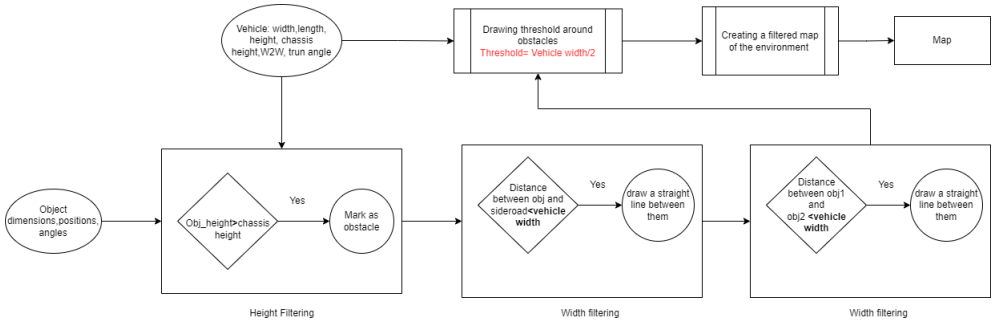


Figure 3.38: Obstacle filtering steps based on the vehicle properties

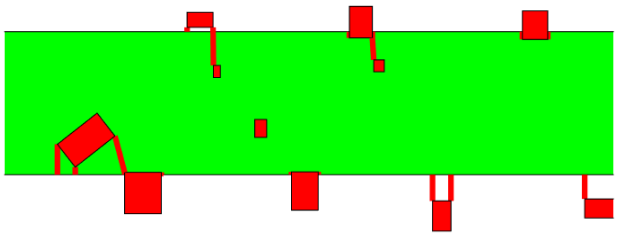


Figure 3.39: Map with filtered obstacles

Following all the filtering steps, a threshold area will be drawn around the boxes. The rationale behind this and the specific size of the area will be elucidated later.

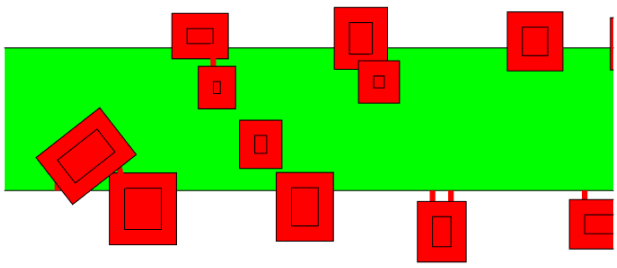


Figure 3.40: Map with filtered obstacles and threshold around the objects

3.7.2 Turning radius

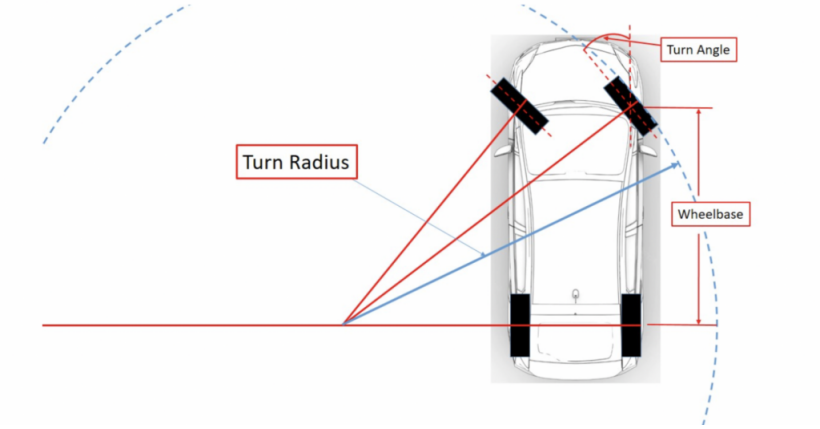


Figure 3.41: Relation between turning angle and truning raduis [30]

The equation below demonstrates the method for determining the turning radius based on the turning angle and the distance between the wheels according to Figure 3.41.

$$Turn\ Radius = Wheelbase/sin(Turn\ Angle); \tag{3.7}$$

3.7.3 Path planning

The RRT* algorithm has been employed to determine a safe and efficient path towards the goal point. As the planned path represents the trajectory of the vehicle's center point, it is necessary to establish a threshold (half of the vehicle's width) around objects to ensure the safety of the paths for both the left and right tires, preventing them from colliding with the objects.However, a crucial constraint that has yet to be addressed is the vehicle's turning radius. Upon computing the vehicle's turning radius using Equation (3.7), modifications have been made to the RRT* algorithm to accommodate this parameter. As outlined in the theoretical framework of the RRT* algorithm, each iteration involves the generation of a random point within the navigable area, followed by the identification of its nearest point and the determination of its parent. To align with the vehicle's wheel-to-wheel length, which defines the step size, the following procedures are undertaken to ascertain the angle between $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$:

$$\overrightarrow{V_1} = \overrightarrow{nearset - random}\ , \overrightarrow{V_2} = \overrightarrow{nearset - parent} \Rightarrow cos(\theta) = \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{||\overrightarrow{V_1}|| * ||\overrightarrow{V_2}||} \tag{3.8}$$

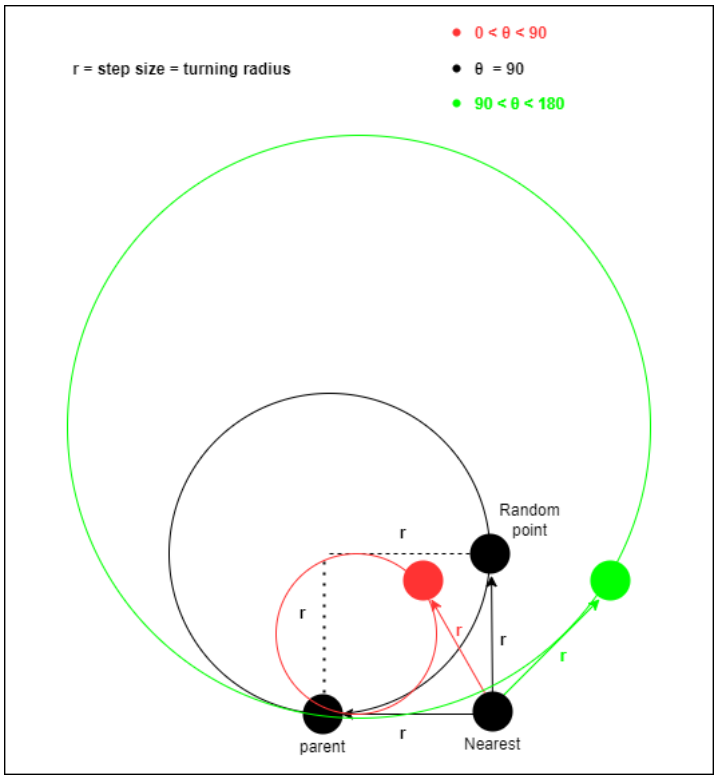


Figure 3.42: Comparison of the angular deviation between the parent, nearest, and random points

As Figure 3.42 illustrates, if the angle between these three points exceeds 90 degrees, the radius of the turning circle will be greater than the turning radius (green circle). If the angle is exactly 90 degrees, the circle's radius will be equal to the turning radius (black circle). In both of these scenarios, the vehicle can easily follow the circular path without any issues. Conversely, if the angle between these three points is less than 90 degrees, the turning circle's radius will be smaller than the turning radius (red circle), making it impossible for the vehicle to execute such sharp turns. After generating a new point, the angle between that point, its nearest point, and the nearest point's parent should be calculated. If this angle is less than 90 degrees, the new point should be discarded.

3.8 Experiments and results

In this section, three validation scenarios have been designed to evaluate the performance of the final model.

3.8.1 Scenario 1

This scenario includes logs with varying orientations and other objects. The objective of this experiment is to assess the mathematical model's ability to accurately calculate the different orientations of the logs. Additionally, the road surface comprises three distinct surface types(see Figure 3.43), aiming to evaluate how well the

prediction model can handle a mixture of different road types.

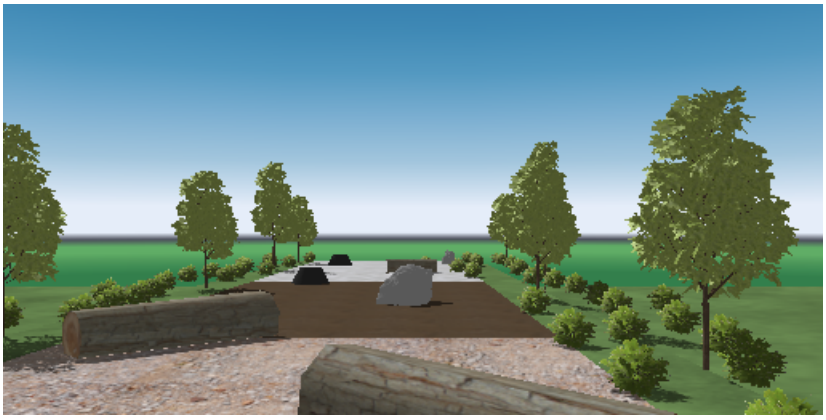


Figure 3.43: Camera image of scenario 1

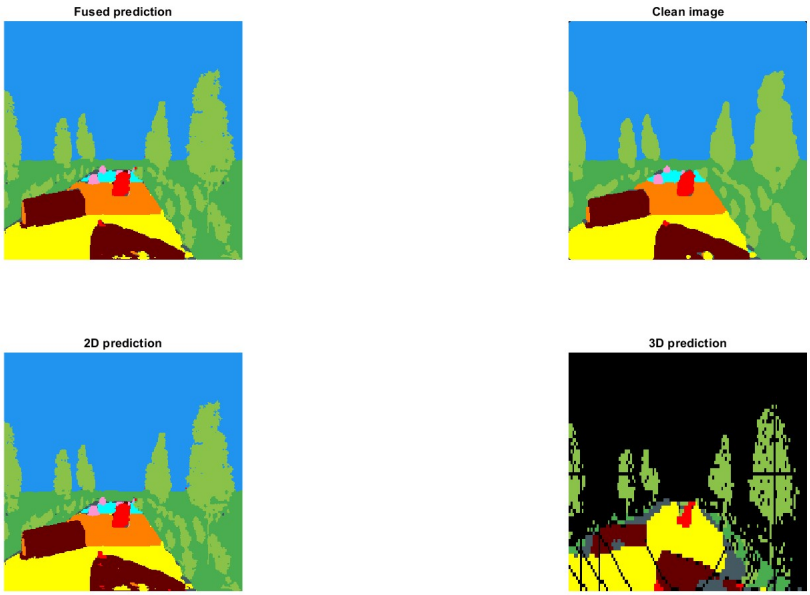


Figure 3.44: 2D, 3D, and fused models' predictions as well as the clean fused image

As illustrated in Figure 3.44, the fused prediction image is more accurate. Small erroneous predictions from the 2D model, such as minor red areas on the log, have been eliminated in the fused prediction image. The cleaned image further enhances accuracy by removing small spots from the fused image.

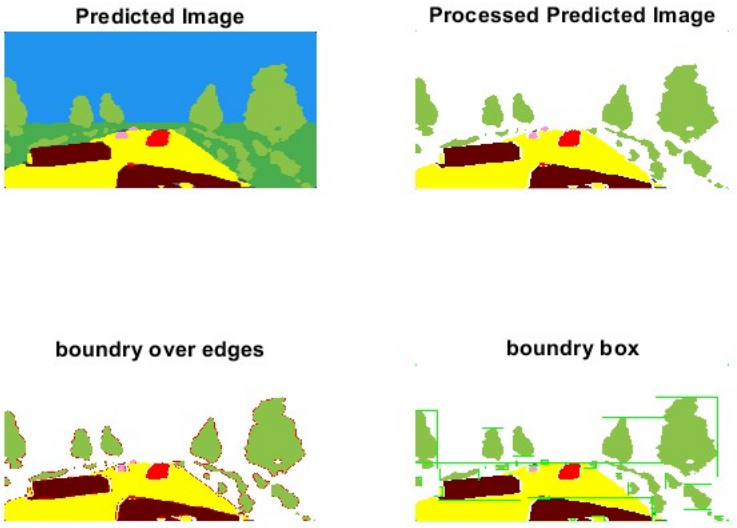


Figure 3.45: Different phases of mathematical model

Figure 3.45 illustrates the drawn boundary boxes around each object. It is evident that the algorithm considers different terrain types as a single, unified terrain.

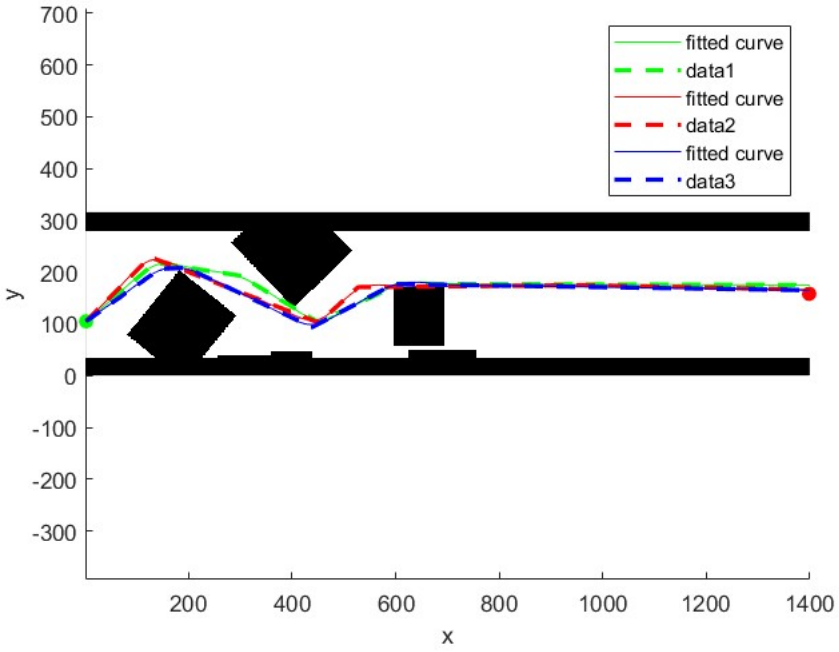


Figure 3.46: The boolean map of the obstacles in front of the vehicle with threshold around around them. The figure consists of three distinct and safe deriving paths.

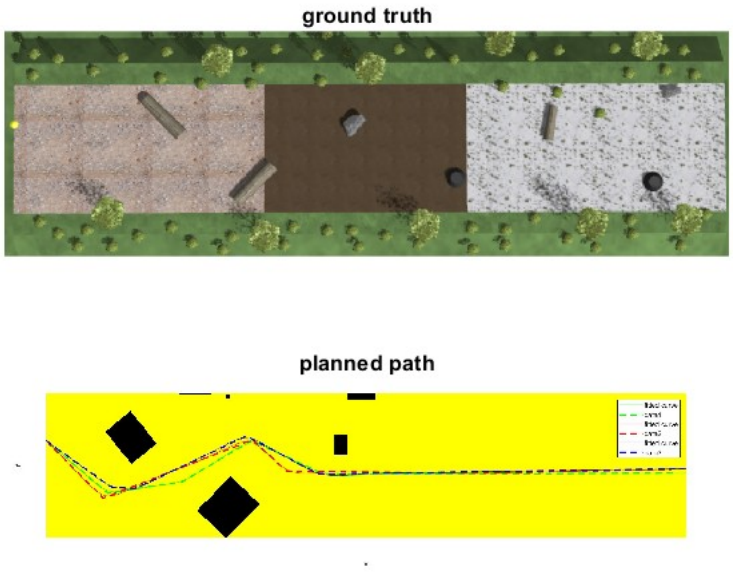


Figure 3.47: A comparison between the real map and the boolean map of the scenario is presented. The yellow point in the ground truth image represents the center point of the vehicle, while the starting point of the planned paths corresponds to the front of the vehicle.

Based on Figure 3.47, the mathematical model successfully determined the orientation of the logs. However, a discrepancy exists between the calculated width of the logs and their ground truth width. This inconsistency can be attributed to the presence of outlier lidar points associated with the logs. Objects located at a distance were either undetected by the model or were of such small size that they were removed by the obstacle filtering model. Despite this, the three planned paths appear to be smooth, lacking sharp turns.

3.8.2 Scenario 2

This scenario presents a mixture of different road types, arranged differently from Scenario 1 (see Figure 3.48). Additionally, it includes multiple cones positioned closer together than the vehicle width. Therefore, we anticipate that the model's output will indicate that the goal is unreachable and driving through the cones is unsafe.

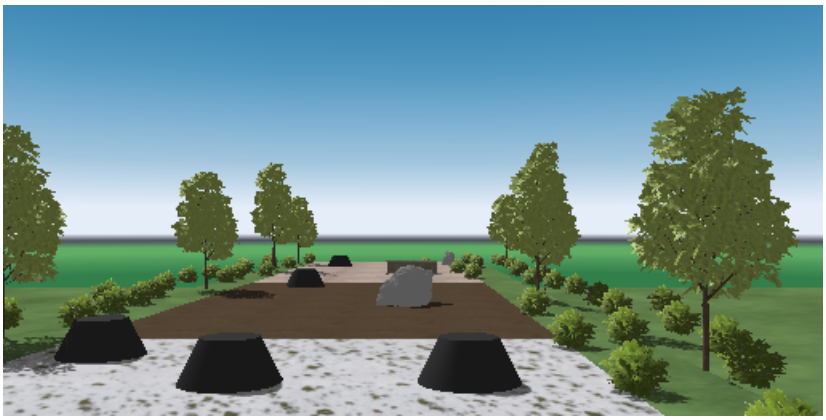


Figure 3.48: Camera image of scenario 2

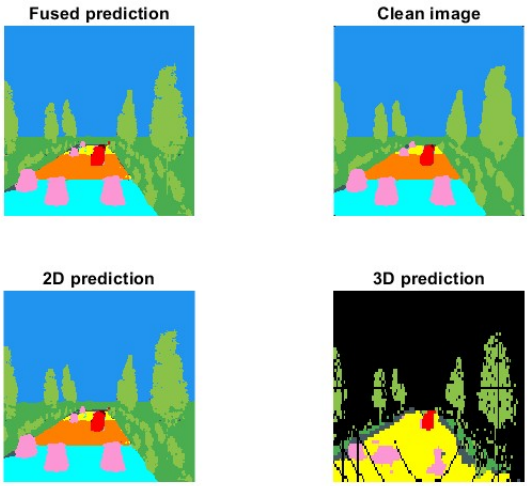


Figure 3.49: 2D, 3D, and fused models' predictions as well as the clean fused image

As illustrated in Figure 3.49, the fused prediction image is more accurate. Small erroneous predictions from the 2D model, have been eliminated in the fused prediction image. The cleaned image is removing small spots from the fused image.

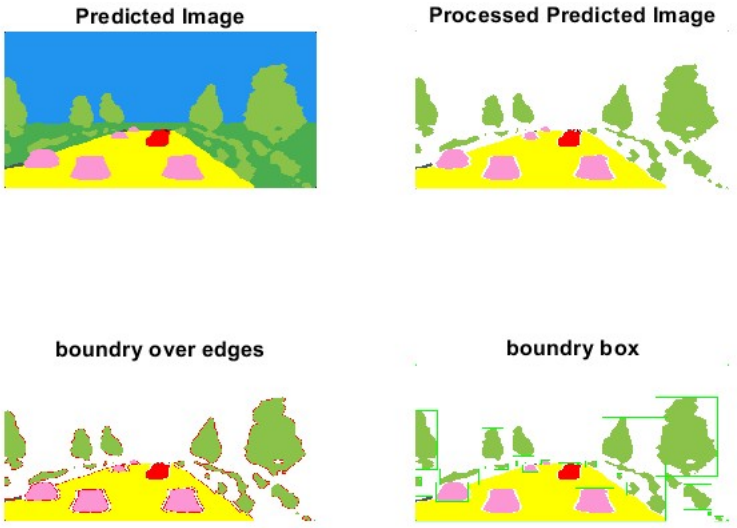


Figure 3.50: Different phases of mathematical model

Figure 3.50 illustrates the drawn boundary boxes around each object. It is evident that the algorithm considers different terrain types as a single, unified terrain.

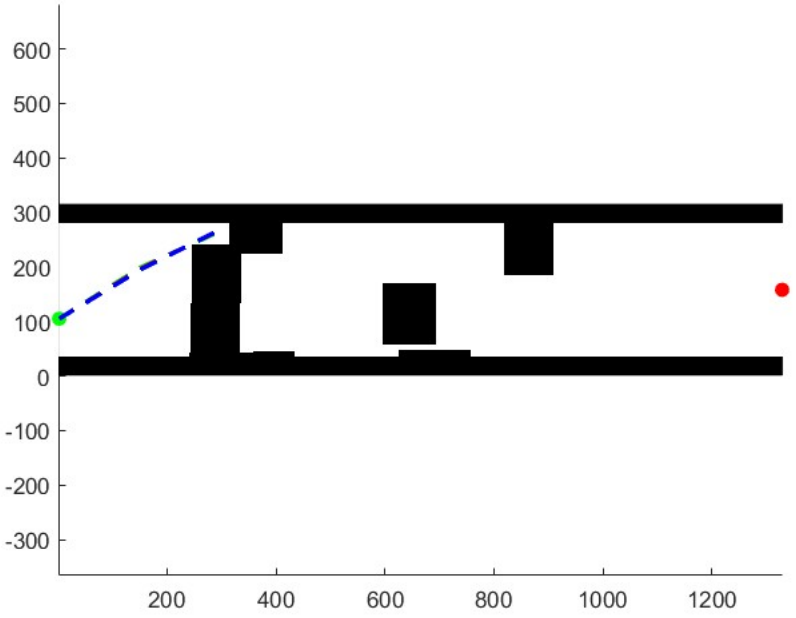


Figure 3.51: The boolean map of the obstacles in front of the vehicle with threshold around around them. The figure consists of one deriving path.

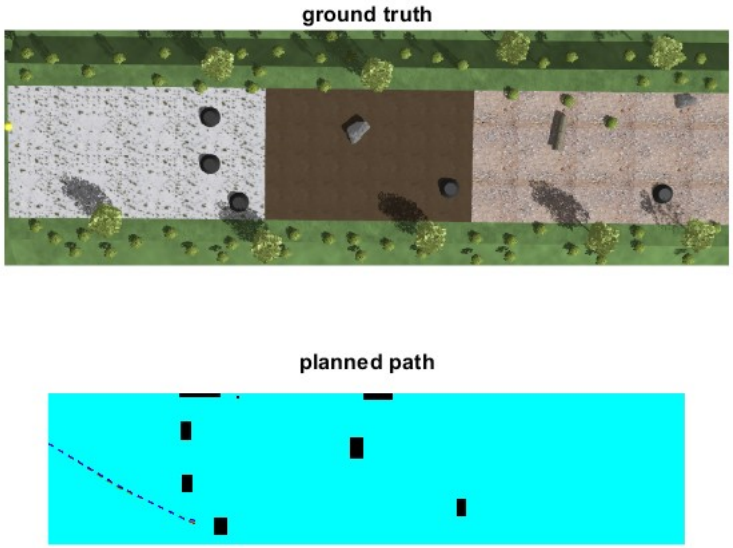


Figure 3.52: A comparison between the real map and the boolean map of the scenario is presented. The yellow point in the ground truth image represents the center point of the vehicle, while the starting point of the planned paths corresponds to the front of the vehicle.

According to Figure 3.52, the mathematical model accurately determined the physical properties (width, length) of the objects, along with their positions relative to the front of the vehicle. The model's performance indicates that the goal is unreachable and that the vehicle can safely navigate through the cones. Objects situated at a distance were either not detected by the model or were removed by the obstacle filtering model due to their small size.

3.8.3 Scenario 3

The road type designated for this scenario is a muddy road (refer to Figure **??**). Additionally, it features multiple cones arranged in a zigzag pattern in front of the vehicle. It is anticipated that the model's planned path will incorporate numerous turns, and the goal will be reachable.

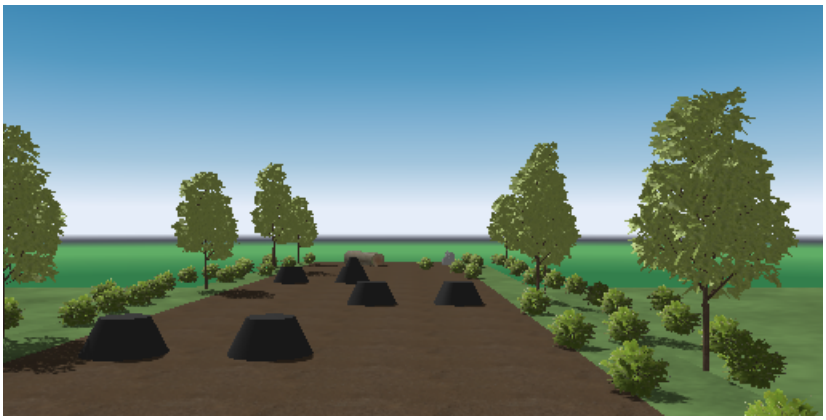


Figure 3.53: Camera image of scenario 3

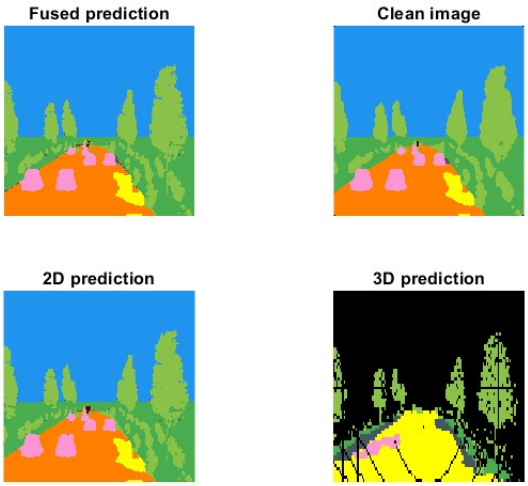


Figure 3.54: 2D, 3D, and fused models' predictions as well as the clean fused image

As illustrated in Figure 3.54, the fused prediction image is more accurate. Small erroneous predictions from the 2D model, have been eliminated in the fused prediction image. The cleaned image is removing small spots from the fused image.

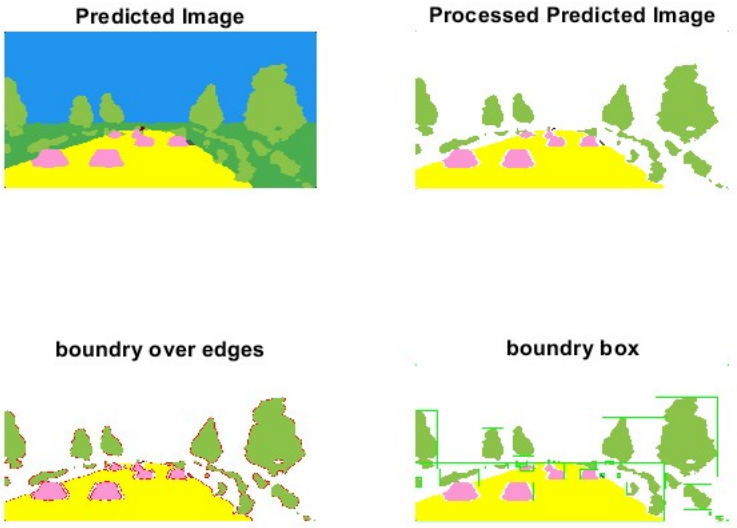


Figure 3.55: Different phases of mathematical model

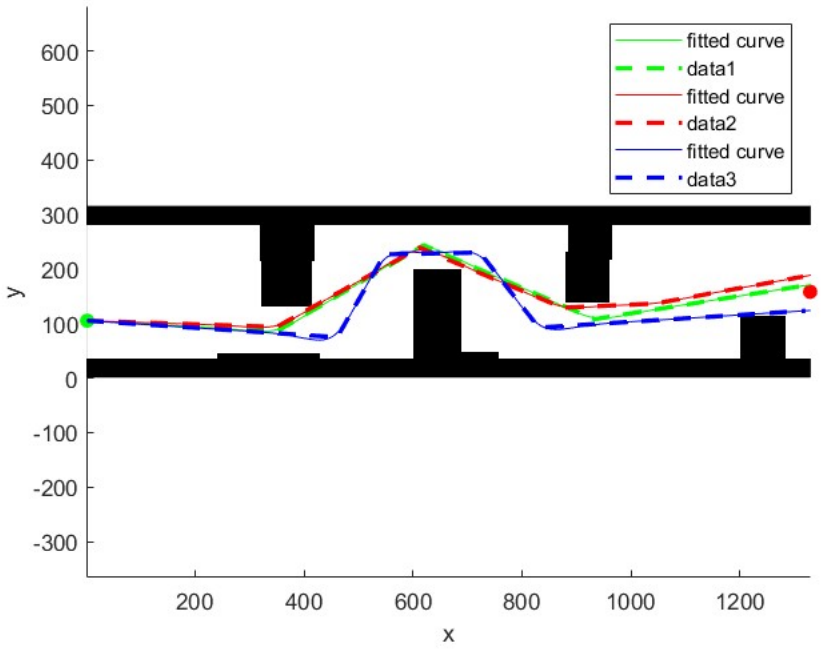


Figure 3.56: The boolean map of the obstacles in front of the vehicle with threshold around around them. The figure consists of three distinct and safe deriving paths.

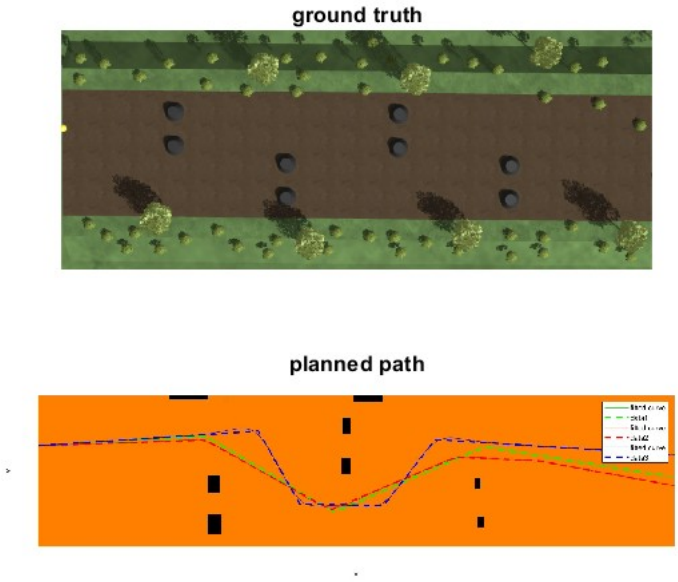


Figure 3.57: A comparison between the real map and the boolean map of the scenario is presented. The yellow point in the ground truth image represents the center point of the vehicle, while the starting point of the planned paths corresponds to the front of the vehicle.

Upon comparing the ground truth image in Figure 3.57 with the camera image in Figure 3.53, it is evident that the first two cones were not detected by the camera. Consequently, the planned path image depicted in Figure 3.57 does not include these two cones.
According to Figure 3.57, the mathematical model accurately determined the physical properties (width, length) of the objects, along with their positions relative to the front of the vehicle. The model's performance indicates that the goal is reachable and that the vehicle can safely navigate through the cones. The planned paths consist of multiple turns.

4

Conclusion

This thesis presents an improved approach to terrain detection and obstacle classification through an enhanced sensor fusion algorithm. The research effectively addresses the challenges of accurately assessing environmental conditions, delivering precise and reliable information about both terrain types and obstacles. The study culminated in the successful development and validation of an algorithm capable of accurately identifying terrain types and classifying obstacles. The experimental results detailed in this work demonstrate that the use of advanced sensor fusion, accurate mathematical modeling, and precise analytical methods leads to superior estimation and trajectory map generation. This enhanced robustness and accuracy provide drivers with a detailed and reliable depiction of the terrain, significantly improving the overall driving experience.

It is evident that as shown in Figure 3.29, neither the 2D nor the 3D models provide accurate predictions for distant objects. In the 2D model, distant objects occupy only a few pixels in the camera image, resulting in unclear and insufficiently detailed detection. Consequently, the 2D model struggles to predict these objects accurately. Similarly, the 3D model encounters challenges with distant object prediction due to the limited number of point clouds available, making it difficult to establish meaningful relationships between these sparse data points and leading to inaccuracies.

During the initial training of the 3D model, the scenario was kept consistent with the 2D model. However, the imbalanced dataset led to low accuracy and Intersection over Union (IoU) results, particularly for smaller objects. To address this, an alternative scenario was created that included some more small objects, resulting in a more balanced dataset and significant improvements in the model's output. Since the simulator tool has a limitation of generating a maximum of 10,000 point clouds. However, for optimal training, the number of point clouds should ideally match the number of pixels in the annotated image, with the image size divisible by 64. This restricts the maximum allowable point clouds to 64×128, or 8,192 point clouds. This limitation is challenging, as it may not provide enough data points for accurate predictions. To improve the 3D model's performance, access to a higher number of point clouds would be essential.

In conclusion, this research has made significant strides in the field of off-road driving by developing an advanced sensor fusion algorithm that significantly enhances terrain detection and obstacle classification (Positive and Hanging obstacles). The successful integration of 2D and 3D modeling, despite the inherent challenges with

distant object prediction, underscores the effectiveness of the proposed approach. By addressing the limitations of traditional models and improving the balance of datasets used for training, this work has demonstrated marked improvements in accuracy and reliability. The algorithm's ability to deliver precise environmental assessments and superior trajectory map generation not only advances the state of the art but also promises to substantially improve the safety and overall driving experience in off-road environments.

5

Future Works

The current algorithm developed for terrain detection and obstacle classification for better guidance, several key areas of improvement should be addressed to enhance the system's performance in dynamic environments. These improvements include the integration of real sensors, training a comprehensive model, estimating vehicle orientation, enabling dynamic adaptation, and considering the shape and friction of obstacles

5.1 Utilization of Real Sensors

While the current algorithm may rely on simulated or limited data, future work should prioritize the integration of real sensors to capture actual environmental data. Real-world sensors such as high-resolution cameras, and LiDAR sensors can provide accurate and diverse data, essential for improving the robustness of the terrain detection and obstacle classification system. By incorporating data from these sensors, the algorithm will be able to handle a wider range of environmental conditions and complexities, such as varying lighting, weather, and terrain types, which are often difficult to replicate in simulations. Moreover, real sensor data can help in refining the algorithm's ability to distinguish between different types of obstacles and surfaces more accurately.

5.2 Training a Comprehensive Model

Currently, the model may use separate systems for processing camera and LiDAR data. Future work should aim to train a comprehensive, unified model that simultaneously incorporates data from both these sensors. By fusing camera and LiDAR inputs, the model can leverage the complementary strengths of each sensor—cameras providing detailed visual information and LiDAR offering precise distance measurements. This combined approach can improve the accuracy of terrain classification, object detection, and depth estimation, leading to more reliable navigation and obstacle avoidance. A single, integrated model can also reduce computational complexity and streamline the processing pipeline, making the system more efficient and responsive in real-time applications.

5.3 Orientation Estimation

In dynamic environments, accurately estimating the vehicle's orientation is critical for effective navigation. The Inertial Measurement Unit (IMU) can play a pivotal role in this aspect by providing real-time data on the vehicle's pitch, roll, and yaw. Future research should focus on integrating IMU outputs into the algorithm to continuously update the vehicle's orientation. This integration will enhance the model's ability to predict the impact of terrain and obstacles on the vehicle's movement, enabling more precise control and stability, especially on uneven or sloped surfaces. Moreover, the IMU data can be used to compensate for any sensor drift or noise, further improving the reliability of the system.

5.4 Dynamic Adaptation

The current algorithm is designed for a static vehicle state, which limits its applicability in dynamic environments where the vehicle is in motion. Future developments should focus on extending the algorithm to handle dynamic scenarios, where the vehicle's speed, direction, and acceleration constantly change. This requires the model to be capable of real-time adaptation, adjusting its predictions and decisions based on the current state of the vehicle and the surrounding environment. Techniques such as real-time sensor fusion, predictive modeling, and adaptive control algorithms should be explored to ensure the system can respond effectively to rapid changes in terrain and obstacle configurations.

5.5 Shape and Friction Consideration

The physical properties of obstacles, such as their shape and friction, have a significant impact on the vehicle's ability to traverse or avoid them. Future research should incorporate these factors into the algorithm to improve its decision-making process. For instance, understanding the shape of an obstacle can help in determining whether it is traversable, climbable, or should be avoided entirely. Similarly, assessing the friction coefficient of different surfaces can inform the vehicle's speed and handling strategies, particularly in slippery or loose terrain conditions. By considering these physical properties, the algorithm can make more informed decisions, leading to safer and more efficient navigation.

5.6 Negative and Hanging Obstacles

In addition to positive obstacles (those protruding from the ground), negative obstacles (such as pits or ditches) and hanging obstacles (such as low-hanging branches or wires) pose significant challenges for navigation. The current algorithm primarily focuses on positive obstacles, but future research should expand its capabilities to detect and classify negative and hanging obstacles. This will require the development of specialized detection techniques, such as downward-facing sensors for

negative obstacles and upward-facing sensors for hanging obstacles. Additionally, the algorithm should be trained to recognize and respond to these types of obstacles, ensuring comprehensive obstacle awareness and avoidance.
Overall, By addressing these areas of improvement, the terrain detection and obstacle classification algorithm can be significantly enhanced for dynamic and complex environments. The integration of real sensors, comprehensive model training, orientation estimation, dynamic adaptation, and consideration of shape and friction will result in a more robust, reliable, and versatile system. This will enable the algorithm to navigate challenging terrains with greater accuracy and safer navigation.

Bibliography

[1] R. Manduchi, A. Castano, A. Talukder, *et al.*, "Obstacle detection and terrain classification for autonomous off-road navigation", *Autonomous Robots*, vol. 18, pp. 81–102, 2005. DOI: 10.1023/B:AURO.0000047286.62481.1d. [Online]. Available: https://doi.org/10.1023/B:AURO.0000047286.62481.1d.

[2] C. Ye, H. Pan, X. Yu, and H. Gao, "A spatially enhanced network with camera-lidar fusion for 3d semantic segmentation", *Neurocomputing*, vol. 484, pp. 59–66, 2022, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.12.135. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221015770.

[3] F. Islam, M. Nabi, and J. E. Ball, "Off-road detection analysis for autonomous ground vehicles: A review", *Sensors (Basel, Switzerland)*, vol. 22, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253354970.

[4] T. Liu, D. Liu, Y. Yang, and Z. Chen, "Lidar-based traversable region detection in off-road environment", in *2019 Chinese Control Conference (CCC)*, 2019, pp. 4548–4553. DOI: 10.23919/ChiCC.2019.8865250.

[5] J. Choi, J. Lee, D. Kim, *et al.*, "Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment", *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 974–982, 2012. DOI: 10.1109/TITS.2011.2179802.

[6] P. Duraisamy and S. Natarajan, "Multi-sensor fusion based off-road drivable region detection and its ros implementation", in *2023 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, 2023, pp. 1–5. DOI: 10.1109/WiSPNET57748.2023.10134440.

[7] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection", in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4603–4611. DOI: 10.1109/CVPR42600.2020.00466.

[8] A. Abdelkader and M. Moustafa, "Camera and lidar fusion for point cloud semantic segmentation", in *Proceedings of Seventh International Congress on Information and Communication Technology*, X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi, Eds., Singapore: Springer Nature Singapore, 2023, pp. 499–508, ISBN: 978-981-19-2394-4.

[9] Z. Zhuang, R. Li, Y. Li, K. Jia, Q. Wang, and M. Tan, "Perception-aware multi-sensor fusion for 3d lidar semantic segmentation", *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16 260–16 270, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235670035.

[10] B. Gao, A. Xu, Y. Pan, X. Zhao, W. Yao, and H. Zhao, "Off-road drivable area extraction using 3d lidar data", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1505–1511. DOI: 10.1109/IVS.2019.8814143.

[11] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1110–1116. DOI: 10.1109/ICRA48506.2021.9561251.

[12] S. Sharma, L. Dabbiru, T. Hannis, *et al.*, "Cat: Cavs traversability dataset for off-road autonomous driving", *IEEE Access*, vol. 10, pp. 24 759–24 768, 2022. DOI: 10.1109/ACCESS.2022.3154419.

[13] Y. Jin, D. Han, and H. Ko, "Memory-based semantic segmentation for off-road unstructured natural environments", in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 24–31. DOI: 10.1109/IROS51168.2021.9636620.

[14] I. Sgibnev, A. Sorokin, B. Vishnyakov, and Y. V. Vizilter, "Deep semantic segmentation for the off-road autonomous driving", *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 617–622, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221615905.

[15] O. Mayuku, B. W. Surgenor, and J. A. Marshall, "Multi-resolution and multi-domain analysis of off-road datasets for autonomous driving", in *2021 18th Conference on Robots and Vision (CRV)*, 2021, pp. 165–172. DOI: 10.1109/CRV52889.2021.00030.

[16] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6230–6239. DOI: 10.1109/CVPR.2017.660.

[17] H. Alokasi and M. B. Ahmad, "The accuracy performance of semantic segmentation network with different backbones", in *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*, 2022-03, pp. 49–54. DOI: 10.1109/CDMA54072.2022.00013.

[18] Z. Gao, Q. Wang, Z. Pan, Z. Zhai, and H. Long, "Pointpainting: 3d object detection aided by semantic image information", *Sensors*, vol. 23, no. 5, 2023, ISSN: 1424-8220. DOI: 10.3390/s23052868. [Online]. Available: https://www.mdpi.com/1424-8220/23/5/2868.

[19] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud", *UC Berkeley*, 2018, https://arxiv.org/pdf/1809.08495.

[20] Y. Ma, T. Xu, S. Han, and K. Kim, "Ensemble learning of multiple deep cnns using accuracy-based weighted voting for asl recognition", *Applied Sciences*, vol. 12, no. 22, p. 11 766, 2022, Submission received: 18 October 2022 / Revised: 16 November 2022 / Accepted: 17 November 2022 / Published: 19 November 2022. (This article belongs to the Special Issue Advances in Applied Signal and Image Processing Technology), ISSN: 2076-3417. DOI: 10.3390/app122211766.

[21] Z. Chen, X. Zhang, L. Wang, and Y. Xia, "A fast path planning method based on rrt star algorithm", in *2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 2023, pp. 258–262. DOI: 10.1109/ICCECE58074.2023.10135365.

[22] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved rrt algorithm in complex environments", *IEEE Access*, vol. 6, pp. 53 296–53 306, 2018. DOI: 10.1109/ACCESS.2018.2871222.

[23] Y. Ying, Z. Li, G. Ruihong, H. Yisa, T. Haiyan, and M. Junxi, "Path planning of mobile robot based on improved rrt algorithm", in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 4741–4746. DOI: 10.1109/CAC48633.2019.8996415.

[24] C. J. Hong and V. R. Aparow, "System configuration of human-in-the-loop simulation for level 3 autonomous vehicle using ipg carmaker", in *2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, Bandung, Indonesia, 2021, pp. 215–221. DOI: 10.1109/IoTaIS53735.2021.9628587.

[25] C. Min *et al.*, "Orfd: A dataset and benchmark for off-road freespace detection", in *2022 International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, 2022, pp. 2532–2538. DOI: 10.1109/ICRA46639.2022.9812139.

[26] N. Ahn, A. Höfer, M. Herrmann, *et al.*, "Real-time simulation of physical multi-sensor setups", *ATZ Electron Worldw*, vol. 15, pp. 8–11, 2020-06. DOI: 10.1007/s38314-020-0207-1. [Online]. Available: https://doi.org/10.1007/s38314-020-0207-1.

[27] Y. Guo, G. Nie, W. Gao, and M. Liao, "2d semantic segmentation: Recent developments and future directions", *Future Internet*, vol. 15, no. 6, p. 205, 2023. DOI: 10.3390/fi15060205. [Online]. Available: https://doi.org/10.3390/fi15060205.

[28] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud", *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1887–1893, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:206853127.

[29] R. Nayak, "Focal loss: A better alternative for cross-entropy", *Towards Data Science*, 2022-04. [Online]. Available: https://towardsdatascience.com/focal-loss-a-better-alternative-for-cross-entropy-1d073d92d075.

[30] H. Mees. “Why some cars have a bigger turning radius than others”. (2022), [Online]. Available: https://www.theautopian.com/the-engineering-behind-why-some-cars-can-turn-tighter-than-others/ (visited on 2022-06-01).

A

Appendix 1

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY