

IoT-BASED MULTI-SENSOR FUSION FOR ADAPTIVE COOPERATIVE ASSISTIVE MOBILITY DEVICES

Oladele Daniel Ayo

A thesis submitted in fulfilment of the requirements for the degree

DOCTOR OF ENGINEERING IN ELECTRICAL ENGINEERING

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment, and Information Technology

at the

Central University of Technology, Free State

Supervisor: Prof. E.D. Markus

Co-supervisor: Prof. Adnan Abu-Mahfouz

Bloemfontein

2023

DECLARATION

I, OLADELE DANIEL AYO, student number #####, do hereby declare that this research project, which has been submitted to the Central University of Technology Free State, for the degree: Doctor of Engineering in Electrical Engineering, is my independent work and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State. This project has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.

OLADELE DANIEL, AYO

Date: 15 September 2023

DEDICATION

This thesis is dedicated to the memory of my late father, Dr. Oladele, who provided unwavering support and encouragement, inspiring me to always aim for the highest goals. I would also like to express my deep gratitude to my family for their love, support, and invaluable advice throughout this journey. Lastly, I extend my heartfelt appreciation to my wife, Christine Oladele, for her selflessness, love, constant encouragement, and unwavering desire for me to achieve a Ph.D.

ACKNOWLEDGEMENT

I am highly grateful to God for his unmerited favour, protection, and guidance in bringing me to this point in my life. I am grateful to Professor Elisha D. Markus, my supervisor for being incredibly patient and insightful throughout this research. I would also like to express my profound appreciation to my co-supervisor, Professor Adnan Abu-Mahfouz for his advice before the start and end of my thesis.

I would like to express my humble appreciation to my sponsors, the Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa, through the Smart Networks collaboration initiative and Internet of Things–Factory Program funded by the Department of Science and Innovation, South Africa.

To my lovely wife, I appreciate your understanding and support throughout my time studying to achieve this dream.

I would be doing an unpardonable injustice if I did not mention the following: my illustrious colleagues and staff at the Electrical, Electronic and Computer Engineering Department for their support and advice.

My final and most heartfelt thanks go to all the authors of the various works of literature that I referenced in this project work; they are the primary reason for this work's success.

ABSTRACT

The increasing prevalence of mobility impairments among the global population has underscored the need for innovative assistive technologies to enhance the lives of individuals with disabilities. IoT-based Adaptive Cooperative Assistive Mobility Devices (IoT-ACAMDs) represent a transformative solution, integrating IoT capabilities to empower users with tailored support, promoting safety, and fostering independence. However, the integration of multiple sensors within ACAMDs poses challenges related to accuracy, reliability, and performance. This study addresses these challenges through the development of an IoT-based multi-sensor fusion architecture that combines LiDAR and camera data to create a unified bird's-eye view (BEV) mapping of the environment.

The study introduces the FastSeg3D algorithm, a novel approach for real-time segmentation of LiDAR 3D points, including the segmentation of drivable points in complex terrains. Utilising the FastSeg3D algorithm, deep learning techniques, Inverse Perspective Mapping (IPM), and Bayesian occupancy grid techniques, the study integrates LiDAR data with information from six monocular cameras, captured simultaneously from different angles. The fusion process results in a BEV mapping of the environment which can be used for simultaneous localisation and mapping (SLAM) of a mobile device (robot in which the sensors are mounted).

The study outcomes demonstrate the effectiveness of FastSeg3D achieving real-time ground and non-ground point segmentation across varying terrain complexities with an impressive average speed of approximately 25 frames per second (fps) on a single core of a 7th-generation Intel i5 processor. Moreover, the multi-sensor fusion architecture achieves an inferencing fusion speed of 17.8 fps without augmentation and 11.3 fps with augmentation on a Ryzen 9 5950 CPU, maintaining an acceptable level of accuracy. The FastSeg3D algorithm also outperforms existing methods in terms of recall and intersection over union (IoU) metrics, achieving the highest recall of ground points (0.9758-0.998) and non-ground points (0.8975-0.9582) in 5 out of 10 scenarios, and the second highest recall

of non-ground points in the remaining 5 scenarios. It also achieves the highest mean IoU (0.8711-0.944) in 9 out of 10 scenarios, and the second highest mean IoU in the 10th scenario with a difference of only 0.221.

This study contributes a comprehensive literature review on sensor data preprocessing, automatic extrinsic calibration, and multi-sensor fusion techniques. A novel ground removal algorithm, an innovative multi-sensor fusion architecture, a view transformation algorithm, and a tailored IoT framework designed specifically for ACAMDs were developed during the study. This study also addresses the urgent demand for adaptive and robust navigation assistive devices for individuals with mobility impairments, representing a significant advancement in the field of multi-sensor fusion for assistive mobility devices.

ABBREVIATIONS

3D/2D	Three/Two Dimension
ACAMD	Adaptive Cooperative Assistive Mobility Device
ADLs	Activities of daily living
BEV	Bird's Eye View
BiFPN	Bidirectional Feature Pyramid Network
DOF	Degree of Freedom
DL	Deep Learning
DLT	Direct Linear Transform
EffDet/EffNet	Efficient Detection/Efficient Network
EKF/UKF	Extended Kalman Filter / Unscented Kalman Filter
FastSeg3D	Fast and Adaptive Ground Segmentation Algorithm of 3D Point Cloud
GPS	Global Positioning System
IMU	Inertial Measurement Unit
IoT	Internet of Things
IoMT	Internet of Medical Things
IoVT	Internet of Video Things
LiDAR	Light Detection and Ranging
LSTM	Long Short-term Memory
LSS	Lift Splat Shoot
MCM	Monte Carlo model
MCC/CC	Mobile cloud computing/ Cloudlet computing
ML	Machine Learning
MBConv	Mobile Inverted Bottle Neck Convolution
MRF/CRF	Markov Random Field/Conditional Random Field
NLP	Natural Language Processing
OpenCV/CV	Open Computer Vision/ Computer Vision
PCA	Principal Component Analysis

ROS/OS	Robotic Operating System/ Operating System
RNN	Recurrent Neural Network
RANSAC	Random sample consensus
REE	Range Elevation Estimation
RPM	Rotation per minute
SGD	Stochastic Gradient Descent
SLAM	Simultaneous Localization and Mapping
SVM	Support Vector Machine
USI	User system interface
WBMF	Window-Based Model Fitting

CONTENTS PAGE

DECLARATION	ii
DEDICATION	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
ABBREVIATIONS	vii
CONTENTS PAGE	ix
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF CODE SNIPPETS	xv
Chapter 1: INTRODUCTION	1
1.1 BACKGROUND	1
1.2 PROBLEM STATEMENT	5
1.3 THE RESEARCH AIMS AND OBJECTIVES	5
1.4 METHODOLOGY AND RESEARCH DESIGN	6
1.5 CONTRIBUTION TO KNOWLEDGE	7
1.6 RESEARCH QUESTIONS	8
1.7 PUBLICATIONS DURING THE STUDY	9
1.7.1 Journal paper(s)	9
1.7.2 Patent(s)	10
1.7.3 Conference paper(s)	10
1.8 ORGANISATION OF THE THESIS	11
Chapter 2: LITERATURE REVIEW	12
2.1 INTRODUCTION	12
2.2 SENSOR TYPES AND CHARACTERISTICS	12
2.3 SENSOR DATA PREPROCESSING	14
2.3.1 Point cloud preprocessing	15
2.3.2 Image preprocessing	20

2.4 AUTOMATIC EXTRINSIC CALIBRATION	24
2.5 MULTI-SENSOR FUSION TECHNIQUES	32
2.5.1 Probabilistic-based techniques	34
2.5.2 Learning-based techniques.....	41
2.5.3 Unified BEV for multi-sensor fusion techniques	45
2.5.4 IoT protocols and data management techniques for ACAMDs	50
2.6 CONCLUSION.....	51
Chapter 3: LIDAR POINT CLOUDS	53
3.1 INTRODUCTION	53
3.2 LIDAR POINT CLOUD	53
3.3 FASTSEG3D	56
3.3.1 Problem definition	57
3.3.2 Organising point clouds: Point cloud representation.....	58
3.3.3 Algorithm 1: Range elevation estimation (REE)	62
3.3.4 Algorithm 2: Window-based model fitting (WBMF).....	63
3.4 CONCLUSION.....	67
Chapter 4: THE CAMERA MODEL AND AUTOMATIC EXTRINSIC CALIBRATION	68
4.1 INTRODUCTION	68
4.2 INTRINSIC AND EXTRINSIC PARAMETERS.....	68
4.2.1 Image sensors	69
4.2.1.1 Camera modelling	72
4.2.1.2 Direct Linear Transform (DLT).....	78
4.3 AUTOMATIC EXTRINSIC CALIBRATION	80
4.3.1 Remove ground points	80
4.3.2 Transform object points to the image camera frame and projection to 2D BEV	81
4.3.3 Project 3D object points in the camera frame to BEV.....	82
4.3.4 Perform image processing to make model translation and rotation invariant...	84
4.4 CONCLUSION.....	88
Chapter 5: BEVMSF3D and IoT FRAMEWORK.....	89

5.1 INTRODUCTION	89
5.2 MULTI-SENSOR FUSION ARCHITECTURE	91
5.2.1 Sensor data preprocessing	91
5.2.2 Feature extraction: The backbone module	91
5.2.3 Inverse perspective mapping (IPM) and the view transform module	96
5.2.3.1 Resample features	97
5.2.3.2 View transform module	99
5.2.3.3 Applying the view transform to all feature levels	102
5.2.4 Feature fusion	103
5.2.5 Classifier module and loss function	103
5.3 THE PROPOSED IOT-FRAMEWORK.....	106
5.3.1 IoT-framework	106
5.3.1.1 Data latency: Fog computing and the MQTT application protocol	108
5.3.1.2 Security: Authentication management system.....	110
5.4 CONCLUSION.....	110
Chapter 6: RESULTS AND DISCUSSION	111
6.1 INTRODUCTION	111
6.2 FASTSEG3D	111
6.2.1 Experimental setup:	113
6.2.2 Experimental parameters for FASTSEG3D	113
6.2.3 Evaluation metrics:	113
6.2.4 FASTSEG3D: REE and WBMF modular performance increase:	115
6.2.5 FASTSEG3D versus other proposed algorithms:	116
6.2.6 Discussion:	118
6.3 BEVMSF3D PIPELINE:.....	121
6.3.1 Experimental setup	121
6.3.2 Experimental parameters	121
6.3.3 Evaluation metrics	124
6.3.4 Discussion	126
6.4 CONCLUSION.....	127

Chapter 7: CONCLUSION.....	129
7.1 CONCLUSION.....	129
7.2 LIMITATIONS AND FUTURE WORKS.....	130
REFERENCES	132
APPENDIX.....	162

LIST OF TABLES

Table 2-1 Sensor types and characteristics	14
Table 2-2 Summary of extrinsic calibration techniques.....	31
Table 3-1 Algorithm 1: Coarse segmentation of point cloud using REE technique	62
Table 3-2 Window-based model fitting (WBMF) algorithm for fine-grain segmentation	66
Table 4-1 Image preprocessing algorithm for robust rotation and translation invariant sensor fusion network model	85
Table 5-1 Resample features algorithm.....	98
Table 5-2 Algorithm of view transform features	100
Table 5-3 Algorithm to apply view transform to all feature levels (also called Pyramid features)	102
Table 5-4 Algorithm for Bayesian classifier with dropout sampling	104
Table 5-5 High-level conversion from BEV mappings to JSON	109
Table 6-1 Description of selected NuScene dataset for evaluating segmentation methods	114
Table 6-2 The performance-to-speed result for each FASTSEG3D module	116
Table 6-3 The performance-to-speed comparison of FASTSEG3D with other algorithms.	117
Table 6-4 Test and validation results comparison on the NuScene dataset.....	125
Table 6-5 Comparison of mAP test results across 10 classes	126

LIST OF FIGURES

Figure 2-1 Image augmentation techniques	22
Figure 2-2 Projecting point clouds onto an image	44
Figure 2-3 Architecture for Multi-image Fusion to BEV [212]	50
Figure 3-1 LiDAR point cloud with depth information	56
Figure 3-2 Developed FastSeg3D block diagram	57
Figure 3-3 Point Cloud Representation	58
Figure 3-4 Flowchart of the FastSeg3D Algorithm	59
Figure 3-5 The window-based model fitting technique	64
Figure 4-1 Checkerboard Images with different possible kinds of lens distortions [253]	71
Figure 4-2 Pattern-based image, undistorted [153].....	72
Figure 4-3 Camera modelling	74
Figure 4-4 Point cloud of object points only after FastSeg3D	82
Figure 4-5 3D Point cloud of object points in camera frame projected to 2D BEV image	84
Figure 4-6 Sample Image Augmentation.....	87
Figure 5-1 The sensor Fusion Pipeline	90
Figure 5-2 A Single EfficientNet Mobile Inverted Bottle Neck Convolution (MBConv) and its improvement for EfficientNet version 2 [278].....	92
Figure 5-3 BiFPN and the FPN family [279]	93
Figure 5-4 Features of a single camera image	95
Figure 5-5 Traditional IPM from Front to Top-down View	97
Figure 5-6 Relationship between the image plane and the BEV plane	99
Figure 5-7 IoT Framework for multi-sensor fusion for ACAMDs	107
Figure 6-1 FASTSEG3D: Before and after the WBMF	111
Figure 6-2 Qualitative comparison between FASTSEG3D before WBMF (REE), after WBMF, and some previous works [87,297,298].....	112
Figure 6-3 Comparison of FASTSEG3D against Shen <i>et al.</i> [298] and Lee <i>et al.</i> [297] proposed algorithms for ground removal	120
Figure 6-4 Ground truth (in blue) versus the BEVMSF3D (in red). NuScene Sample ..	124

LIST OF CODE SNIPPETS

Code 1 Range elevation estimation (REE) snippet	162
Code 2 Window-based model fitting (WBMF) snippet	163
Code 3 Project 3D point cloud to BEV image	164
Code 4 Image augmentation	164
Code 5 Resampler	165
Code 6 View transform module	165
Code 7 Camera-LiDAR feature fusion Snippet	166

CHAPTER 1: INTRODUCTION

Assistive mobility holds significant importance in improving the quality of life for individuals with mobility impairments. The emergence of the Internet of Things (IoT) and multisensor fusion technologies has opened new possibilities in various domains like healthcare, offering the potential to enhance the assistive mobility experience. By seamlessly integrating multiple sensors and cooperative systems within the IoTs, more reliable assistance can be developed for individuals in need of assistive mobility [1–3]. This chapter explains the research process, including background to the study, problem statement formulation, research objective, methodological approaches, contributions to knowledge, research hypothesis, publications, and a comprehensive organisational outline of the thesis.

1.1 BACKGROUND

The rapid advancement of technology in recent years has paved the way for innovative solutions in various fields, particularly in the realm of assistive mobility devices. As the world's population ages and the number of individuals with mobility impairments increases, there has been a growing need for efficient and effective assistive technologies to enhance the quality of life for people with disabilities. Adaptive cooperative assistive mobility devices (ACAMD) are designed to empower individuals with disabilities or injuries to perform activities of daily living (ADLs), such as mobility, self-care, communication, and environmental interactions, with greater ease and confidence. These transformative devices include smart wheelchairs, smart walkers, cutting-edge prosthetic limbs, and advanced hearing aids, among many others. The development of ACAMD has been driven by the need to improve the confidence of the acquired information which includes the interaction between the user, the environment, and the device. It is also driven by the need to follow design specifications, as well as enable customisation to cater for the unique requirements of each user to ensure a personalised experience. The 'cooperative' ability of the ACAMD ensures it is not just efficient as a standalone device, but also its ability to integrate with other technologies, forming a symbiotic relationship to provide

comprehensive support for individuals with disabilities. This cooperative approach fosters a more holistic and inclusive solution, promoting the overall well-being and quality of life of the users. The advent of the Internet of Things (IoT) has enabled this cooperative aspect of the ACAMD to become increasingly successful [1,4–6].

The Internet of Things (IoT) is a network of interconnected devices and sensors capable of collecting, transmitting, and exchanging data without human intervention. Its applicability in the healthcare sector has been recognised for remote monitoring, timely interventions, personalised care, and cooperative assistance by integrating with other smart devices. In the context of assistive mobility, IoT can be utilised to create smart assistive devices, track patient movements, analyse data for improved performance and provide valuable insights to caregivers and healthcare professionals. This can also be referred to as the Internet of Medical Things (IoMT). To achieve this, various sensors such as accelerometers, gyroscopes, cameras, GPS, heart rate monitors, and pressure sensors are integrated into assistive devices like smart walkers, smart wheelchairs, wearable sensors, smart canes and way finders to detect movements, assess the user's environment, identify potential hazards, and create a comprehensive picture of the user's surroundings and needs. The data collected from these sensors can be processed and analysed by IoT systems to offer real-time feedback and adapt assistance according to the user's needs. For instance, IoT can allow a smart wheelchair to sense its environment, avoid obstacles, navigate autonomously, adjust its speed and posture, collect and analyse sensor data, and alert caregivers in case of emergencies [1,4,7,8].

These sensors, with unique functions, play a crucial role in developing IoT-based ACAMDs (IoT-ACAMDs) that meet the users' specific needs [1,5,7–9]. Although these sensors offer significant benefits for IoT-ACAMDs, the ability to accurately capture precise information and make sense of this information remains an active area of research. The use of multiple sensors to measure a phenomenon, such as perceiving a user's environment, presents a valuable approach to addressing the limitations of individual sensors. By combining data from several sensors, multisensor fusion can enhance reliability,

robustness, resolution, precision, and other desirable properties, while simultaneously reducing uncertainty and ambiguity. In the context of IoT-ACAMDs, multisensor fusion plays a crucial role in providing a comprehensive understanding of the user's movements and environmental conditions. This leads to more precise and context-aware assistance, resulting in improved safety and adaptability in various scenarios. For example, the integration of data from GPS, IMU, cameras, Radio Detection and Ranging (RaDAR), ultrasonic sensors and LiDAR can significantly enhance the overall perception which includes the localisation, navigation and object detection ability of IoT-ACAMDs [10]. Similarly, combining data from heart rate monitors, pressure sensors, cameras, and microphones allows for comprehensive monitoring of the user's physiological condition and behaviour across different modalities [11].

Generally, the IoT architecture for the IoT-enabled multisensor fusion for ACAMDs consists of a perception layer, network layer and application layer. To achieve IoT-enabled multisensor fusion for ACAMDs, several factors are considered, depending on the specific ACAMDs under consideration. These factors include the user system interface (USI), perception techniques that occur in the perception layer of the IoT architecture, communication and sensing frameworks performed in the network layer of the IoT architecture, the degree of shared control, the ability to autonomously navigate in unknown environments, fall prediction and prevention, and personalised activity identification [12]. The communication and sensing framework is required to communicate and share information in real-time with other devices, if and when needed. Perception techniques are essential for accurately perceiving the user's environment and movements. These techniques need to be memory-efficient, easily deployable and compact for mobile devices, capable of real-time perception, and scalable. With accurate and real-time perception, successful autonomous navigation in unknown environments becomes more achievable, as the focus can be directed towards localisation and path planning. Consequently, perception techniques are one of the most critical factors determining the success of IoT-enabled multisensor fusion for ACAMDs [1,10].

Perception techniques refer to the methods and processes used by a system or device to sense and interpret information from the surrounding environment. In the context of ACAMDs, perception techniques refer to the methods and strategies used to gather and interpret sensory information from multiple sensors to enhance the understanding and awareness of the environment. These techniques play a crucial role in improving the functionality and adaptability of assistive mobility devices when performing actions like autonomous navigation or user monitoring. Perception techniques like multisensor fusion pipeline can be applied to the camera, LiDAR sensor, GPS, and IMUs to capture and combine information for recognizing objects, detecting patterns, and interpreting scenes, using algorithms like extended Kalman filter (EKF), particle filters (PF), reinforcement learning (RL), or deep learning (DL) [13]. The output of this pipeline returns holistic information about the user's environment, which can then be used by SLAM and path-planning algorithms for autonomous navigation.

IoT-enabled multisensor fusion for Adaptive Cooperative Assistive Mobility Devices is a complex process and still an emerging area of research. For instance, to effectively achieve a communication and sensing framework, the network layer (protocols) of the IoT architecture is required to address a myriad of challenges. These challenges include ensuring data privacy and security to protect sensitive user information, establishing reliable data transmission for seamless communication between devices, promoting interoperability to enable devices from different manufacturers to work together harmoniously, ensuring scalability to accommodate a growing number of connected devices, and focusing on accessibility and usability to ensure that the technology is inclusive and easy to use for all users [3,14,15].

Additionally, the bird's eye view (BEV) multi-sensor fusion pipeline (BEVMSF3D), which is a critical aspect of this study, encompasses the development of efficient and accurate algorithms for data fusion, context awareness, and decision-making. This BEV multi-sensor fusion pipeline (BEVMSF3D) has certain prerequisites like proper preprocessing and filtering of sensor information, as well as ensuring well-calibrated and synchronized

sensors for accurate data fusion [16,17]. Moreover, BEVMSF3D needs to have an acceptable computational complexity, and scalability, and strike a balance between precision and efficiency to enable real-time multi-sensor fusion for practical deployment. For this reason, there are still existing challenges and research opportunities in the development and deployment of such a pipeline/architecture for IoT-ACAMDs.

1.2 PROBLEM STATEMENT

The motivation for this research is to address the research problem of multi-sensor fusion data for autonomous navigation of assistive mobility devices. This problem is challenging due to the diversity, imperfection, and complexity of sensor data, which require effective preprocessing and integration methods. This problem is also relevant for individuals with severe physical disabilities, who constitute a significant portion of the global population and face difficulties in accessing suitable mobility options [10,18]. While traditional assistive mobility devices provide some degree of mobility for individuals with physical impairments, they are not suitable for those with combined physical, cognitive, or perceptual impairments, such as users with spinal cord injuries (SCI) or cerebral palsy. These users need devices that can adapt to their needs and preferences, and navigate autonomously with minimal user input, especially in unfamiliar and/or complex environments [1,10,19]. Addressing these limitations and providing a comprehensive solution that enables adaptive and context-aware assistance is therefore needed.

1.3 THE RESEARCH AIMS AND OBJECTIVES

This research aims to develop a novel sensor fusion architecture for the adaptation of mobility assistance for the user and securely manage this information in the cloud, using an IoT framework. The research aim was achieved by addressing the following objectives.

1. **Recommend an efficient multi-sensor calibration method:** The first objective of this research is to evaluate various automatic extrinsic calibration strategies for multiple sensors and recommend a method that ensures efficient and accurate calibration for navigation purposes.

2. **Develop a novel algorithm for Point cloud processing for multi-sensor fusion, localisation and behaviour monitoring:** The second objective is to develop a ground removal algorithm capable of processing 3D point clouds in real-time for BEVMSF3D, and the IoT framework for localising users and monitoring their behaviour while operating assistive mobility devices.
3. **Develop an architecture for multi-sensor data fusion:** The Third objective is to develop an architecture that effectively fuses data from multiple sensors to facilitate seamless navigation in complex environments.
4. **Recommend a scalable and efficient IoT framework:** The fourth objective is to investigate and propose a scalable and efficient Internet of Things (IoT) framework for managing data transfer between ACAMD devices and the cloud.

1.4 METHODOLOGY AND RESEARCH DESIGN

The methodology used is as follows:

1. **A comprehensive literature review:** A literature review of existing solutions for IoT-enabled multi-sensor fusion for ACAMDs was discussed and the evaluation of existing extrinsic calibration strategies was highlighted.
2. **FASTSEG3D:** The development of a novel, fast and adaptive algorithm for ground segmentation of 3D point cloud data. This was developed using the ROS 2 platform, the C++ point cloud library (PCL), and the C++ OpenCV library.
3. **Sensor data preprocessing and calibration:** This involved:
 - a. Acquisition of required sensor data. The nuScenes dataset and KITTI dataset were considered for the development of the BEV multi-sensor fusion pipeline/architecture.
 - b. Recommendation of an image preprocessing technique like image normalisation and image augmentation for BEVMSF3D. This was achieved using the PyTorch platform, and a set of Python libraries (Albumentation, matplotlib, numpy, and OpenCV).

- c. Recommendation of an extrinsic calibration strategy for BEVMSF3D. An extrinsic calibration method was recommended for the automatic calibration of images and LiDAR (3D point cloud data).
4. **Development of the sensor fusion architecture:** After the relevant preprocessing and calibration steps were considered, a multi-sensor fusion architecture was developed using the PyTorch platform and its necessary libraries for BEVMSF3D.
5. **Recommendation of an IoT framework for ACAMDs:** Finally, an IoT framework was proposed, specifically an IoT protocol and an algorithm for data reduction was discussed to address low latency and scalability challenges common to IoT-enabled ACAMDs.

1.5 CONTRIBUTION TO KNOWLEDGE

This section outlines the main contributions of this study to the field of IoT-enabled multi-sensor fusion for ACAMDs. The primary objectives of this thesis were elucidated in section 1.3, and throughout this study, several significant contributions have been realised. The contributions are:

1. **In-depth literature review:** A comprehensive literature review on the existing methods and techniques for sensor data preprocessing, automatic extrinsic calibration, and multi-sensor fusion in the context of ACAMDs, identifying the gaps and challenges in the current state-of-the-art.
2. **Innovative ground removal algorithm:** A novel, fast, and adaptable ground removal/segmentation algorithm (FASTSEG3D) that achieves superior performance in terms of accuracy and speed compared to the state-of-the-art methods for 3D point cloud segmentation, achieving a minimum recall of ground points of 0.97, 0.89 recall for non-ground points and a minimum IoU of 0.87 on the benchmark dataset, and an average speed of approximately 25 fps on a single core of a 7th-generation Intel i5 processor. This algorithm achieves above 1.31% improvement in ground recall when compared to similar and recent algorithms (See Chapter 6).

3. **Advanced multi-sensor fusion architecture:** A novel multi-sensor fusion architecture with automatic calibration that integrates data from image sensors and point cloud sensors, resulting in comprehensive and insightful environmental information that enhances the localisation and monitoring capabilities of ACAMDs, achieving an inferencing fusion speed of 17.8 fps without augmentation and 11.3 fps with augmentation on a Ryzen 9 5950 CPU, and incorporating a novel automatic extrinsic calibration method that does not require any manual intervention or calibration objects.
4. **Innovative view transformation algorithm:** A novel algorithm for view transformation that transforms the fused information from various sensors into a unified bird's-eye view (BEV) mapping of the environment, which can be used for simultaneous localisation and mapping (SLAM) of a mobile device, and preserves the semantic information of the fused data, utilising Inverse Perspective Mapping (IPM) and Bayesian occupancy grid techniques to generate a high-resolution BEV mapping of the environment, and preserving the object labels and the drivability status of the fused data.
5. **IoT-framework recommendation:** A recommended IoT framework tailored specifically for ACAMDs has been proposed. This framework proposes a protocol and an algorithm for data reduction to integrate Internet of Things (IoT) technologies into the realm of ACAMDs, thereby reducing data latency, and enhancing their connectivity and overall performance.

1.6 RESEARCH QUESTIONS

How can an IoT-based multi-sensor fusion architecture be developed and utilised to enhance adaptive cooperative assistive mobility devices (ACAMDs) for individuals with combined physical, cognitive, or perceptual impairments, such as spinal cord injuries (SCI) or cerebral palsy?

To answer this question, the following sub-questions are formulated:

1. What are the existing methods and techniques for sensor data preprocessing, automatic extrinsic calibration, and multi-sensor fusion in the context of ACAMDs, and what are the limitations and challenges that need to be addressed?
2. How can a novel, fast, and adaptable ground removal/segmentation algorithm (FASTSEG3D) be developed and applied to 3D point cloud data to enable efficient and accurate segmentation of ground and non-ground points in complex terrains?
3. How can a novel multi-sensor fusion architecture with automatic calibration be designed and implemented to integrate data from image sensors and point cloud sensors, resulting in comprehensive and insightful environmental information that enhances the localisation and monitoring capabilities of ACAMDs?
4. How can a novel algorithm for view transformation be proposed and utilised to transform the fused information from various sensors into a unified bird's-eye view (BEV) mapping of the environment, which can be used for simultaneous localisation and mapping (SLAM) of a mobile device?
5. How can a recommended IoT framework tailored specifically for ACAMDs be proposed and evaluated to integrate Internet of Things (IoT) technologies into the realm of ACAMDs, thereby reducing data latency, and enhancing their connectivity and overall performance, while ensuring data privacy and security?

1.7 PUBLICATIONS DURING THE STUDY

1.7.1 Journal paper(s)

Oladele DA, Markus ED, Abu-Mahfouz AM. Adaptability of Assistive Mobility Devices and the Role of the Internet of Medical Things: Comprehensive Review. JMIR Rehabil Assist Technol 2021;8(4):e29610. doi: 10.2196/29610. PMID: 34779786. (Published)

Oladele DA, Markus ED, Abu-Mahfouz AM. Enhancing Security in IoT-Based Smart Wheelchair Networks with an Intrusion Detection System for smart cities. MDPI-IoT (Submitted)

Oladele DA, Markus ED, Abu-Mahfouz AM. An Efficient Feature Selection Technique for An Intrusion Detection Framework for IoT-Devices. MDPI-IoT (Submitted)

Oladele DA, Markus ED, Abu-Mahfouz AM. BEV-CAM3D: A Unified Bird's Eye View Architecture (BEV) for multiple monocular cameras (CAM) and three-dimensional (3D) point clouds. IEEE Transactions on Neural Networks and Learning Systems (Submitted)

1.7.2 Patent(s)

Oladele DA, Markus ED, Abu-Mahfouz AM. (2023). FastSeg3D: A fast and adaptive Ground Segmentation Method of 3D Point clouds using Range Elevation Estimation (REE) and window-based model fitting (WBMF) of the 3D point clouds. Reference No.: P96434ZP00 VIP/jbg (Submitted to the South African Patent Office)

1.7.3 Conference paper(s)

O. Ayo Daniel, E. Didam Markus and A. M. Abu-Mahfouz, "Internet of Things Based Multi-Sensor Fusion for Assistive Mobility Devices," 2021 Conference on Information Communications Technology and Society (ICTAS), 2021, pp. 115-120, doi: 10.1109/ICTAS50802.2021.9395010.

Daniel A. Oladele, Elisha D. Markus, Adnan M. Abu-Mahfouz, Towards an adaptable framework for mobility assistive technologies, Procedia Computer Science, Volume 198, 2022, Pages 301-306, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2021.12.244>

1.8 ORGANISATION OF THE THESIS

This thesis is organised as follows:

- Chapter two discusses an in-depth literature review of the foundational components crucial for the successful implementation of multi-sensor fusion in the context of ACAMDs.
- Chapter three presents the point cloud preprocessing technique used for this study, it further presents the novel FastSeg3D algorithm used for fast segmentation of the point cloud, a vital component of the subsequent BEV multi-sensor fusion pipeline (BEVMSF3D).
- Chapter four examines the image preprocessing techniques employed by this study for BEVMSF3D. It further highlights the automatic calibration technique used for the automatic alignment of point clouds and image features for the sensor fusion pipeline.
- Chapter five presents BEVMSF3D based on all the previously elaborated techniques and the pipeline itself. It also highlights the recommended IoT framework for the multi-sensor fusion of ACAMDs.
- Chapter six presents a discussion of the results.
- Finally, Chapter seven summarises the conclusion and anticipated future works.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

Multi-sensor fusion is a technology that integrates data from multiple sensors to enhance the performance of robotic applications, such as perception, navigation [20,21], and decision-making [22,23]. IoT-enabled multi-sensor fusion extends this technology by accessing data from different locations and devices [24], which can improve the functionality of assistive mobility devices, such as smart wheelchairs. However, IoT-enabled multi-sensor fusion is still an emerging field and faces many challenges, such as scalability, security, latency, complexity, precision, efficiency, and calibration errors [1,14,25–27]. This literature review examines the current research on IoT-based multi-sensor fusion for ACAMDs, covering the topics of sensors, sensor data preprocessing, automatic extrinsic calibration, multi-sensor fusion, IoT frameworks, and localization.

2.2 SENSOR TYPES AND CHARACTERISTICS

In the realm of IoT-based multi-sensor fusion for adaptive cooperative assistive mobility devices, understanding the strengths and limitations of various sensor types is crucial for achieving optimal performance. Sensors serve as the eyes and ears of these mobility devices, capturing the surrounding environment in a manner these devices can understand. However, each sensor type possesses unique characteristics, leading to inherent strengths and limitations that impact their efficacy. The concept of sensor fusion emerges as a strategic approach to harness the advantages of diverse sensors while mitigating their drawbacks. The strengths and drawbacks of these sensors can be attributed to their unique features and properties also known as the sensor's characteristics [1,10].

Dead reckoning sensors that measure the position, velocity, acceleration and orientation have been used for decades for autonomous navigation. However, they are generally considered to be low-precision sensors due to their unreliability as a result of external interferences/factors and accumulated errors over time [28–30]. To improve its accuracy these sensors have been combined. An example is the Inertial Measurement Unit (IMU)

sensor, which is a combination of an accelerometer, gyroscope, and some magnetometers, used to measure and report the specific force, angular rate, and orientation of the device, enabling IoT-ACAMDs to adjust their speed, direction, and posture according to the user's preferences and environmental conditions. However, its accuracy/precision depends on the quality of the sensors, the calibration process, and the error correction methods, and can be affected by external magnetic fields, drift, and accumulated error [29,30].

Another example is the integration of the Global Positioning System (GPS) and the Global Navigation Satellite System (GNSS) with IMUs to provide real-time location data and map the surroundings for improved navigation. However, the accuracy of GPS is reliant on the signal quality received, atmospheric conditions, electromagnetic interference, and the ionosphere. Additionally, GPS signals cannot penetrate solid walls or structures, resulting in reduced signal quality when obstructed by obstacles such as buildings, trees, mountains, or tunnels [31,32]. To further improve accuracy and reliability, optical sensors are often used in combination with dead reckoning sensors, as they provide additional information about the environment [33–35].

Optical sensors use light to measure the physical properties of the environment such as distance, colour, shape, and reflectivity. These sensors are used to greatly improve navigation precision for mobility devices and also to capture a more complete estimate of the 3D map of the environment relative to the robot. Optical sensors like cameras and Light Detection and Ranging (LiDAR) capture visual information, enabling IoT-ACAMDs to recognize obstacles, signage, doorways, and other critical environmental cues [36].

However, cameras have a limited dynamic range, which can cause overexposed or underexposed areas in images. The quality of the image is also dependent on factors such as illumination, high ISO, and compression levels, while noise, dust, or moisture can also affect the image quality. Monocular cameras, which are more affordable, cannot directly measure the depth information of their three-dimensional (3D) environment [37,38]. LiDAR sensors, on the other hand, can suffer from signal loss during adverse weather

conditions like heavy rain and fog, but their main limitation lies in their high cost, making them less accessible for certain applications. The more affordable ones suffer from sparse resolution due to the fewer number of laser beams that make up the cheaper LiDAR sensors [39].

Table 2-1 shows a summarized sensor types and characteristics of sensors commonly used for assistive mobility devices.

Table 2-1 Sensor types and characteristics

Sensors	Categories	Merits	Drawbacks
IMU	Dead reckoning	High accuracy, low latency, independent of external signals	Drift error, high cost, complex calibration
LiDAR	Active optical	High-resolution, long-range, 3D mapping	Sensitive to weather, expensive, heavy
Cameras	Passive optical	Low cost, easy to use, rich information	Limited by lighting, occlusion, and perspective
RADAR	Radio	Robust to weather, long-range, doppler effect	Low resolution, interference, clutter
Position (GPS/GNSS)	Radio	Global coverage, high accuracy, low power	Dependent on satellite signals, affected by multipath and jamming
Odometry	Motion	Simple, low-cost, relative measurement	Cumulative error, wheel slip, limited by terrain
Ultrasonic	Sound	Low cost, easy to use, distance measurement	Low range, affected by noise, reflection

2.3 SENSOR DATA PREPROCESSING

Sensor data preprocessing is an important step in multi-sensor fusion for ACAMDs. It aims to improve the quality, efficiency and reliability of sensor data before the fusion process,

which affects the accuracy and reliability of the fusion results. Sensor data preprocessing also ensures scalability and meaningful insights for decision-making and assistive functionalities in IoT-based multi-sensor fusion [40,41].

The preprocessing steps depend on the type of sensor fusion approach and the type of sensor data [42,43]. For point cloud data, preprocessing steps include structuring, filtering, and segmenting the point clouds, and extracting features for early, intermediate, and late fusion [44–47]. Features from these point clouds can be extracted for intermediate fusion [48,49] or late fusion [43,50]. For image data, preprocessing steps include noise reduction, edge enhancement, image normalisation and augmentation [51–56], and feature extraction for pixel-level, feature-level, and decision-level fusion [57–60].

Although sensor data preprocessing is a very useful step towards multi-sensor fusion, the choice of sensor data preprocessing techniques can either reduce or increase the computational complexity of the overall architecture. This section will discuss some of the preprocessing approaches used for 3D point cloud sensor data and image sensor data for early fusion.

2.3.1 Point cloud preprocessing

Point clouds are collections of 3D points that represent the real 3D world with geometric information and attributes. They are useful for many applications, especially when captured by sensors that are robust to illumination and weather changes, such as LiDAR and RADAR [61]. However, point clouds have some drawbacks, such as being unstructured, can be extremely large, and noisy, and lacking semantic meaning and topology of objects in the environment. Therefore, they need to be preprocessed to become meaningful. Point cloud preprocessing involves structuring, organising, filtering, and removing outliers from point clouds [62,63]. This enables further analysis of point clouds to extract features, such as objects and their locations, and to perform localisation, i.e., determining the location of the robot relative to objects [64–68]. Point cloud preprocessing techniques apply to various

domains. However, the techniques for preprocessing point clouds are the same and applicable to multi-sensor fusion of IoT-enabled ACAMDs.

To achieve a structured format for easy extraction of similar geometric features between point clouds, earlier preprocessing techniques approached point cloud representation using a defined data-structured [69–71] for establishing complex neighbourhood relations and extracting point features. In these techniques, raw point clouds were first sorted using space partitioning data structures like KD-tree [72] or a tree data structure like Octree [73] before a segmentation or clustering algorithm was used to extract point clouds with similar features into subsets. This approach has been used extensively in many studies for structuring point clouds [74–76], however as the points become larger the complexity drastically increases. This is because of the time complexity for sorting point clouds, space complexity or memory overhead for large and rapidly changing point clouds, and the balancing error that could occur while sorting point clouds [75,77,78]. To address this the mapping representation i.e. the elevation or terrain map method was first introduced by researchers at Stanford University [79]. This method represents point clouds in a 2D grid. Each grid cell has one of three possible values; occupied, free, and unknown. Occupied grids were determined based on comparing the vertical heights of two nearby grids with threshold height, and a probabilistic test was used to validate the presence of an obstacle. Although this project won the 2005 DARPA challenge, this representation did not model the free vertical space between a pair of points and therefore was very sensitive to uneven terrain. To address this Himmelsbach *et al.* [80] introduced the efficient partitioning of point clouds by representing the XY-plane as a circle of infinite radius and divided into a discrete number of segments [81]. Points are assigned to segments or bins according to their angular component and range or laser beam component. This approach has become a common technique for point cloud preprocessing in recent years [82–84].

Another very important preprocessing step for point clouds is ground segmentation. Since most of the points in the raw point cloud belong to the ground, an accurate ground segmentation can significantly reduce the computational cost of performing object

detection, or multi-sensor fusion tasks. It is also a very useful step for LiDAR odometry to obtain strong constraints. Some common segmentation techniques applied to achieve ground segmentation include the elevation map method, the plane or ground modelling method, the method based on the relationship between adjacent points, the method based on Markov Random Field (MRF), and the deep learning method [80,85,86].

The ground, plane or modelling method are improvements of the RANSAC plane segmentation [87]. The basic concept is to generate a ground skeleton by adopting regression algorithms. This approach performs best on flat roads with apparent boundaries. To address curved/uneven terrains and increase accuracy, the Gaussian process regression (GPR) is commonly used. This approach [88,89] divides a two-dimensional (2D) ground segmentation in the 3D polar grid representation map into many one-dimensional (1D) GPR problems with a covariance function to lower complexity and improve computational efficiency. However, this method is insensitive to stair steps and since it is divided into independent segments the overall ground elevation continuity is not assured. A more robust regression approach was applied by the authors in [90] to accurately estimate ground heights. It integrated the Gaussian process regression with a robust locally weighted regression (RLWR) to form a hybrid regression model for estimating road surfaces. However, the success of this method is dependent on the accurate selection of pavement seed points. Also, because the original RLWR method is slow, the hybrid regression is more computationally demanding when compared to the GPR-based method.

The MRF and its variants such as the Conditional Random Field (CRF) use gradient cues of the geometry of the road to construct MRF and implement a belief propagation for inferencing to classify the robot's environment into different regions. This approach has been proven to be robust to uneven terrain [91]. Zhang *et al.* [91] proposed the use of MRF for ground segmentation and the Loopy Belief Propagation algorithm for inferencing. Rummelhard *et al.* [85] Proposed a SpatioTemporal Conditional Random Field (STCRF) with previous timestamp information for ground segmentation and the Expectation Maximisation (EM) algorithm for inferencing. Although the MRF and CRF methods have been proven to be quite robust to undulated terrain, they are computationally expensive for

inferencing and approaches similar to the methods reported in [85], which also require additional information from GPS and IMU sensors. This makes its applicability to real-time scenarios less realistic.

Because 3D point clouds are usually unstructured and contain objects of varying sizes with variable point densities, many researchers choose to exploit similar geometry features between points for segmentation, like the distance between points, height difference, angle difference, or shape-based. For instance, Moosmann *et al.* [92] performed ground segmentation using the local convexity criteria between points. A neighbourhood graph construction was used to structure points, and then segmentation was done based on points meeting the neighbouring surfaces criterion, respective to an arbitrary fixed coordinate system and their normals. Yang *et al.* [93] classified shapes using the support vector machine (SVM) algorithm and proposed a shape-based method for the segmentation of point clouds. The major drawback of this approach is that segmentation accuracy is heavily reliant on accurate calculation, and prior knowledge of segmentation criteria (parameters), this makes it a computationally demanding approach and not requiring a lot of handcraft skills to successfully reuse.

To avoid the complexity of full 3D maps, elevation maps were proposed as an attractive alternative. The elevation method was first introduced by researchers at Stanford University [79]. This algorithm separates the ground from the non-ground points based on the relative height difference. Like Zhang *et al.* [94], Zermas *et al.* [95] addressed the uneven terrain problem by dividing the ground surface evenly into segments. The region-growing clustering algorithm was used on the initial seed ground points for the N number of ground surface segments to segment the ground planes that meet a particular distance threshold. Generally, the main advantage of the elevation map method is its inexpensive computation cost. It is not sensor-specific and is effective when used on fused point clouds built from many 3D scans. However, this approach struggles to do well in complex scenes like protruding or overhanging objects. To address these challenges, Goga *et al.* [96] proposed the use of a multi-volume grid elevation map to store the start and end heights of the

consistent vertical region in each given cell to keep track of the ground planes. However, they did not focus on coarse-to-fine ground segmentation and their proposed solution was insensitive to slope terrain.

In recent years many deep learning approaches have been introduced for 3D point cloud segmentation [97,98]. This technique falls under the category of supervised machine learning techniques and is a more advanced technique for extracting abstract features and deducing semantic meanings from 3D point clouds. This approach is considered to be the most influential and fastest-growing technique in pattern recognition, computer vision, and data analysis [99,100]. Because it is a data-driven approach, it is usually computationally expensive for model training and inferencing and requires a very expensive human annotation process to successfully perceive a handful of objects. For instance, the PointNet architectures [97,98,101] proposed a novel model that adheres to permutation invariance, transformation invariance and point interactions. These trends of architecture performed well on the accuracy benchmarking test but not so well for inferencing time having its best inferencing of 4 frames per second (FPS). Besides, the FrustumPointNet which is an improvement over the PointNet/PointNet++ requires pre-processing using the camera sensor based on another network model to handle calibrated camera images [98]. This also increases the inferencing time and is not suitable for LiDAR-only segmentation. Although many variants of these deep learning approaches have been proposed [102–104], in their benchmarking, they all tend to pay emphasis on accuracy and the number of trainable parameters but not inferencing speed. An attempt to improve inferencing speed led to a significant drop in accuracy. Furthermore, deep learning approaches require a time-consuming process of ground truth labelling and performance is potentially degraded when applied to new previously unlearned terrains.

For a ground removal/segmentation technique to be effectively used as a preprocessing step it needs to be sufficiently fast, efficient, and robust to be employed as a preprocessing step. Unlike supervised approaches that involve learned parameters and lots of training, point cloud segmentation algorithms including unsupervised clustering algorithms are

fundamentally the results of geometric constraints and statistical rules [100]. These algorithms generally have a well-defined approach towards decision-making (segmenting regions) and are interpretable and reusable (data efficient). Also, they do not need to be retrained or relabelled to improve accuracy and effectively identify ground points that were not previously included in the training dataset [105]. These attributes make them more computationally efficient than deep learning approaches. Although deep learning approaches are very efficient for applications that do not require real-time usage or have limited memory, point cloud segmentation algorithms are still a very common choice for applications that require real-time usage or fast preprocessing for downstream applications. For instance, for 3D geodata, indoor scenes and autonomous driving scenarios, unsupervised clustering algorithms also referred to as unsupervised segmentation can be used as a preprocessing step for several processes like feature extraction, classification or 3D modelling [106–108].

2.3.2 Image preprocessing

Some common techniques include noise reduction, edge and contrast enhancement, edge detection, image normalisation and augmentation (this includes translation, rotation, flipping, resizing, colour space conversion, scaling and skew). For IoT-enabled multi-sensor fusion, image preprocessing plays a pivotal role in harmonising, standardising and reducing the computational complexity of image data for a robust BEV multi-sensor fusion pipeline [57,58,60].

Pixel-level preprocessing techniques, like the Gaussian filter, calculate the weighted average of a pixel and its neighbours, effectively reducing high-frequency noise while preserving image edges [57]. However, these techniques are computationally intensive and may lead to the loss of some pixel details. Meanwhile, edge detection methods like Canny edge detection, the Prewitt operator, and the Sobel operator utilise multiple kernels to smooth or identify specific types of edges (horizontal or vertical). Nevertheless, this proves unnecessary for sensor fusion preprocessing as these tasks are managed by the pipeline's feature extraction kernel, rendering these techniques unnecessary [57,60,109].

The more common image preprocessing techniques include image normalisation and augmentation. Image normalization also referred to as spatial and intensity transformation [112,113], popularized by the Viola-Jones face detection algorithm [110], adjusts pixel intensity values to establish similarities between images in terms of lighting and contrast effects [111]. Various approaches have been proposed for image intensity normalisation [111,112,114–116], each with different levels of computational complexity. For multi-sensor fusion, the min-max normalisation technique proves sufficient, as it addresses the issue of slow convergence during the training and development of the multi-sensor pipeline or algorithms [117].

Image augmentation goes beyond image normalization, creating new instances to prevent overfitting and enhance the generalization capabilities of trained ML and deep learning (DL) models. Image augmentation can be viewed as a form of data warping [118], originally applied in tasks such as handwritten digit classification using convolutional neural networks (CNNs). Over the years, this technique has evolved and now encompasses a range of methods including geometric transformations and colour space variations [119], kernel filters, mixing images [120], random erasing [121], feature space augmentation [122], and generative adversarial networks (GANs) [123] as illustrated in Figure 2-1. Many of these approaches have been suggested by researchers as valuable tools for improving ML and DL models while mitigating the challenge of overfitting. Some examples includes geometric transformation, colour manipulation and brightness adjustment.

Geometric transformations, often referred to as traditional augmentation [119], encompass operations like flipping, cropping, rotation, translation, scaling, and noise injection. These transformations contribute to extending the training dataset, enhancing the rotation and translation invariant properties of DL models, and addressing challenges like occlusion [124]. Color manipulation and brightness adjustment further diversify image characteristics. However, it's crucial to note that these techniques are not exhaustive, and their selection depends on the specific objectives of the task at hand.

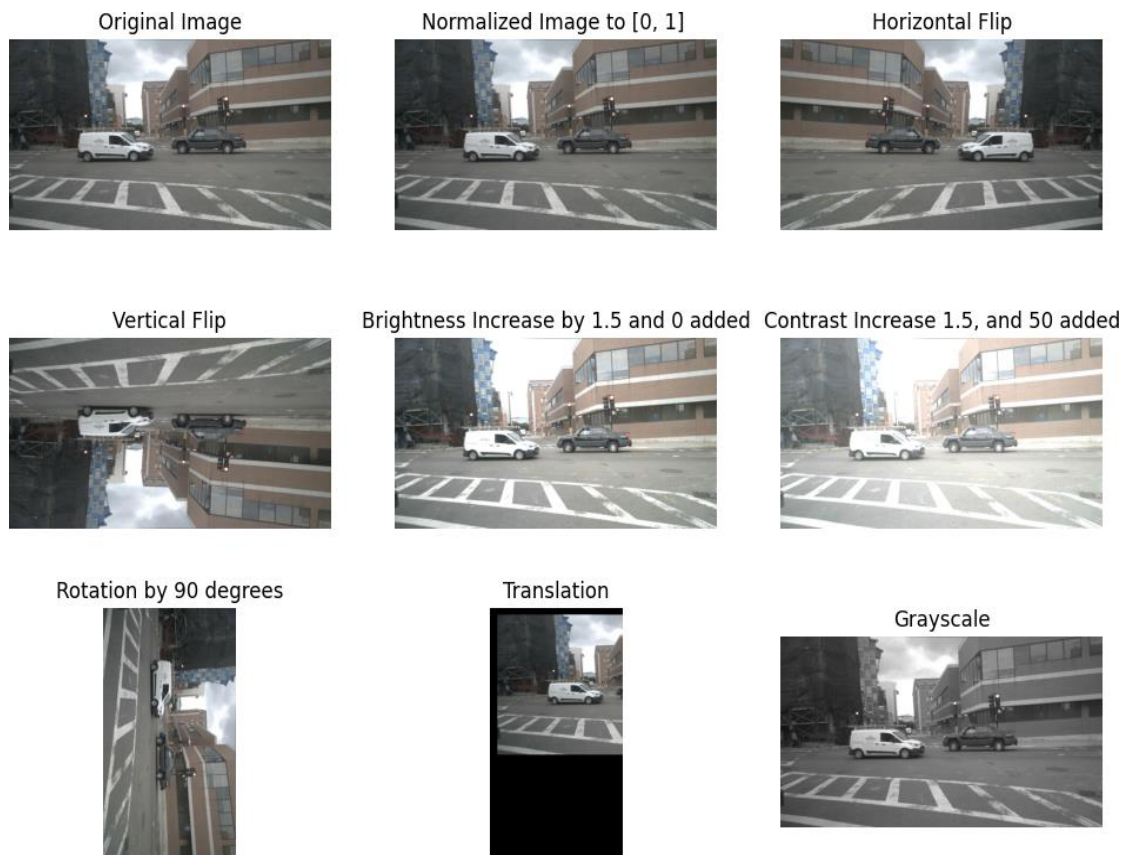


Figure 2-1 Image augmentation techniques

Taylor *et al.* [119] suggested the application of generic data augmentation techniques to substantially diminish the necessity for an extensive collection of labelled training datasets. Various augmentation strategies were explored and their respective performances were compared. The findings indicated that cropping and geometric transformations exhibited the most favourable outcomes. However, the authors acknowledged that their study did not encompass a comprehensive exploration of all potential augmentation methods.

Another valuable and prevalent augmentation strategy involves the random selection and removal of image regions during the training process. As outlined by Zhong *et al.* [125], the authors seek to mitigate the occlusion challenge frequently encountered by deep

learning (DL) models. They achieved this by randomly selecting and erasing specific rectangular pixel areas within an image. This practice results in the creation of new images with instances of occlusion during training. The complementary nature of this approach lead to consistent improvements across image classification, object detection, and person re-identification tasks.

Image mixing stands out as another image augmentation technique, operating within the input or data space, by combining information from two or more images to forge novel compositions. A thorough survey of this technique was undertaken by Naveed *et al.* [126]. Over the years, many methods have been proposed to combine multiple images for generating new instances and enhancing generalisation, for instance, SamplePairing [127]. Notably, a prominent technique in recent times is the CutMix approach [128], which combines the random erasing augmentation technique and the image mixing technique. A new image is generated by cutting out rectangular regions of an existing image and replacing those regions with rectangular regions of another image. This technique was also successful because, unlike the random erase technique, empty regions are replaced by patches of images, allowing for efficient use of training pixels and retaining the regularisation effect of regional dropout. Experimental evaluations were conducted on CIFAR-100, ImageNet, CUB200-2011, Pascal VOC, and MS-COCO datasets, using various CNN architectures and results showed that CutMix consistently outperformed state-of-the-art augmentation strategies in terms of classification accuracy, localisation performance, and transfer learning to object detection and image captioning tasks.

Feature spacing is an additional image augmentation technique commonly employed within the learned space, involving the creation of a fresh dataset through insights gathered during the training phase. DeVries *et al.* [129] proposed a domain-agnostic strategy for dataset augmentation, conducting transformations within a learned feature space rather than the input space. Specifically, these transformations were carried out in the context vector space generated by sequence-to-sequence models. Starting from the existing dataset, the authors applied straightforward transformations, including noise addition, interpolation, or

extrapolation, after identifying the 10 nearest in-class neighbours through a search in the learned feature space. This methodology was exemplified using the Arabic digits dataset. While the proposed augmentation approach exhibited utility, the author's findings indicated that the interpolation technique often led to undesirable overfitting of the model.

Additionally, Li *et al.* [130] proposed a feature space augmentation method named "moment exchange (MoEx)," which harnessed moment information (mean and standard deviation) derived from learned features in recognition models for dataset normalisation and augmentation. The authors accomplished this by substituting the moments of learned features from one training image with those from another image, while simultaneously interpolating the target labels within the feature space. Experiments conducted on various recognition benchmark datasets using diverse models consistently revealed enhancements in the generalisation capacity of the baseline networks. However, similar to all feature space image augmentation techniques [131,132], these methods entail computational complexity. They require preliminary model training to accumulate useful information within the feature space, which is a computationally demanding process, before augmentation can be performed, followed by retraining the model using both the original and augmented data.

Although image augmentation is an important image preprocessing technique, it can be very computationally demanding and there is also a need to be aware of the safety during the augmentation process to avoid losing the original features of the original image dataset [133,134].

2.4 AUTOMATIC EXTRINSIC CALIBRATION

Multi-sensor fusion requires accurate synchronisation of sensors to align their orientation, position, and timing. This reduces the error caused by inaccurate calibration of different sensors in a fusion framework. Calibration is the process of finding intrinsic and extrinsic parameters for each sensor, which affect the quality and consistency of sensory data [135].

Calibration is important for subsequent processing stages, such as sensor fusion, obstacle detection, localisation, mapping, and control systems. Intrinsic calibration determines the internal mapping of a sensor, while extrinsic calibration establishes a common spatial coordinate system for multiple sensors. Sensors usually have factory calibration, but extrinsic calibration is more challenging and crucial for multi-sensor fusion. Various calibration methods have been proposed, such as feature-based, motion-based, and learning-based calibration [136].

Extrinsic calibration among multi-modal sensors requires synchronised timestamps, which can be obtained by hardware or software methods. Hardware synchronisation uses a shared clock, usually based on GPS time, to align timestamps from different sensors, but it may have errors or delays due to varying sampling frequencies. Software synchronisation assigns a timestamp to each sensor measurement, using frequency information from various sources, and corrects the synchronisation errors by relating to the lowest frequency sensor [137–139] or applying a least-squares method [137,140,141].

The most common calibration technique is the featured-based calibration technique. This technique is quite straightforward and uses a direct or indirect target to establish extrinsic calibration for multi-modal sensors. Direct target-based calibration involves using a special control field and a ground truth target to determine the sensor poses relative to the reference frame. This can be done using techniques such as checkerboard (e.g. a chessboard) calibration [142,143], pattern-based calibration [144] and Trihedron target calibration [145]. A series of target-based extrinsic calibration approaches of an image sensor (charged-coupled devices, CCD camera) on a robot was proposed by the authors in [143], which entailed determining the sensor's position and orientation. Three distinct classes of calibration procedures were introduced, labelled as A, B, and C, all relying on closed-form solutions using a checkerboard target. Class A demanded a precalibrated reference object, while Class B leveraged the robot's mobility and necessitated a reference frame. Class C, however, did not require a reference object but relied on a visible point. The Tsai-Lenz [146] and Shiu-Ahmad [147] methods were utilised as existing B-type calibration

techniques and validated against real data. The experimental setup involved mounting a CCD camera on a robot and utilising a 2-D grid template for testing the Class B calibration. Results highlight that Class A's calibration yields biased outcomes without error-free pre-calibration, while Class B and C procedures show successful testing. The Tsai-Lenz method demonstrates the smallest average standard deviations. The authors acknowledged limitations, including the requirement for error-free pre-calibration in Class A and the challenge of an underdetermined nonlinear system in calibration, necessitating the manipulator's displacement at least twice to solve rotation angle ambiguities.

Another checkerboard approach for the calibration of multi-model sensors was proposed for a camera and laser range finder [142]. The authors focused on capturing a planar checkerboard pattern and subsequently establishing correlations between the pattern's views captured by the camera and the laser range finder using a direct solution to minimise algebraic errors arising from these constraints. This is followed by a nonlinear refinement phase, to further mitigate re-projection errors. The experiment encompassed diverse scenarios involving varying numbers of checkerboard poses, different orientations of the checkerboard plane, and fine-tuning of camera intrinsic parameters. The estimated extrinsic parameters were compared to the ground truth values and quantifying errors in camera orientation and position. Results showed enhanced calibration accuracy and reduced variance in estimating intrinsic camera parameters. However, the authors acknowledged certain limitations, notably the omission of factors such as the decline in laser points as angles increase and potential accuracy issues in corner detection due to foreshortening. This is similar to the limitations presented by Kim *et al.* [148] which also proposed a checkerboard calibration approach by projecting the corners of the 2D chessboard image onto a 3D space. The authors acknowledged that the calibration results are unreliable for calibration object distances surpassing 5 meters.

A pattern-based target calibration object approach was proposed by Rodriguez *et al.* [144] for combining a multi-layer lidar and a CCD camera to achieve precise relative sensor pose estimation, incorporating confidence information into the calibration, minimising the

calibration target size, leveraging multi-layer lidar data, and conducting an in-depth analysis of calibration errors and confidence intervals. This was achieved by a circle-based target pose estimation, rigid transformation computation, and error-confidence analysis. To enhance robustness, a maximum likelihood (M) estimator algorithm and the Levenberg-Marquardt (LM) algorithm for weight computation and parameter refinement were employed. Real-world and simulated scenarios results showed the efficacy of the calibration method. Although no limitations were highlighted in the paper, the LM algorithm [149] is a combination of the Gauss-Newton algorithm and the gradient descent algorithm, is an iterative process that adjusts the step size during each iteration to balance between the efficiency of the Gauss-Newton method and the robustness of the gradient descent method. This algorithm is common for having the drawback of being stuck in local minima and being computationally expensive.

Despite the well-established accuracy and reliability of the target-based calibration method, its manual execution can be a laborious task, impeding calibration accuracy. The commonly employed error correction algorithms are iterative and computationally demanding, prompting the exploration of alternative targetless or target-of-opportunity calibration techniques. The targetless method also referred to as indirect target, automatic extrinsic, or targetless-based calibration, exploits existing objects within the captured environment to estimate extrinsic calibration parameters. This method is termed "targetless" due to its independence from specific, manually set calibration targets. Motion-based and learning-based calibrations are also classified under targetless calibration, with each addressing the calibration error problem using a distinct technique. For instance, in 2011, a notable advancement was made by Carrera *et al.* [150] in the domain of multi-camera calibration. They introduced a SLAM-based approach for the extrinsic auto-calibration of fixed multi-camera setups, extending its utility even to camera setups with non-overlapping views. This was a significant breakthrough because non-overlapping views pose challenges due to the reduced environmental information available and its incompatibility with point registration algorithms to directly establish equivalent points across overlapping camera frames.

Some researchers [151–153] proposed automating the target-based calibration approaches by eliminating the need for user intervention during the calibration process, although an artificial target was used. For instance, although Sim *et al.* [151] used a checkerboard, the approach is considered indirect because it does not directly find corresponding features in the LiDAR and camera data. Instead, it estimates the extrinsic calibration parameters by minimising the distance between corresponding features projected onto the image plane. The algorithm uses a v-shaped calibration target with edge and centerline features, and the distance between these features is minimised to estimate the calibration parameters.

Guindel *et al.* [152] and Beltrán *et al.* [153] automated the pattern-based calibration using algorithms to perform extrinsic calibration of lidar-stereo camera pairs without user intervention, i.e. extracts corners, edges and/or lines from the target with additional markers on the checkerboard using algorithms like plane segmentation (usually using RANSAC algorithm or its variance) for point clouds and somethings for image or some other edge or plane segmentation algorithm for images, the Levenberg-Marquardt (LM) algorithm for parameter refinement, a point registration algorithm to establish optimal rigid transformation between the extracted planes from images and point clouds and optimisation algorithms to make the approach more robust to errors. The authors highlighted that the calibration approach is designed to handle the constraints commonly found in automotive setups, such as low-resolution and specific sensor poses.

Davison *et al.* [154] proposed an automatic motion-based calibration to capture synchronised video streams during the robot's traversal through unprepared environments. The captured video sequences were individually processed using an adapted version of the MonoSLAM algorithm, culminating in refined maps and feature correspondences. These outputs undergo further refinement and alignment through bundle adjustment, ultimately facilitating global optimisation for the estimation of relative camera poses. Results showed the method's proficiency in a four-camera setup, evidenced by smaller differences between the method's output and the commercial software's results. However, the authors acknowledged the potential constraints of global optimisation in specific scenarios such as

occluded and complex terrains. Furthermore, the construction of an environment map for each camera before bundle adjustment is a computational caveat.

Moghadam *et al.* [155] proposed a targetless line-based extrinsic calibration of range and image sensors using 3D and 2D line feature extraction and matching techniques for range and image respectively in the absence of external artificial targets. This novel algorithm capitalises on the inherent linear features present within the scene to accurately determine the rigid transformation between the sensors' coordinate frames. To achieve this, the technique involves a dual approach of extracting 3D lines from the point cloud and 2D line segments from the image, followed by establishing correspondence between them. For the 2D image line extraction, the process utilises the Canny edge detector along with conventional morphological operations to discern dominant edge features within the input images. Subsequently, the Hough Transform technique is employed to identify potential line segments from the extracted edge features.

To extract the 3D lines, the point cloud data is organised via a kd-tree structure, and the region-growing segmentation technique is employed to extract lines from the point cloud. Two distinct types of 3D lines are extracted: line segments originating from the intersections of two planes, and linear features situated at the boundaries of planes. To establish correspondence between 2D image lines and 3D point cloud lines, a point registration technique is employed. The nonlinear least squares optimisation technique enabled the accurate estimation of relative translation and rotation between the coordinate frames of the sensors. Although the algorithm showed promising results and is not limited to particular types or configurations of sensors, the region-growing algorithm can be a resource-intensive process and the overall accuracy of the calibration is dependent on the accuracy of the line detection for both sensors. This approach is similar to the approach of calibration using multiple natural targets in the environment like line and plane correspondence by Zhou *et al.* [156].

Pandey *et al.* [157] proposed a method to achieve automatic extrinsic calibration between the lidar and an optical camera system using the maximising mutual information derived from surface intensities measured by the sensors. It is a data-driven approach that eliminates the need for fiducial calibration targets. To achieve this the Cramér-Rao lower bound (CRLB), was computed for estimated calibration parameter variance. The algorithm relies on a mutual information (MI)-based cost function and leverages a kernel density estimate for a probability distribution. Results showed that the proposed estimator serves as an unbiased, minimum variance estimate for calibration parameters. Remarkably, the mean error tends toward zero as the amount of calibration data increases. Although the authors did not indicate any drawbacks, the success of this approach is dependent on its ability to accurately measure surface intensities which could be misleading in low illumination or the presence of reflected surfaces. Furthermore, the accuracy increases with data therefore it can be CPU intensive to guarantee its efficiency.

Yang *et al.* [158] highlighted the need for continuous refinement of automatic calibration by introducing an online refinement process. The study proposed a rapid deployment of monocular visual-inertial systems (VINS) through the utilisation of the natural visual features in the environment. A monocular VINS pipeline incorporating IMU preintegration and nonlinear VINS with calibration refinement was presented consisting of three key phases: estimating camera-IMU rotation, recovering camera-IMU translation and other navigation parameters, and optimising the complete 6-DOF camera-IMU calibration. This was achieved using a combination of linear sliding window estimation, nonlinear optimisation, and continuous refinement. Results indicated the method's ability to achieve highly accurate calibration of the camera-IMU transformation, with errors notably below 0.02 m in translation and 1° in rotation. However, specific techniques employed for this online refinement are not detailed. Moreover, the efficiency of this method is dependent on its ability to continuously access the internet update environment's natural visual features.

The advancement of deep learning has brought about a paradigm shift in various fields including robotic perception. The use of machine learning for calibration is generally

referred to as learning-based calibration. Machine learning has led researchers to propose robust frameworks/pipelines and tools for multi-sensor calibration. For instance, CalibNet, a geometrically supervised extrinsic calibration using 3D spatial transformer networks was proposed by Iyer *et al.* [159] to develop a self-supervised deep network for accurate estimation of 6-DoF rigid body transformation between the LiDAR and camera without relying on special targets or human intervention. CalibNet uses a 3D Spatial Transformer network to learn to solve the calibration problem by leveraging geometric and photometric consistency. The network was trained using RGB images and miscalibrated depth maps obtained by projecting point clouds onto the camera plane. The network is trained to minimise the photometric and point cloud distance losses. Results showed that CalibNet can correct for miscalibration up to -20 degrees in rotation and -0.2 meters in translation, with low mean absolute errors. CalibNet, however, requires large amounts of data and good motion estimates for accurate results, which could limit its applicability to offline scenarios.

Lv *et al.* [160] also proposed a CNN-based LiDAR-camera extrinsic calibration technique to enhance the effectiveness of CNN-based approaches for LiDAR-camera calibration by incorporating correspondences across various locations. This is achieved by integrating a correlation layer to facilitate matching capabilities and outlining a calibration flow to depict the initial projection's deviation from the ground truth. This guides the estimation of extrinsic parameters using the EPnP algorithm within the RANSAC framework. The performance of CFNet was evaluated on two public datasets, KITTI and KITTI360, and demonstrates its superiority over state-of-the-art CNN-based methods in terms of calibration accuracy. However, details on its applicability to real-time calibration were not discussed.

Table 2-2 Summary of extrinsic calibration techniques

Reference	Calibration Technique	Advantage	Disadvantage
[142,143,148],	Checkerboard calibration	Mid-high accuracy	Tedious process and can be resource-intensive.

[144]	Sphere-based	Robust and mid-high accuracy	Can be a tedious process and can be computationally expensive
[150]	Motion-based	Low-mid accuracy	May require continuous calibration
[151–153]	Automatic (Automating the target-based calibration)	Low-mid accuracy	May require continuous calibration
[154]	Motion-based	Low-mid accuracy	Can have a global optimisation problem in complex terrain. Can be CPU intensive
[157]	MI-based	Low-mid accuracy	Accuracy is dependent on data and Intensity values can be misleading.
[158]	Online-based	Low-mid accuracy	Accuracy is dependent on visual features and internet availability
[159,160]	Learning-based	Low-mid accuracy	A large amount of training data, and its applicability in real-time calibration

2.5 MULTI-SENSOR FUSION TECHNIQUES

Autonomous navigation requires the mobile robot to perceive, recognise, and map the surrounding objects and obstacles, as well as to plan and execute optimal paths to reach its goals. To achieve autonomous navigation, robots often rely on multiple sensors that provide different types of information, such as image sensors, which include stereo, monocular, wide area, and fish-eye, 2D Laser scanners and 3D LiDAR sensors, RaDAR sensors, ultrasonic sensors, IMUs, GNSS/GPS, and wheel encoders [161–163]. Multi-sensor fusion

techniques are essential for precise positioning and state estimation of mobile robots. These techniques involve collecting data from multiple sensors and fusing them to improve the accuracy and reliability of the perception and navigation of mobile robots. The fusion process involves analysing and screening the signal quality of sensors, filtering, and processing sensor information to reduce noise, and fusing data from multiple sensors.

The fundamental purpose of multi-sensor fusion is to accurately determine the robot's location to which the sensor is mounted or infer its location by relying on some known landmarks in the environment relative to the sensor, and finally identifying obstacles in the environment that could obstruct the smooth navigation of the robot the sensors are mounted on [164]. Once this has been successfully achieved, a mapping of the environment [165–168] can be achieved and the most optimised path for the motion of the robot can also be achieved [169,170].

Multi-sensor fusion can be broadly divided into three main categories according to the fusion technique applied; early fusion, intermediate fusion, and late fusion [161–163]. Early sensor fusion involves combining raw data obtained from different sensors to create a more informative representation of the environment or the phenomenon under observation. Intermediate sensor fusion involves extracting relevant features from the raw data of each sensor and then combining them to form a unified feature vector or matrix. Late sensor fusion involves making independent decisions from each sensor data and then combining them to obtain a final decision or output [21,135].

Some authors categorise multi-sensor fusion into centralised and decentralised fusion architecture. Centralised fusion architecture involves the integration of data from multiple sensors at a central location, while decentralised fusion architecture involves the integration of data from multiple sensors at different levels for different tasks, in a distributed manner. Decentralised fusion architecture is mostly adopted in mobile robots for assisted navigation. Although multi-sensor fusion is divided into three main categories, the techniques like the Kalman filter, and Dempster-Shafer theory used for fusion are employed

across the different categories [171]. These categories are not clearly distinguished by some authors, especially when the hybrid multi-sensor fusion approach is applied. This is particularly common in a decentralised sensor fusion architecture, where fusion happens in a decentralised manner.

2.5.1 Probabilistic-based techniques

Probabilistic-based fusion techniques are methods that combine sensor data based on probabilities of relevance calculated from the performance of the individual sensors. These techniques are based on probabilistic models that use statistical methods to estimate the state of the system based on the data from the sensors. These techniques have been applied to various domains such as information retrieval, text processing, and multisensor environments. Probabilistic-based fusion techniques aim to improve the quality and reliability of the fused data by taking into account the uncertainty and variability of the source data. Probabilistic-based fusion techniques include Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), Kalman filter (KF) and its variants, Monte Carlo Models (MCMs) and Particle filter (which is a variant of MCMs, also called sequential filtering).

Bayesian filtering is the most common technique used by probabilistic-based fusion algorithms and it uses Baye's theorem to estimate the probability of a state given the observations. HMMs [172–174] and CRFs [175,176] are examples of Bayesian filters (BFs). KF and its variants are also a type of BF that uses recursive Bayesian filtering to estimate the state of a system based on noisy measurements [177–180]. The MCMs [181] and Particle filters [182] are another type kinds of BFs that use random sampling techniques to approximate the posterior distribution of the state given the observations. By training these filters on a set of data, the algorithm's parameters are optimised to accurately estimate the state of the system based on the observations.

These filters are based on Baye's theorem, which is a formula that expresses the conditional probability of an event given some evidence. The Bayes' theorem can be written as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2.1)$$

where $P(A | B)$ is the posterior probability of event A given evidence B , $P(B | A)$ is the likelihood of evidence B given event A , $P(A)$ is the prior probability of event A , and $P(B)$ is the marginal probability of evidence B .

The Kalman filter is a probabilistic model that estimates the state of a linear system from noisy and inaccurate measurements over time, by solving the linear-quadratic-Gaussian problem. The filter uses Bayesian inference to update the state recursively in two stages: prediction also known as the likelihood of evidence or prior probability and correction also known as the posterior probability. In the prediction stage, the filter uses the system dynamics model to forecast the next state. It uses the state transition function, F_k and the control input, B_k to project the state, $\hat{x}_{k|k-1}$ and its error covariance, $P_{k|k-1}$ to the next time step. In the correction stage, based on the new observation, the filter uses the measurement model to refine the predicted state, $\hat{x}_{k|k}$ and covariance, $P_{k|k}$ using a weighted average called the Kalman gain, K_k . The filter repeats these stages for each new measurement. The main drawback of the Kalman filter is that it assumes linearity and normality of the system and the noise. It also assumes that the noise is white and Gaussian [178,180]. The KF can be seen as a generative model which has an identical structure to the HMM, except that the discrete state and observations are replaced with continuous variables sampled from Gaussian distributions. To handle non-linearity, the Extended Kalman Filter (EKF) was developed. It linearises the system around the current state estimate using the first-order Taylor series. However, this approximation is only valid for small deviations and does not fully address the non-linearity. KF can be expressed mathematically as:

Predict:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \quad (2.2)$$

$$P_{k|k-1} = F_k P_{k|k-1} F_k^T + Q_k \quad (2.3)$$

Correct:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \quad (2.4)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - H_k \hat{x}_{k|k-1}) \quad (2.5)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (2.6)$$

where:

- $\hat{x}_{k|k-1}$ is an estimate of x at time $t = k$ given measurements up to time $t = j$.
- F_k is a state transition matrix that describes how to evolve x from time $t = k - 1$ to time $t = k$ in the absence of control inputs.
- B_k is a control-input matrix that describes how control inputs affect x .
- u_k is a control input vector at time $t = k$.
- $P_{k|k-1}$ is an estimate of the error covariance matrix at time $t = k$ given measurements up to time $t = j$.
- Q_k is a process noise covariance matrix that describes how much noise is added due to process dynamics.
- K_k is a Kalman gain matrix that determines how much weight should be given to measurements in the correction stage.
- y_k is a measurement vector at time $t = k$.
- H_k is a measurement matrix that maps true state space into measurement space.
- R_k is a measurement noise covariance matrix that describes how much noise is added due to measurement errors.

The prediction stage predicts what will happen next, i.e. at the time $t = k$ based on previous estimates i.e. at time $t = k - 1$, while the correction stage corrects these estimates based on new measurements. The Kalman gain, K_k determines how much weight should be given

to measurements in the correction stage. The process noise covariance matrix, $P_{k|k-1}$ describes how much noise is added due to process dynamics, while the measurement noise covariance matrix, $P_{k|k}$ describes how much noise is added due to measurement errors.

The Unscented Kalman Filter (UKF) is a recursive Bayesian filtering algorithm that estimates the state of a nonlinear system from noisy and inaccurate measurements over time. It deals with non-linearity by using a deterministic sampling technique called unscented transformation (UT) to select a minimal set of sample points, called sigma points around the mean. The sigma points are then propagated through the nonlinear functions, from which a new mean and covariance estimate are then formed. The UKF avoids the need to linearise the system or compute Jacobians, which can introduce errors or be computationally expensive. The UKF was proposed to overcome the limitations of the KF and EKF. The UKF does not require linearisation, which makes it suitable for nonlinear systems described by the state-space model. The UKF can be expressed mathematically as follows [183]:

Predict:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}) + W_{k-1} \quad (2.7)$$

$$P_{k|k-1} = E[(\hat{x}_{k|k-1} - \bar{x}_k)(\hat{x}_{k|k-1} - \bar{x}_k)^T] + Q_{k-1} \quad (2.8)$$

Correct:

$$\hat{y}_k = h(\hat{x}_{k|k-1}) + V_k \quad (2.9)$$

$$S_k = E[(\hat{y}_k - \bar{y}_k)(\hat{y}_k - \bar{y}_k)^T] + R_k \quad (2.10)$$

$$C_k = E[(\hat{x}_{k|k-1} - \bar{x}_k)(\hat{y}_k - \bar{y}_k)^T] \quad (2.11)$$

$$K_k = C_k S_k^{-1} \quad (2.12)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(\hat{y}_k - \bar{y}_k) \quad (2.13)$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T \quad (2.14)$$

where:

- $\hat{x}_{k|k-1}$ is an estimate of x at time $t = k$ given measurements up to time $t = j$.
- f is a state transition function that describes how to evolve x from time $t = k - 1$ to time $t = k$ in the absence of control inputs.
- W_{k-1} is a process noise vector that describes how much noise is added due to process dynamics.
- $P_{k|k-1}$ is an estimate of the error covariance matrix at time $t = k$ given measurements up to time $t = j$.
- E denotes the expected value.
- $\hat{x}_k$ is the mean of sigma points at time $t = k$.
- Q_{k-1} is a process noise covariance matrix that describes how much noise is added due to process dynamics.
- h is a measurement function that maps true state space into measurement space.
- V_k is a measurement noise vector that describes how much noise is added due to measurement errors.
- $\hat{y}_k$ is a measurement vector at time $t = k$.
- $\bar{y}_k$ is the mean of predicted measurement sigma points at time $t = k$
- R_k is a measurement noise covariance matrix that describes how much noise is added due to measurement errors.
- S_k is an innovation covariance matrix at time $t = k$ given measurements up to time $t = j$.
- C_k is a cross-covariance matrix at time $t = k$ given measurements up to time $t = j$.

The Particle filter (PF), also known as sequential Monte Carlo (SMC) filtering, is a recursive Bayesian filtering algorithm that estimates the state of a nonlinear system from

noisy and inaccurate measurements over time. The PF finds a group of random particles propagating in the state space to describe the state of the system. It uses a set of random samples also called particles with associated weights to represent the posterior distribution of the state. The PF applies to any nonlinear system described by the state-space model, and its accuracy approaches optimal estimation. It can adapt to nonlinear and non-Gaussian systems well and has no restrictions on measurement and processing noises.

The particles are updated using two steps: prediction and correction. In the prediction stage, the filter uses the state transition function and some process noise to move the particles to the next time step. In the correction stage, based on the new observation, the filter uses Bayes' rule to update the weights of the particles. The PF can handle multimodal distributions and non-Gaussian noise, but it may suffer from particle depletion or degeneracy. The PF is considered a solution to the limitations of the Kalman filter (KF) and its variants, but this comes at a computational cost, especially in high-dimensionality state spaces [184,185].

PF can be applied by utilising Monte Carlo (MC) simulation and integration to represent the state posterior probability distribution function, $p(x_k|y_{1:k})$, with the help of a set of $N \in \mathbb{N}$ particles $x_k^i, i = 1, 2, \dots, N$, with corresponding weights $w_k^i, i = 1, 2, \dots, N$. Then, $p(x_k|y_{1:k})$ can be approximated by the discrete mass probability distribution of the weighted particles, $\{x_k^i\}_{i=1}^N$ as:

$$p(x_k|y_{1:k}) \approx \sum_{i=1}^N w_k^i \delta(x_k - x_k^i) \quad (2.15)$$

where x_k is the state of the system at time k , $y_{1:k}$ are the measurements up to time k , w_k^i are the weights associated with each particle, and $\delta(x_k - x_k^i)$ is a Dirac delta function centered at x_k^i , which represents the discrete mass probability distribution of the i -th particle. As $p(x_k|y_{1:k})$, is usually unknown; this MC simulation is based on importance

sampling, namely a known probability (importance) density, $q(x_k|y_{1:k})$ which is usually used to represent x_k^i , and w_k^i is represented as:

$$w_k^i \propto w_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, y_k)} \quad (2.16)$$

To reduce the degeneracy of the particle set and improve the approximation of the posterior distribution, resampling is needed to get the effective number of particles. The effective number of particles is defined as:

$$N_{eff} = \frac{1}{\sum_{i=1}^N (w_k^i)^2} \quad (2.17)$$

There are different methods for resampling, such as multinomial, stratified, systematic, or residual resampling [186,187].

Many Bayesian filtering methods have been proposed by researchers for the fusion of multiple sensors. For instance, He *et al.* [188] proposed a hidden Markov model (HMM) framework for the probabilistic fusion of multiple sensors within a hidden Markov model (HMM) framework for user positioning on mobile devices, for indoor scenarios. Motion sensors, lidar and cameras, and WiFi RSSI were fused to estimate the user's positioning in an indoor scenario. A graph structure was used for storing the model constructed by multiple sensors during offline training and developing a multimodal particle filter for seamless information fusion during online tracking.

The result of this technique was promising; however, it was an indoor-based experiment implemented on the iOS platform and the authors acknowledged that it would need to be executed with state-of-the-art indoor positioning methods. Yang *et al.* [189] proposed a two-stage multi-sensor fusion positioning system with seamless switching for a cooperative

mobile robot and manipulator system. Ultrasonic sensors were fused with inertia measurement units (IMU) using an extended Kalman filter (EKF), and detection of an ArUco marker with a visual sensor. A positioning-switching strategy is implemented based on the state of the visual sensor, allowing the robotic manipulator to reposition the mobile robot seamlessly. This proposed approach, like most applications of Bayesian filters for multi-sensor fusion [190,191] that are used for positioning or the SLAM process, requires each sensor to be able to provide meaningful measurement information about the position of the robot before they are used to fuse and estimate position. Therefore, the success of the positioning system, when used with vision sensors is dependent on the efficient localisation of known landmarks by visual sensors.

For this reason, Obando-Ceron *et al.* [192] proposed the use of CRFs to model with multiple geometry and appearance potentials for estimating dense depth maps from camera and LiDAR data. The model fuses raw sensor streams at a low level, resulting in dense and precise depth estimates that can be used as a unified multi-modal data source for higher-level estimation problems. The proposed method is evaluated using the KITTI benchmark dataset. The proposed method is compared with state-of-the-art techniques using the KITTI dataset, and its performance is evaluated in terms of depth estimation accuracy.

2.5.2 Learning-based techniques

Deep learning-based fusion techniques, particularly convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory (LSTM) networks and transformers have gained significant attention, over the past decade, due to their ability to identify complex patterns in achieving feature-level and pixel/early fusion of multi-sensors for mobile robots. CNNs excel in image and video data fusion tasks, while RNNs are effective for sequential data fusion from sensors like IMUs.

Many researchers have proposed various architectures and attention mechanisms, to capture spatial and temporal dependencies for multi-sensor fusion. This includes approaches that process information from individual sensors individually before fusing

them, projecting raw LiDAR or RaDAR point cloud onto the image plane and using deep learning techniques to perform a semantic representation of fused information [193–195], performing a 2D bounding boxes to 3D regression on objects detected in the image plane to estimate the 3D equivalent object localisation in image coordinate of the 2D detected images [196], using an attention mechanism to aggregate features extracted from multiple sensors and leave only important features for object detection [197–199] and several other early or feature-level fusion techniques.

The proposed approaches by Dong *et al.* [193] and Zhuang *et al.* [194] are examples of sensor fusion by the projection of point cloud data from LiDAR and RaDAR onto images, see Figure 2-2. Dong *et al.* [193] developed a method for radar-camera association using deep representation learning and global reasoning. Radar points were clustered, and the centre of each object cluster was projected onto the corresponding image, using the camera and radar intrinsic and extrinsic matrices. The resulting image was trained using the anchor-based RetinaNet object detection network. A loss sampling mechanism and an ordinal loss using a deep association by representation learning called AssociationNet were introduced to address imperfect labelling from the radar noisy measurements and enforce critical human-like reasoning. Reported results showed a performance of 92.2% F1 score, which is 11.6% higher than the rule-based teacher. However, this approach relies on the image to address the noisy radar measurement. Furthermore, object detection happens in the image plane which becomes an issue in the environment with low illumination.

Zhuang *et al.* [194] introduced perception-aware losses to measure the perceptual difference between the two modalities (monocular camera and LiDAR). LiDAR point clouds were projected onto the image to provide station-depth information for the RGB image, a two-stream network to extract features from RGB images and point clouds separately and the extracted features were fused using effective residual-based fusion modules. Unlike [193] the projected image was used as a reference by the perception-aware multi-sensor fusion framework (PMF) to optimise the network that fuses the extracted perceptual features from the LiDAR-only and Camera-only Network.

Results showed that it outperformed the state-of-the-art method by 0.8% in mIoU on the nuScenes dataset. However, like the authors in [189], the reference image used for optimising multi-modal fusion is in the image frame. This is a major drawback because object localisation and position estimation are dependent on the camera view angle and field of view.

The network by Chen *et al.* [195] is one of the early studies that propose a network for high-accuracy 3D object detection in autonomous driving scenarios. This study was a major contribution to deep learning-based fusion techniques for two major reasons: BEV representation of LiDAR 3D point clouds and 2D to 3D bounding box regression. The research also reiterated the importance of depth information for deep learning object localisation tasks. The MV3D network encodes the sparse 3D point cloud with a compact multi-view representation, like a BEV image or BEV map. It generates 3D object proposals from a bird's eye view (BEV) map and fuses multi-view features hierarchically for category classification and oriented 3D box regression. The network is trained in an end-to-end fashion using Stochastic Gradient Descent (SGD) with a learning rate of 0.001 for 100K iterations. Results showed that the MV3D network achieved superior performance when compared to the state-of-the-art methods in terms of 3D localisation, 3D detection, and 2D detection. It outperformed the existing methods by around 25% and 30% AP in 3D localisation and 3D detection tasks, respectively. The major drawback was the computational cost of using a region proposal network variant with the VGG-16.

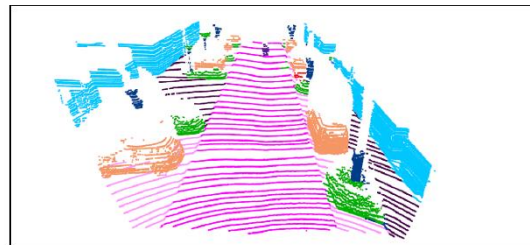
With the advent of vision transformers [200], many researchers have begun exploring the use of its attention for depth estimation, global context among the LiDAR and camera features and its robustness to sensor misalignment. Studies like [197–199] explored the use of transformers to capture global contextual information and improve driving performance. Chen *et al.* [199] used a modality-agnostic feature sampler (MAFS) to sample and fuse extracted features from each sensor. Proposal-based methods and feature projection-based methods for LiDAR and camera fusion were also incorporated.



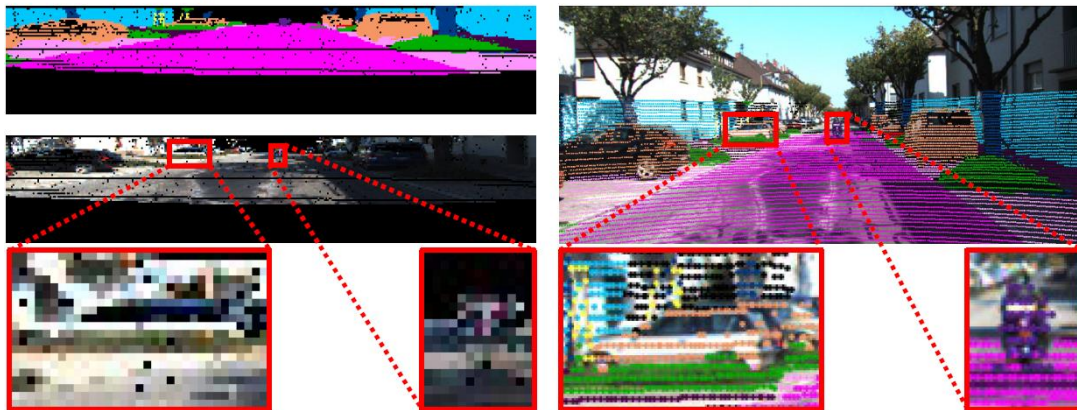
(a) Project Radar points unto an image [193]



(a) Input RGB Image



(b) Input Point Cloud



(b) Projecting a Lidar point cloud unto an image [194]

Figure 2-2 Projecting point clouds onto an image

Drewe *et al.* [197] proposed InverseAug and LearnableAlign, for effectively aligning and fusing camera and lidar features in multi-modal 3D object detection. InverseAug inverses geometric-related augmentations to enable accurate geometric alignment, while LearnableAlign leverages cross-attention to capture correlations between image and lidar features during fusion. Results from these approaches showed their effectiveness in multimodal sensor fusion. However, like most vision transformers, performance increases and it becomes robust when trained on a huge dataset as compared to its CNN counterparts. This makes training with vision transformers computationally and memory expensive process [201].

The primary purpose of fusion via deep learning is to train the models to accurately relate the objects or landmark location in the sensor frame to the global or world coordinate system and align with other sensor modalities. This could be LiDAR, RaDAR, or camera image. Many studies approach this without the need to use the intrinsic or extrinsic parameters of the sensors, however, because these approaches are entirely data-dependent their robustness is limited to the data used for training the model. More recently, many researchers have approached this problem as a unified BEV problem and have explored the use of deep learning techniques in achieving a unified BEV for multimodal sensors.

2.5.3 Unified BEV for multi-sensor fusion techniques

Although the use of transformers to develop BEV semantic images has been explored [202–205], this section will focus on research that employs multi-layer perceptron (MLP) networks for transforming monocular images to BEV fusion of images with other sensor modalities. The main drawback of the standard CNN network is that it does not excel in global context awareness-related tasks. As a result, they are not able to take into account the surrounding context of an object or landmark when making predictions, or decisions. This is one of the main attractive features that made the vision transformer [200], which was introduced in 2021, excel in view synthesis and view perceptive transformation tasks because of their self-attention mechanism. Before the use of vision transformers, MLPs

[206–216], the use of generative adversarial networks (GANs) [217] and time series networks like LSTMs [218,219] were used to address this limitation.

Particularly, for view transformation in the field of autonomous driving, one major contribution that has led to other research [206–210,213,214] is the lift splat shoot (LSS) concept proposed in 2021 by Phillion *et al.* [211]. The authors proposed an end-to-end architecture that extracts a bird’s-eye-view (BEV) representation able to learn how to fuse predictions from multiple cameras into a single cohesive representation of the scene while being robust to calibration errors. The proposed view transformation method lifts each image individually into a frustum of features for each camera and then splats all frustums into a rasterized BEV grid. LSS addressed the view transformation of monocular images as a rasterization problem.

Inspired by the use of inverse perspective mapping (IPM) for view transformation [212,215] and particularly multi-view synthesis techniques [220,221], the intrinsic matrix of the camera is used to establish an approximate depth value for each pixel in the image and this is used to lift the 2D image to its equivalent 3D point cloud. A common BEV grid is created, and the lifted point cloud is splatted onto the BEV grid. The shooting aspect of this work is a test of its ability to perform motion planning i.e. to effectively localise objects and landmarks and use them for position estimation of the robot’s location in the environment. Motion planning is performed using the inferred BEV map to predict the trajectory for the ego vehicle i.e. the mobile robot on which the sensors are mounted. Although the result showed it was superior when compared to similar camera-only BEV representation, it was still not as efficient as methods that utilised depth information in view transformation tasks. Furthermore, this approach was a computationally intensive process due to the rasterization of each pixel. However, the authors introduced the quick cum sum and a data argumentation technique to improve computational cost and performance.

A notable improvement to the LSS [211] concept was the BEVDepth concept proposed by Li *et al.* [210] which introduced a camera-awareness depth estimation module and a Depth

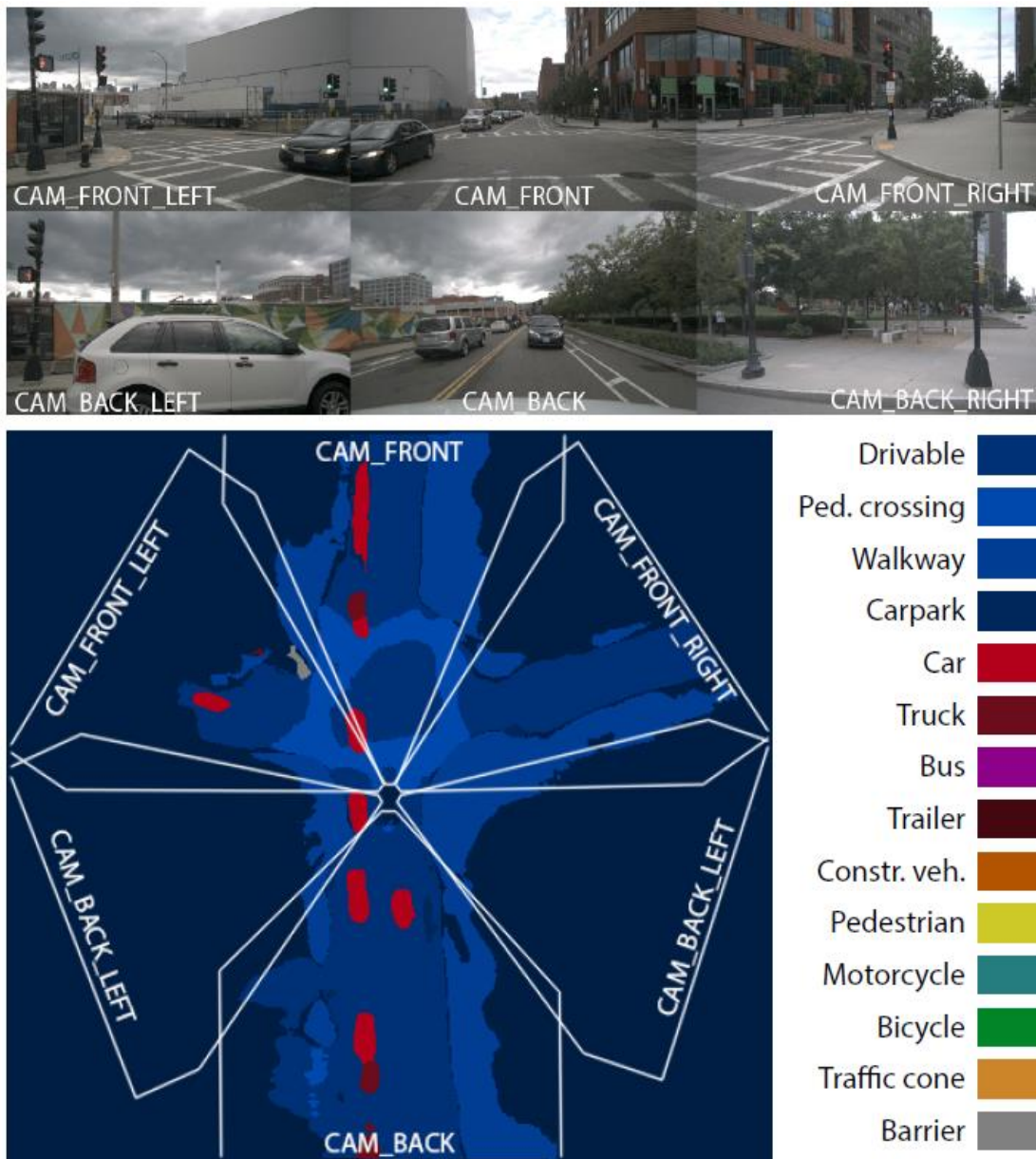
Refinement Module to improve the depth prediction accuracy of the LSS and address the inadequacy of the depth estimation in camera only approach. This module used explicit depth supervision like data from radar or lidar sensors to supervise the depth estimation of the camera and counter the side effects of imprecise feature unprojection. This was surprisingly successful and BEVDepth achieved a new state-of-the-art 60.9% NDS score on the challenging nuScenes test set while maintaining high efficiency. It is the first camera model to reach a 60% NDS score in the year 2022. This also highlights the importance of multi-sensor fusion especially the inclusion of a direct depth measurement sensor to complement the camera sensor which is rich in colour information and resolution.

The BEVDepth contribution led many researchers to explore and propose many variants [206–209,213,214] of the LSS and BEVDepth architecture. One such is the BEVFusion [208], which fine-tunes existing approaches, reducing the computation cost of multi-sensor fusion models by half and improving accuracy on small and distant objects, as well as in rainy and night conditions. It diagnosed and lifted efficiency bottlenecks in the view transformation with optimised BEV pooling, reducing latency by more than 40×. It also introduced some notable image and LiDAR data augmentation techniques. BEVFusion achieved state-of-the-art results on the nuScenes detection benchmark, with higher mAP and NDS for 3D object detection and higher mIoU for BEV map segmentation compared to previous methods. It also demonstrates close-to-real-time inference speed on a desktop GPU. However, the authors highlighted that BEVFusion has performance degradation in joint multi-task training and has not fully unlocked the potential for larger inference speedup in the multi-task setting.

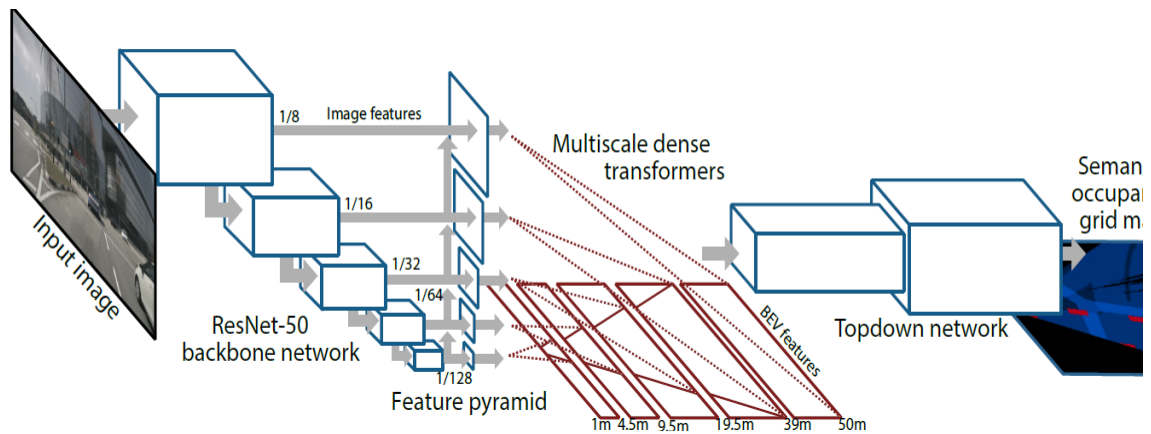
The use of the sensor's calibration matrix has also been explored by [212,215,216,222] as a viable approach to establishing robust and reliable models for global context awareness for view transformation of monocular camera images. These techniques use the well-known computer vision concept called inverse perspective mapping (IPM) [223] as a baseline for training deep learning MLP networks to view the transformation of camera images to BEV images. This is further described in chapters 4 and 5 of this thesis. For

instance, a study by Roddick *et al.* [212] proposed in 2020, is an example of one of the earlier studies that explored this method for view transformation. The authors introduced a unified approach for estimating maps directly from monocular images using a single end-to-end deep learning architecture. It adopts a semantic Bayesian occupancy grid framework $p(m_i^c | z_t)$ based on Baye's theorem used by Bayesian filters, as presented in section 2.4.1. This concept gained popularity in the deep learning community [224] in 2018 for the BEV maps, allowing information accumulation over multiple cameras and timesteps. It also incorporates ground plane estimation, road segmentation, and 3D object detection in the deep learning architecture. This approach is in line with the end goal of BEV multi-sensor for autonomous mobile robots, which is to create an estimated map of the robot's environment and the location of the robot relative to the objects. It also uses two well-known computer vision concepts i.e. the IPM [225] and the Bayesian occupancy grid [224,226] as the baseline of the network. Results showed a relative improvement of 9.1% and 22.3% compared to the best-performing existing method on the NuScenes and Argoverse datasets, respectively.

Sim2Real [215] is another study that explored the use of the IPM approach to transform 360° camera image to BEV. This approach does not rely on manually labelled data and was trained on the CARLA synthetic dataset. The result showed its efficiency in achieving semantic BEV from images. However, its performance on real-world data is not explicitly mentioned. Further research [222] explored the use of just the IPM method for the front-view monocular image to BEV panoptic semantic segmentation at a pixel level. Figure 2-3 shows the Roddick *et al.* [212] architecture as an example of how camera to BEV is achieved.



(a) Fusing multi-cameras to semantic BEV map using Bayesian occupancy grid



(b) Applying the IPM concept to extracted image features for a single image

Figure 2-3 Architecture for Multi-image Fusion to BEV [212]

2.5.4 IoT protocols and data management techniques for ACAMDs

For an IoT framework to be deemed suitable for multi-sensor fusion which involves time-sensitive networking, it needs to meet the following requirements: low latency, high throughput, and real-time processing capabilities amongst other requirements [227–231]. One of the IoT architectures that has been extensively explored and recommended for real-time multi-sensor networks in smart cities is the edge architecture. This is because of its setup to allow data processing at the network's edge, which reduces latency and bandwidth usage [232–237]. For instance, the Cloudlet computing (CC) architecture which is the type of mobile cloud computing (MCC) [238], consists of small cloud infrastructures called cloudlets for performing data processing and storage close to the end devices. This in turn improves the idea of a decentralised multi-sensor fusion architecture and also improves security because it is difficult for a cyber attack to occur on all cloudlets at the same time. It also improves scalability as more cloudlets can be added to the entire architecture.

Another example is the fog computing framework [239–241] that allows a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralised devices to communicate and potentially cooperate among themselves and with the network to perform storage and processing tasks without the intervention of third parties. De Donno *et*

al. [233] defined the fog computing network as a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things, thereby accelerating the velocity of decision-making. This summarises the advantages of the fog network and its suitability for multi-sensor fusion architecture. More details of the fog architecture for the IoT Framework are highlighted in chapter six of this report.

2.6 CONCLUSION

This chapter offers a comprehensive literature review on IoT-based multi-sensor fusion techniques for ACAMDs, with a focus on three critical aspects: sensor data preprocessing techniques, automatic extrinsic calibration methods, and multi-sensor fusion techniques. The existing body of work concerning sensor data preprocessing techniques encompasses various approaches aimed at noise reduction, enhancing sensor data quality, extracting pertinent information, and minimising resource consumption during the multi-sensor fusion process. Automatic extrinsic calibration techniques are designed to ensure accurate sensor synchronisation while harmonising their orientation, position, and temporal data acquisition. However, the selection of preprocessing and extrinsic calibration techniques depends on their intended application. Furthermore, the ongoing challenge lies in striking a balance between speed and performance.

The multi-sensor fusion techniques subsection presents algorithms for effectively combining sensors, such as cameras and lidar sensors, to generate rich and dependable information for informed decision-making. Two prominent categories of techniques are probabilistic-based and learning-based methods. Additionally, a recent learning-based approach known as the unified bird's eye view (BEV) technique, was introduced. This approach employs deep learning techniques to fuse camera sensors with lidar or radar sensors, creating a unified top-down view suitable for Simultaneous Localisation and Mapping (SLAM) and autonomous navigation tasks. However, it is essential to note that while the BEV technique holds promise, it is still in the early stages of development,

requiring extensive testing and optimisation to fully realise its potential and applicability in real-time scenarios.

Lastly, the chapter reviews IoT protocols and management techniques aimed at improving data latency in robotic communication networks. These protocols facilitate efficient data sharing among sensors or mobile robots.

CHAPTER 3: LIDAR POINT CLOUDS

3.1 INTRODUCTION

Light Detection and Ranging (LiDAR) sensor is a common sensor of choice for autonomous navigation systems (robots) to capture 3D point cloud information for perception tasks, like object detection and SLAM. This is because of its ability to capture accurate distance information about objects in the environment. This makes it a very crucial sensor for significantly improving the accuracy of depth perception and environmental mapping tasks. The fusion of Lidar data with other sensors enables more robust and reliable SLAM systems.

This chapter explores the concept of LiDAR point cloud and its significance in the context of IoT-based multi-sensor fusion techniques for adaptive corporative assistive devices. The chapter presents the generation and processing of LiDAR point clouds, including the challenges associated with processing large amounts of data and segmenting point cloud information in complex terrain. Furthermore, it proposes a novel algorithm titled “FASTSEG3D: A Fast and Adaptive Ground Segmentation Method of 3D Point Cloud for the efficient segmentation of ground and non-ground points in complex environments. This algorithm would be used to process the lidar point cloud information before it is passed into the multi-sensor fusion architecture presented in chapter five.

3.2 LIDAR POINT CLOUD

LiDAR technology is a remote sensing system that captures spatial information about the environment by emitting laser beams and measuring the time it takes for the beams to return to the sensor. It uses the concept of time-of-flight measurement, which means that it calculates the distance to an object or surface by multiplying the speed of light by the time it takes for the laser beam to emit, reflect and return. The data collected by LiDAR is stored as a point cloud, which is a collection of data points plotted in 3D space using a 3D laser scanner. Each point in the point cloud represents a real point in the environment, and the data captured includes the following: spatial information, such as the x , y , and z

coordinates of each point in the point cloud, intensity, which is the strength of the laser beam returned by each point in the point cloud, and the ring or laser beam value. LiDAR systems can also be integrated with IMUs or GNSS to capture GPS information alongside the spatial information of the environment [223]–[225], however, the data captured by the LiDAR system itself is the major focus of this chapter.

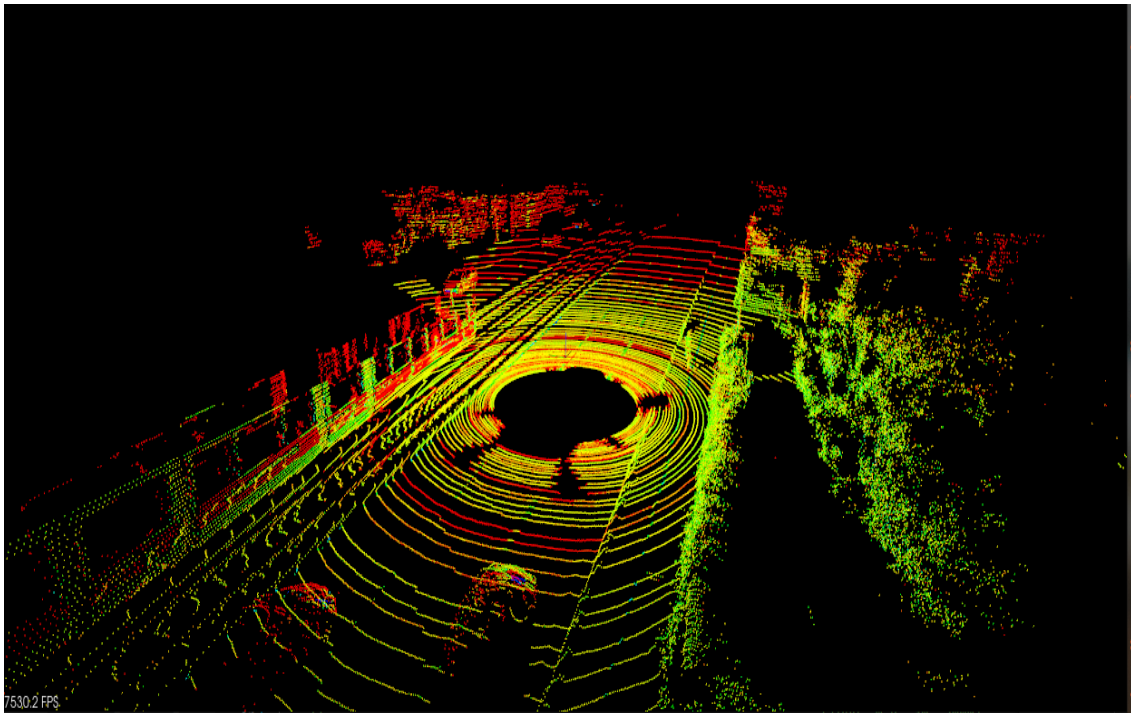
The design of the LiDAR sensor requires a comprehensive calibration of its intrinsic parameters before it is deemed ready for use by the end user. Therefore, information like its revolutions per minute (RPM), the laser beam divergence angle, also known as the vertical angle, angular resolution (horizontal/Azimuth angles), laser pulse duration, laser wavelength, detector sensitivity and scan pattern are always known and documented in the sensor datasheet. The information captured by a LiDAR sensor for a single point p_i in the point cloud, P can be expressed as:

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ i_i \\ r_i \end{bmatrix} = \begin{bmatrix} d \cos(\theta_v) \sin(\theta_h) \\ d \cos(\theta_v) \cos(\theta_h) \\ d \sin(\theta_v) \\ i \\ r \end{bmatrix} \quad (3.1)$$

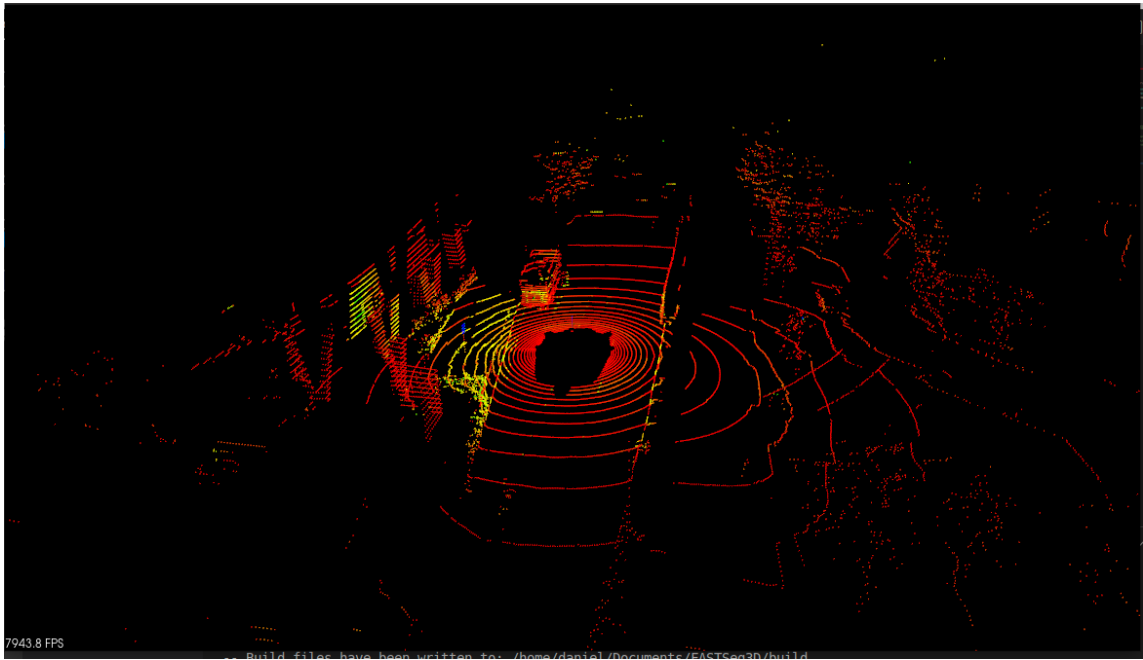
where d is the distance measured by the laser beam, θ_v is the vertical angle of the beam, θ_h is the horizontal angle of the beam, i is the intensity value and r is the ring value. This can also be explained as for a given laser beam r , its horizontal angle θ_h is determined by its position in one revolution and its vertical angle θ_v is determined by its position in one scan line. The distance d and intensity i are measured by the time-of-flight and reflection strength of the laser pulse respectively. Therefore, the point cloud by a LiDAR sensor can be expressed as follows, see Figure 3-1:

$$P = \begin{bmatrix} x_1 & y_1 & z_1 & i_1 & r_1 \\ x_2 & y_2 & z_2 & i_2 & r_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & i_n & r_n \end{bmatrix} \quad (3.2)$$

For the LiDAR sensor to be used with other sensors, certain information and steps are crucial as explained in section 2.2 of chapter two. These include its timestamp information, calibration of its extrinsic parameters relative to the other sensor's coordinate or the world coordinate and preprocessing of the raw captured unorganised point cloud. This chapter presents a novel algorithm titled “FASTSEG3D: A Fast and Adaptive Ground Segmentation Method of 3D Point Cloud Using Range Elevation Estimation (REE) and Window-Based Model Fitting (WBMF) of the 3D Point Clouds.



(a) Raw display of KITTI point cloud which uses the Velodyne 64E



(b) Raw display of nuScenes point cloud which uses the Velodyne 32E

Figure 3-1 LiDAR point cloud with depth information

3.3 FASTSEG3D

FastSeg3D is a novel algorithm developed during this study for segmenting raw captured point clouds from a LiDAR system into ground and non-ground points in real-time. As highlighted in section 2.2.1 of chapter two of this report, for a ground/segmentation technique to be effectively used as a preprocessing step, it needs to be sufficiently fast i.e. in real-time, efficient, and robust. The FastSeg3D algorithm's main novelty lies in its method of organizing raw point clouds via a unique data structure, its significant improvement to the elevation estimation technique and its method of smoothening/refining the initially estimated ground and non-ground point clouds via the window-based model fitting technique, which requires just a single iteration of the raw point clouds to properly classify points into ground and non-ground point. The FastSeg3D algorithm uses a fast point representation technique to quickly organise and filter outlier points in the point cloud at the same time. Subsequently, a fast elevation technique called the range elevation estimation (REE) technique is employed as the first module of the FastSeg3D algorithm to get a rough approximation of ground and non-ground points in a given point cloud in as

little as 7 milliseconds on average. This is further refined by using the window-based model fitting (WBMF) technique to address the over-segmentation problem, making it robust enough for complex terrains, while still achieving real-time segmentation of 19 milliseconds on average. Figure 3-2 shows a block diagram of the FastSeg3D algorithm, which includes the REE and WBMF modules.

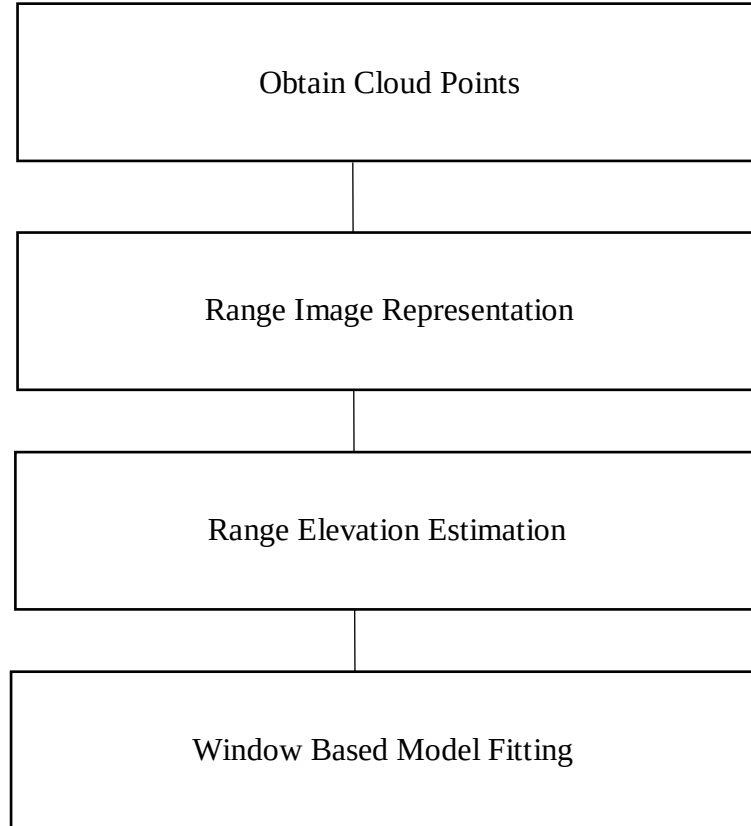


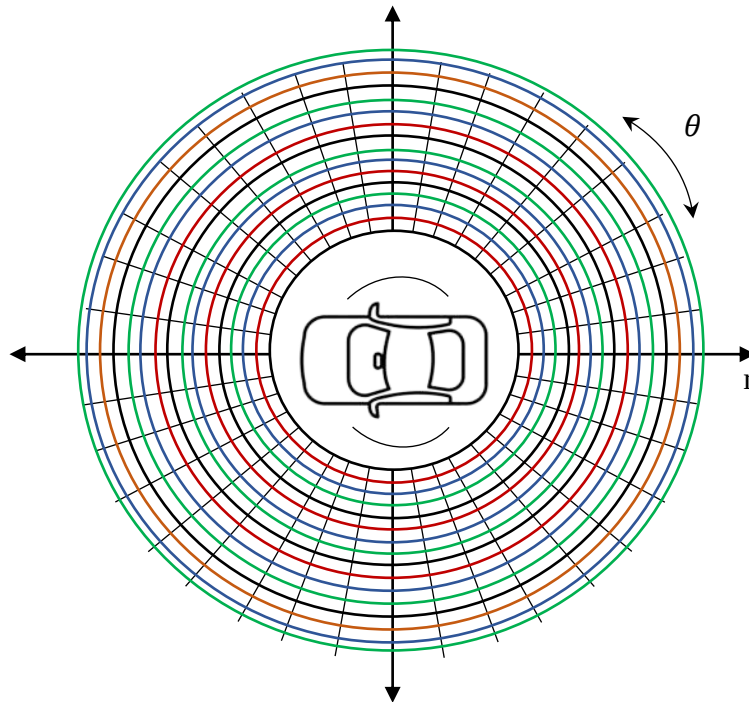
Figure 3-2 Developed FastSeg3D block diagram

3.3.1 Problem definition

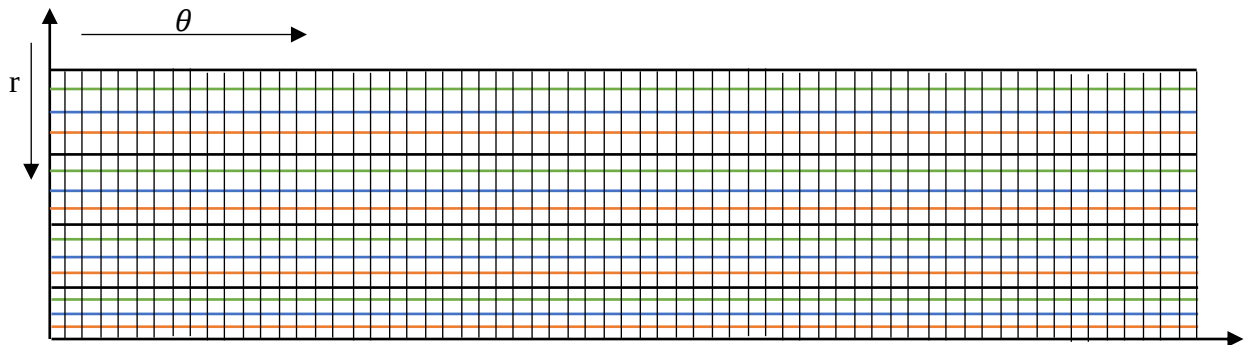
For a given 3D point p_i with cartesian coordinates $\{x_i, y_i, z_i\}$, in a 3D LiDAR point cloud $P = \{\{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}, \{x_3, y_3, z_3\}, \dots, \{x_n, y_n, z_n\}\}$, where $p_i \in R^3$ and P contain N number of points. The task is to estimate the number of points p_i in P , with guaranteed recall and precision performance, that can be classified as a ground class, including relatively flat [226] surfaces like drivable surfaces, terrain, sidewalks, and other flat

surfaces or a non-ground class, which includes vehicles, humans, and static and movable objects.

3.3.2 Organising point clouds: Point cloud representation



(a) A graphical representation of the distribution of points in a point cloud according to its spatial information.



b) The 2D range image representation of Point Cloud

Figure 3-3 Point Cloud Representation

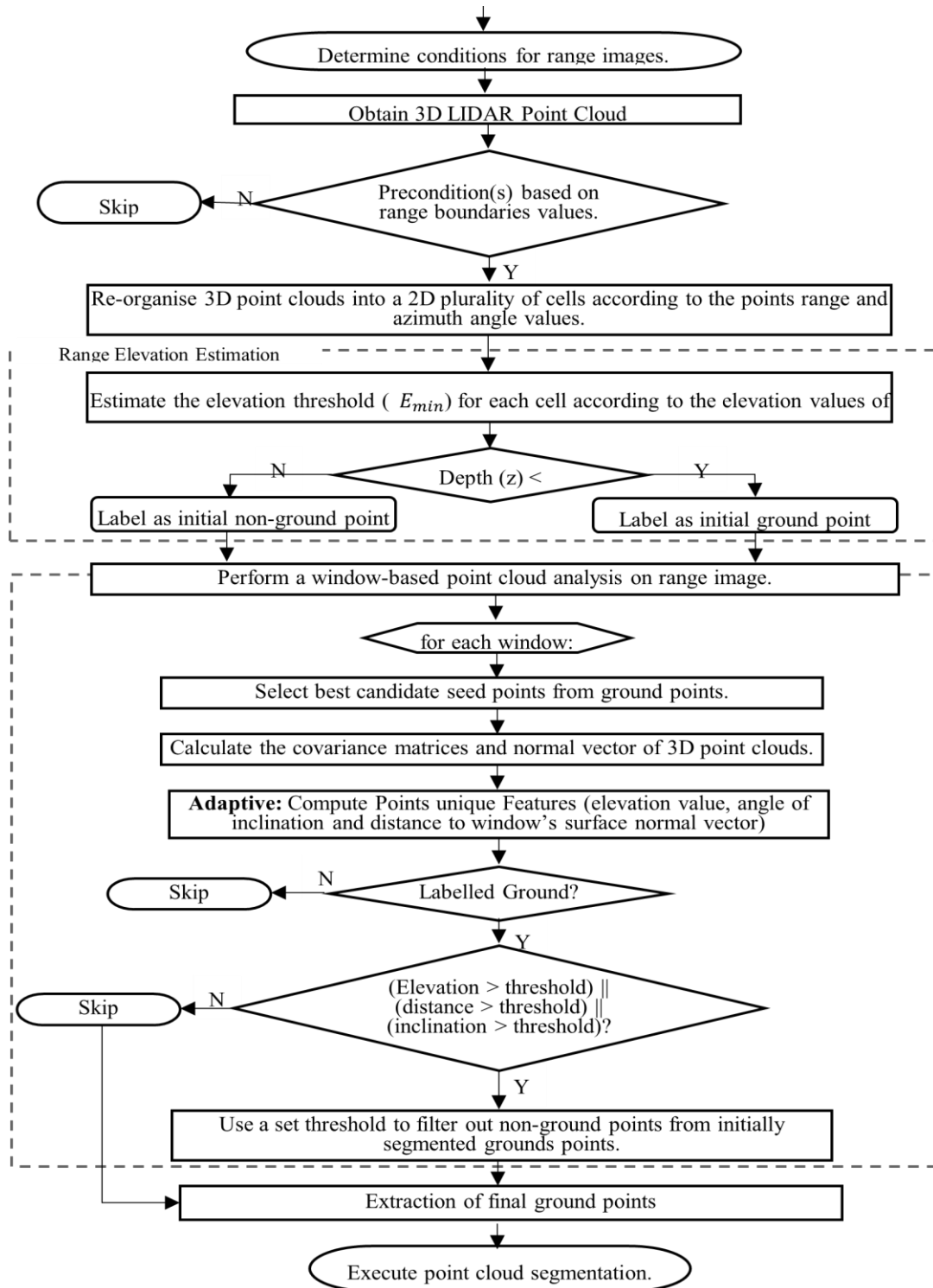


Figure 3-4 Flowchart of the FastSeg3D Algorithm

Figure 3-4 shows the complete flowchart of the FastSeg3D algorithm, which is an expansion of the block diagram shown in Figure 3-2. Figure 3-3 shows the point cloud representation of the raw captured point cloud.

Since the 2D range image representation technique has demonstrated its performance, effectiveness in point distribution over varying distances and inexpensive computational cost for point cloud representation, like the ring elevation map [227] and the polar grid technique [79], the FastSeg3D employs similar representation method to achieving a fast means of organising the raw captured point clouds. The range values r_i and azimuthal angle θ_i (horizontal angles) of each point p_i in the point cloud, P is used to represent all points as a 2D image $P_{(r,\theta)}$.

To achieve this, an empty placeholder 2D image $P_{(r,\theta)}$ is created and also a data structure $\{ I_{(r,\theta)}, L_{(r,\theta)}, E_{(r,\theta)}, d_{(r,\theta)}, \alpha_{(r,\theta)} \}$ is created for fast lookup. Where the attributes in the data structure are the unique features of every point in the point cloud; cloud Index $I_{(r,a)}$, Point Labels $L_{(r,a)}$, Elevation Value $E_{(r,a)}$, Distance of point to origin $d_{(r,a)}$, and Inclination angle $\alpha_{(r,a)}$. The range value and azimuthal angle values for a given point p_i in the point cloud, P can be expressed as:

$$r_i = \sqrt{x_i^2 + y_i^2} \quad (3.3)$$

$$\theta_i = \text{floor} \left(\frac{\theta_{degree}}{360} \right) \quad (3.4)$$

Where, θ_{degree}

$$= \begin{cases} 0, & \text{if } y_{(r,\theta)} = 0, \text{ and } x_{(r,\theta)} = 0 \\ \text{atan2}(y_{(r,\theta)}, x_{(r,\theta)}), & \text{if } y_{(r,\theta)} \geq 0 \\ \text{atan2}(y_{(r,\theta)}, x_{(r,\theta)}) + 2\pi, & \text{if } y_{(r,\theta)} < 0 \end{cases} \quad (3.5)$$

The maximum row range of the 2D image $P_{(r,\alpha)}$ is determined by the approximate maximum range of accurate measurements from most common LiDAR systems like the Velodyne 32E, and 64E, used by autonomous robots. This value is usually around 70 meters, therefore the range of values for the row of the 2D image, which represents the range values of each point in the point cloud, is set to 0 – 70 meters.

The maximum column range of the 2D image $P_{(r,\theta)}$ is determined by using the rated angular resolution (horizontal/Azimuth) of the LiDAR datasheet, which is between the range of 0.1^0 to 0.4^0 . Since the LiDAR sensor/system performs a 360^0 scan, it is divided by the resolution to estimate the maximum column range. That is $\frac{360^0}{0.4^0}$ which gives a value of 900. Now the best case which is $\frac{360^0}{0.1^0}$ gives a value of 3600 but this is only possible if the LiDAR system is run at a very low RPM of about 300 which is not ideal for capturing point clouds in real time. Most LiDAR systems used for navigational purposes are configured to run at maximum speed (approximately 20Hz) with an RPM of about 1200, which accounts for the angular resolution of 0.4^0 . With a calculated value of 900, the formula 2^n is used to find the closest value to but greater than 900 to ensure that the image representation of the point cloud conforms with the standard computer vision resolution for images. Therefore, the maximum column value is set to 1024 for 32-beam laser LiDAR and 2048 for 64-beam laser LiDAR systems.

Every point in the point cloud is allocated to a cell/pixel in the 2D image once its calculated values fall within the 2D image range image. Furthermore, points that fall within the range of 0 – 2 meters are filtered out to account for points that fall on the ego vehicle during the LiDAR system scan. This condition can be expressed as follows:

$$p_{(r,\theta)} \in cell_{(i,j)} \begin{cases} R_i \leq R_{(r,\theta)} < R_{i+1}, & \text{where } i \in \{0, I-1\} \\ i = \Theta_{norm} = \text{floor}\left(\frac{\Theta_{degree}}{360}\right), & \text{where } i \in \{0, I\} \end{cases} \quad (3.6)$$

3.3.3 Algorithm 1: Range elevation estimation (REE)

After allocating points to the 2D range image, the minimum elevation values and the elevation thresholds of each cell/pixel in the point cloud are estimated and used to determine the ground and non-ground points from all valid points in the 2D range image. Traversing across adjacent points i.e. across the columns in the 2D range image, the minimum elevation, E_{min} is determined by comparing the third spatial coordinate value of the previous point, $z_{(i-1,j)}$ and the current point, $z_{(i,j)}$ to find the minimum value. This value is set as the minimum value for the current cell or pixel. The algorithm is shown in Table 3-1.

Table 3-1 Algorithm 1: Coarse segmentation of point cloud using REE technique

Algorithm 1: Coarse Segmentation (REE)	
Input: Raw Point Cloud	
Output: Labelled Point Cloud	
Params:	
1	E_t : elevation tolerance
2	for each $p_{(r,a)}$ in Cloud do
3	$p_{(r,a)} \in cell_{(i,j)}$;
4	$E_{(i,j)} = z_{(r,a)}$;
5	$label_{(r,a)} = ground$;
6	for each E_{min} in $E_{(i,j)}$ do
7	$E_{min(i,j)} = \min(E_{(i,j)}, E_{(i-1,j)} + E_t)$;
8	for each $p_{(r,a)}$ in Cloud do
9	$p_{(r,a)} \in cell_{(i,j)}$;
10	if $z_{(r,a)} \geq E_{min(i,j)} + E_t$
11	$label_{(r,a)} = non_ground$;

The elevation values are calculated using the $z_{(r,\theta)}$ coordinate of each point in the point cloud and an elevation tolerance E_t of 0.2 meters along the row which is also called adjacent points $E_{(i-1,j)}, E_{(i,j)}$ of the range image. The elevation tolerance E_t is allow enough tolerance to accommodate all possible ground points during the first segmentation

process. For the condition $r = 0$, where the row number is zero, the elevation value is the same as the minimum third spatial coordinate of the point(s) in the current cell/pixel. This can be expressed as:

$$E_{\min(i,j)} = \begin{cases} E_{(i,j)}, & P_{(r,\theta)}, \text{ for } r = 0 \\ E_{(i-1,j)} + E_t, & \text{otherwise} \end{cases} \quad (3.7)$$

Using the set elevation threshold E_t and the estimated minimum elevation for each cell $E_{\min(i,j)}$, a threshold value E_{th} is set for each cell and used to segment points in that cell as a ground or non-ground point. Therefore, points equal to or above the threshold value E_{th} are labelled as non-ground points, and points below as ground points. This is expressed as:

$$E_{th} = E_{\min(i,j)} + E_t \quad (3.8)$$

3.3.4 Algorithm 2: Window-based model fitting (WBMF)

The REE coarse segmentation techniques result in a high level of over-segmentation of ground points. This is understandable because all valid points within the range image are labelled as ground points before the elevation threshold E_{th} is used to segment non-ground points. To address this the window-based model fitting technique is employed to adaptively calculate the unique spatial and geometric features of each point in a predefined region/window size which is a subsection of the entire range image. As shown in Figure 3-5 and Table 3-2. This window is traversed across the column and the spatial and geometric features for each point within the current window are re-calculated. These values are updated in the data structure and used to perform a fine-grain segmentation of the initially segmented points in the point cloud.

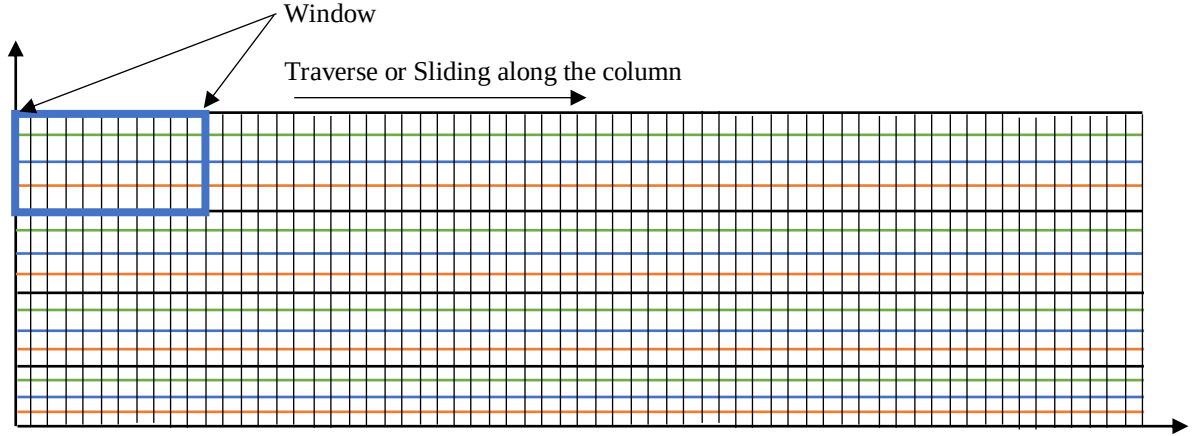


Figure 3-5 The window-based model fitting technique

The 2D image is divided into smaller regions by calculating the subset region dimension. This subset is set to a dimension of $(\frac{R_w}{16}, \frac{R_h}{5})$, where R_w is the range image width and R_h is the range image height. This dimension is depicted as (w, c) which means (window row, window column). This is also referred to as the window dimension. This fixed dimension has the condition that w is less than or equal to r and r divides w without a remainder. The same applies to c and a . This can be expressed as:

$$(w, c) = \left(\frac{R_w}{16}, \frac{R_h}{5} \right) \quad (3.9)$$

Where the following conditions: $w \leq r$, $w \% r = 0$ and $c \leq \theta$, $c \% \theta = 0$ must be met.

Using the initial ground points as candidate points i.e. G^k , in a sliding window manner, as shown in Figure 3-5, The best seed points are determined from initial ground points in the current window. This is selected based on the first k set number of initial ground points with the lowest elevation value $E_{(w,c)}^i$ in the current window which is indicated by superscript i . Where k is a minimum of 5 and a maximum of 20 points in the current window, i . This is expressed as:

$$G^k = \operatorname{argmin}(E_{(w,c)}^i) \quad k = \{0,1,2, \dots, K\} \text{ and } 5 \leq K \leq 20. \quad (3.10)$$

Using the best seed points G^k For the current window, the covariance matrix M is calculated and the eigenvectors (v_1, v_2, v_3) and eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of the covariance matrix M are calculated. M , eigenvectors, and eigenvalues are calculated using the standard covariance matrix formula for principal component analysis (PCA).

A plane model M_w is computed. This model is developed using the normal vector $\vec{n}_w$ to the plane equation. The normal vector $\vec{n}_w$ of the plane model is a, b, c. This can be expressed as:

$$M_w = ax + by + cz + d = 0 \quad (3.11)$$

$$d = -(ax_0 + by_0 + cz_0) \quad (3.12)$$

The normal vector of the plane model is used to compute unique features, adaptively, for all points in the current window. This feature includes the distance for points from the normal vector ($d_{(w,c)}$), the elevation of point reference to normal vector ($(\overrightarrow{E_{(w,c)}})$), and the angle of inclination ($(\overrightarrow{\alpha_{(w,c)}})$) of point reference to the normal vector.). $d_{(w,c)}$ is the distance for points in the current window to the plane origin or normal vector distance d . The normal vector distance d , as expressed in equation 3.12 is the point of origin for the current window. The new values are updated to the data structure of the point cloud therefore the (w, c) coordinates would match the (r, θ) coordinate of the data structure. These values are expressed as:

$$d_{(w,c)} = \frac{\operatorname{abs}(a \times x_{(w,c)} + b \times y_{(w,c)} + c \times z_{(w,c)} + d)}{\sqrt{a^2 + b^2 + c^2}} \quad (3.13)$$

$$\overrightarrow{E_{(w,c)}} = z_{(w,c)} \times c \quad (3.14)$$

$$\overrightarrow{\alpha_{(w,c)}} = \text{atan} \left(\frac{E_{(w,c)}}{d_{(w,c)}} \right) \quad (3.15)$$

A set threshold is used to filter out non-ground points from initially segmented ground points. Thresholds are set based on the common estimated values considered to be associated with ground points. These threshold values are:

- d_{th} which is set as 1 meter. A point of distance less than or equal to 3 meters is generally regarded as a ground point, i.e. $0 \leq d_{th} \leq 3$ meters.
- α_{th} : A slope or inclination angle close to 0° is considered as ground. The threshold is set to 1° . I.e $0^\circ \leq \alpha_{th} \leq 5^\circ$.
- E_{th} : Usually points less than 0.3 meters are considered as ground points. The threshold is set to 0.2m. I.e $0 \leq E_{th} \leq 0.3$ meters.

Table 3-2 Window-based model fitting (WBMF) algorithm for fine-grain segmentation

Algorithm 2: Window-based Model Fitting (WBMF).

Input: REE image

Output: Labelled Point Cloud

Params: NOTE:

$label_{(r,\theta)}$ 0 is ground, 1 is non ground, and 2 is invalid.

W: Window size

1 $P_{(M \times N)}$: REE image

2 j : Number of points per window

3 α_{th} : Slope threshold

4 W, C: Windows dimension.

5 d_{th} : Distance threshold

6 E_{th} : Elevation threshold

7 G^k : Candidate ground points, k is the total candidate points

8 for $W_{(w,c)}$

9 for each $W_{(w,c)}$ in $P_{(M \times N)}$ do

10 if $j \geq k$ do

11 computer M_w , $d_{(w,c)}$, $E_{(w,c)}$, and $\alpha_{(w,c)}$;

12 for $p_{(r,\theta)}$ in $W_{(w,c)}$ do

```

13         if  $label_{(r,a)}$  is 0,  $E_{(w,c)} > E_{th}$ , or  $d_{(w,c)} >$ 
14  $d_{th}$ , or  $\alpha_{(w,c)} > \alpha_{th}$ 
15              $label_{(r,a)} = 1$ ;
16     else
            add  $W_{(w,c)}$  to  $W_{(w+1,c)}$ 

```

3.4 CONCLUSION

In this chapter, a Fast and Adaptive Ground Segmentation Method of 3D Point Cloud, FastSeg3D algorithm was developed. This algorithm begins by converting a raw input 3D point cloud into its equivalent 2D range image by deriving the range information and azimuth angle values of each point from each point in the given 3D point cloud. A data structure is also created to store relevant information for segmentation and the range image is used to perform a quick estimation of ground and non-ground points using a set threshold and establishing a similarity between adjacent points in the range image. The initially estimated points are then used as candidate points to perform a fine-grain segmentation of the given point cloud to address the over-segmentation of ground points using a window-base model fitting technique.

CHAPTER 4: THE CAMERA MODEL AND AUTOMATIC EXTRINSIC CALIBRATION

4.1 INTRODUCTION

Sensor calibration is the process of determining the intrinsic and extrinsic parameters of a sensor. Sensor calibration parameters are crucial for understanding how a sensor derives its measurement or captured values relative to the actual data i.e. the real-world coordinates [247,248]. This chapter presents the process of establishing sensor calibration parameters and proposes an extrinsic calibration process for the sensor fusion pipeline presented in chapter five of this report.

4.2 INTRINSIC AND EXTRINSIC PARAMETERS

Intrinsic calibration refers to the process of determining the internal parameters of a sensor that affect its measurements. These parameters are specific to the sensor itself and do not change as the sensor's position or orientation changes. For instance, for image sensors (cameras), intrinsic calibration involves determining parameters like focal length, principal point, lens distortion coefficients, and sensor resolution. These parameters are necessary to establish a relationship between the 2D pixel coordinates in the pixel frame of the captured image and the image sensors' 3D coordinates in the camera frame.

For LiDAR sensors, the intrinsic parameters involve determining the laser beam divergence angle, laser pulse duration, laser wavelength, and detector sensitivity. These parameters are usually calibrated by the manufacturer before the sensor gets to the customer [249]. The intrinsic parameters like the pulse width, pulse repetition frequency, antenna gain, beamwidth, and transmitter power, for the RaDAR sensors, are also calibrated by the manufacturer before the sensor gets to the customer [250,251]. This intrinsic calibration process is likewise applicable to ultrasonic sensors, which possess intrinsic parameters including pulse width, pulse repetition frequency, antenna gain, beamwidth, and transmitter

power [252]. Sensors with already calibrated intrinsic values do not usually require recalibration when used alone unless they are used in combination with other sensors.

In contrast, extrinsic calibration concerns the determination of the transformation between the local coordinate system of the sensor and the global coordinate system. This entails finding the real-world position and orientation (pose) of the sensor. Extrinsic calibration becomes crucial when integrating data from multiple sensors or when establishing a connection between the sensor's measurements and a shared reference frame. The extrinsic calibration of multiple sensors is usually prone to errors during the fusion process. The most common is data misregistration between sensors and the reprojection error that occurs when projecting a sensor's information into the coordinate frame of another sensor [143,152,157]. To reduce these errors many extrinsic calibration methods have been proposed, which were extensively discussed in section 2.3 of chapter two of this report. The following subsections will explain the process of establishing calibration for an image sensor and propose a lidar-camera automatic calibration technique for BEVMSF3D. It will also propose an automatic calibration technique for the sensor fusion pipeline.

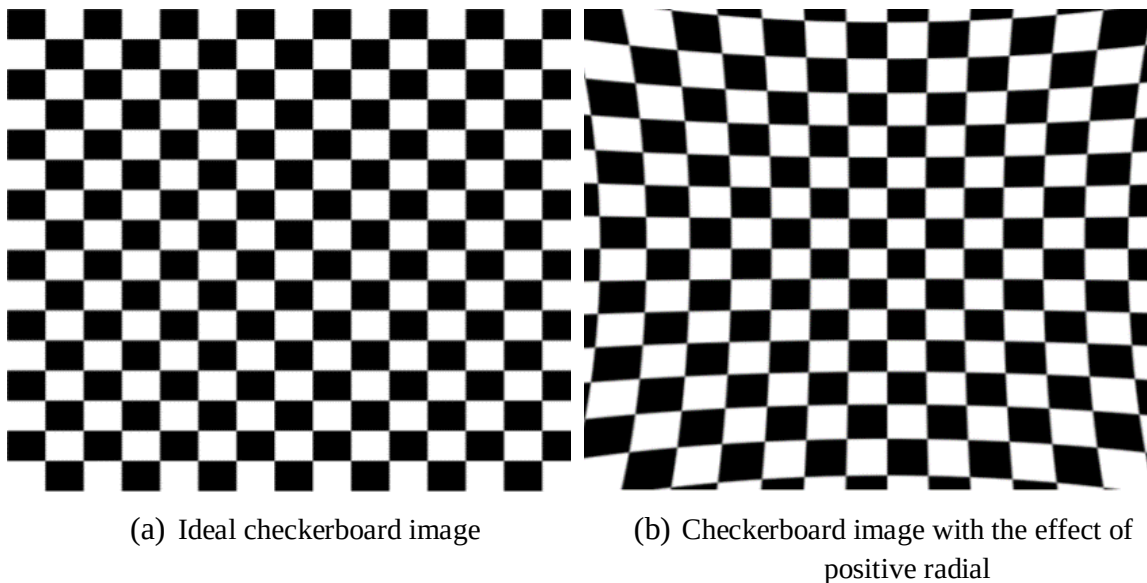
4.2.1 Image sensors

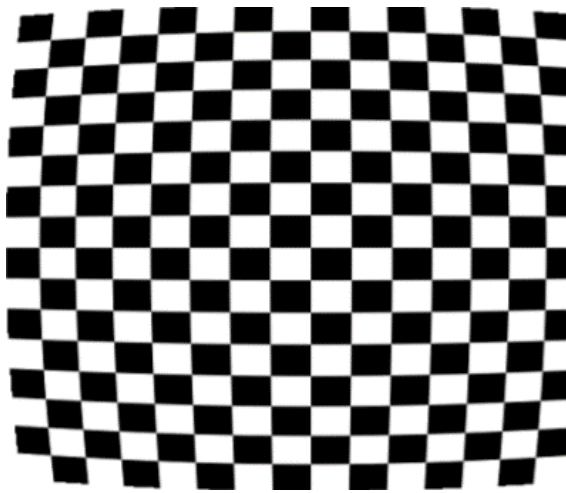
To determine the calibration parameters of an image, several methods can be employed, and these methods can be categorised into four main approaches: checkerboard, pattern-based, self-calibration, and multi-camera calibration. The process of calibration relies on an understanding of specific spatial positions and orientations of objects in the real 3D world. This understanding is then used to establish a relationship by calculating the calibration parameters between the images of the 3D objects and the known geometry.

Checkerboard calibration is performed by taking pictures of a checkerboard pattern, which has clear edges and corners, from various angles. By detecting these corners, the intrinsic and extrinsic parameters of the camera can be estimated. The commonly used algorithms for checkerboard calibration include Zhang's method [253] and Bouguet's method [254]. Multi-camera calibration becomes relevant when multiple cameras capture images of a

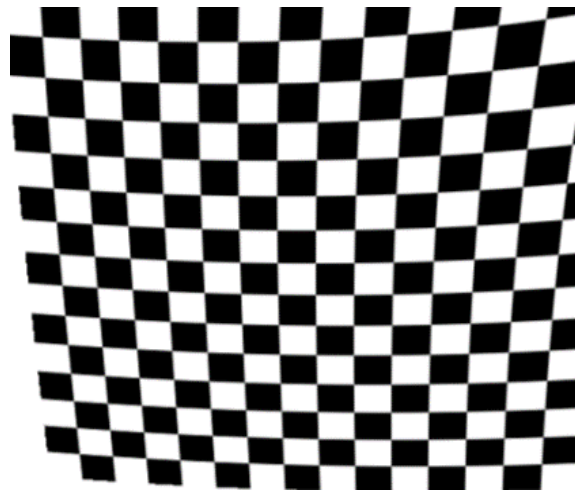
scene with a predetermined geometry. Detected features from these images are then utilised to estimate the intrinsic and extrinsic parameters for each camera. An example is a stereo camera that consists of a left-side and a right-side camera. For multi-camera calibration, Zhang's method and Bouguet's method are also applied to achieve accurate results.

Pattern-based calibration shares similarities with checkerboard calibration. It entails capturing images of recognisable patterns, such as circles, spheres, and grids which are also called squared patterns and ECochecks. Figure 4-2a shows an example of an ECocheck mixed with a circle-based pattern for robust calibration. These patterns are utilised to identify features for estimating the intrinsic and extrinsic parameters of the camera. The widely employed algorithm for pattern-based calibration is Tsai's method [255]. Another algorithm also used is the Heikkila-Jain's method [256].

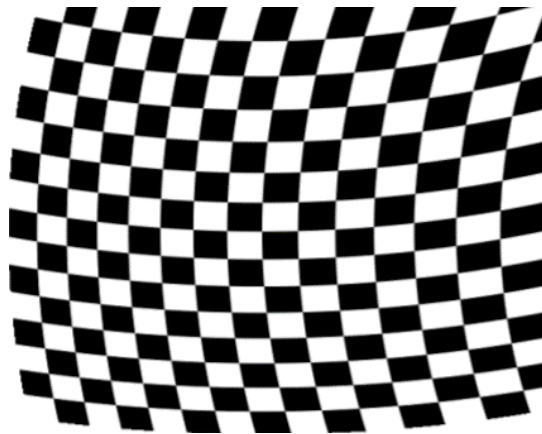




(c) Checkerboard image with the effect of negative radial



(d) Checkerboard image with the effect of decentering



(e) Checkerboard image with the effect of prismatic distortions

(f)

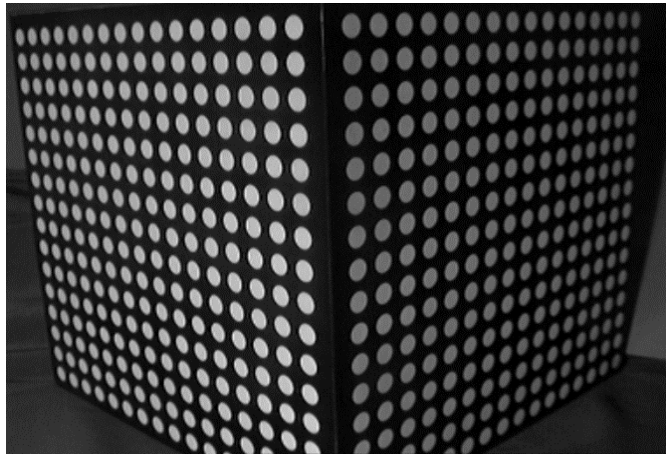
Figure 4-1 Checkerboard Images with different possible kinds of lens distortions [253]



(a) 2D Pattern-based (circle and ECochecks) image for pattern-based calibration.



(b)



(c) 3D Pattern-based image for pattern-based calibration.

Figure 4-2 Pattern-based image, undistorted [153]

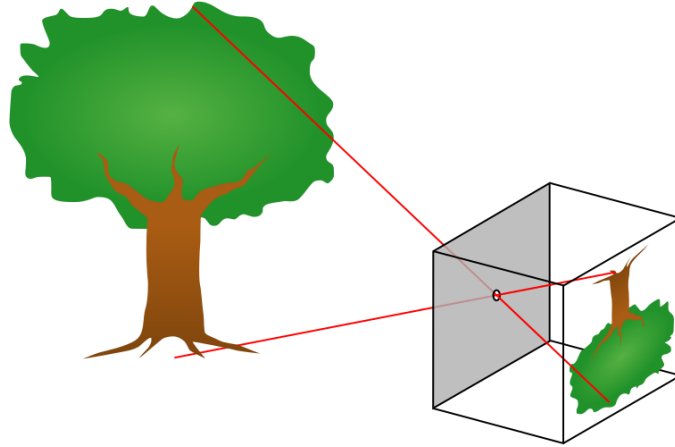
Self-calibration involves using a single camera to capture images of a scene with a known geometry, often a flat surface. By detecting features in these images, the intrinsic and extrinsic parameters of the camera can be estimated [257]. A common algorithm for self-calibration is the Faugeras self-calibration [258,259].

4.2.1.1 Camera modelling

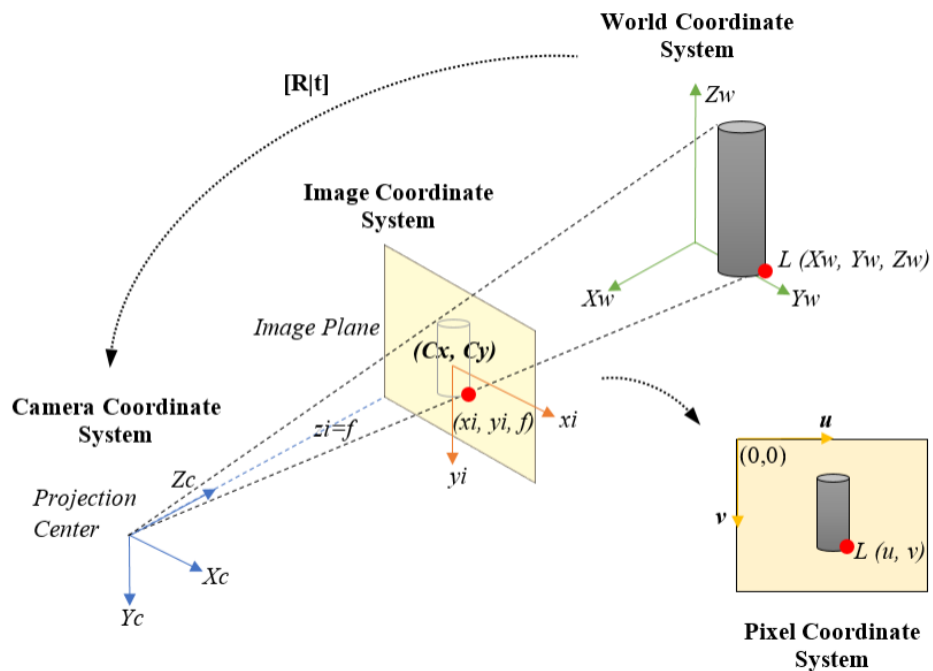
The pinhole camera concept is a simplified model used in computer vision and photography to understand how a camera captures images. This conceptual model operates under the assumption that light travels in straight lines and enters the camera through a small hole, forming an inverted image on the image plane. In this model, the camera aperture is represented as a point, and no lenses are used to focus light. However, the pinhole camera model does not consider geometric distortions, blurring caused by lenses, or discrete image coordinates. It serves as a first-order approximation of the mapping from a 3D scene to a 2D image, with its validity depending on the camera's quality.

The pinhole camera model assumes that the image plane, which captures the light rays, is situated in front of the pinhole, although in reality, it is located behind the pinhole. It also assumes that all light rays from different points converge at the pinhole, also known as the

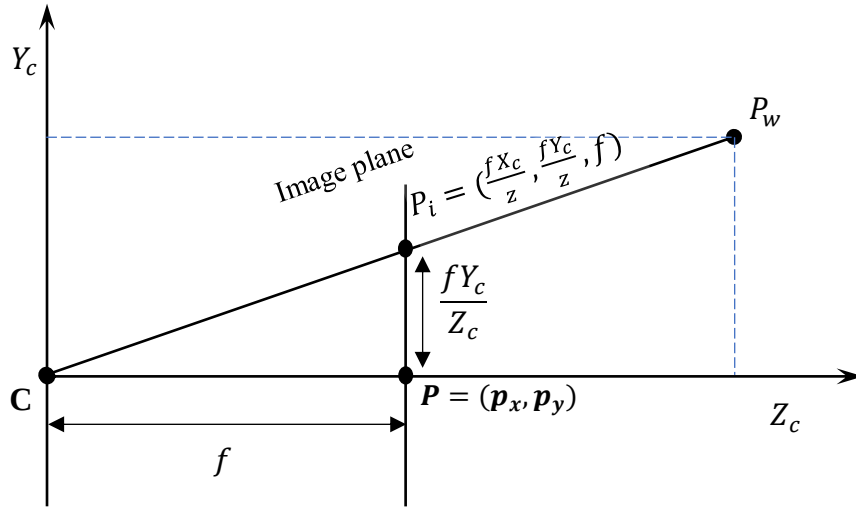
centre of projection or the camera centre. The image of a point is the projection of that point on the image plane, where the line from the camera centre to the point intersects the image plane. Finally, it assumes that the camera, points, and objects are located in the world space [260].



(a) The Pinhole camera concept [261]



(b) Pinhole Camera Modelling: ideal projection of a 3D object on a 2D image [262]



(c) Intrinsic Parameters: Camera-to-image transformation, also called projecting 3D points from the camera onto the image plane

Figure 4-3 Camera modelling

To capture a scene from the 3D world coordinate system to the 2D image coordinate, the extrinsic parameters i.e. the information about the location of the camera in the 3D world coordinate are used to transform the scene onto the camera's frame. After this, the intrinsic parameters are used to transform the 3D scene in the camera's frame into the image coordinate. The general camera model equation can be expressed as:

$$P_i = K \times P_c = K \times [R \mid t] \times P_w \quad (4.1)$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} fX_c \\ Z_c \\ fY_c \\ Z_c \\ w \end{bmatrix} = \begin{bmatrix} f & s & p_x \\ 0 & af & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} f & s & p_x \\ 0 & af & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (4.2)$$

Where:

- w : is the scaling factor.
- P_i : Homogeneous coordinates of a point, P in the image, a (3 x 1) matrix.

- P_c : Homogeneous Coordinates of a point, P in the world with respect to (wrt) the (3D) camera coordinate system. It is a (4×1) matrix.
- P_w : Homogeneous Coordinates of a point, P in the (3D) world coordinate system. It is a (4×1) matrix.
- K : The camera intrinsic matrix which can also be called the camera matrix. It is a (3×3) matrix.
- $[R \mid t]$: The camera extrinsic matrix, a (4×4) matrix.
- R : The camera rotation matrix, a (3×3) matrix.
- t : The camera translation matrix, a (3×1) matrix.
- f : focal length. Also referred to as f_x by the basic pinhole model. See Figure 3-3c.
- (u, v) : The Euclidean coordinates of the point in the image or the pixel location.
- s : the skew factor. This is very close to zero for digital cameras.
- a : the aspect ratio. af is often referred to as f_y by the basic pinhole model. This is usually set to 1.
- p_x and p_y : are the principal points also called the camera offset.
- X_c, Y_c, Z_c : are the coordinates of a point, P in the world with respect to (wrt) the (3D) camera coordinate system.
- r_{11} to r_{33} : is the rotation matrix R .
- t_x, t_y, t_z : is the translation matrix.

The modelling of a camera is based on the pinhole camera concept, as shown in Figure 4-3 and Figure 4-4. This model contains a set of parameters that transform homogeneous 3D world coordinates P_w to homogeneous 2D image coordinates P_i , see equations 4.1 and 4.2. The calibration techniques seek to estimate ten to eleven unknowns. For digital cameras, the skew factor, s can be approximated as zero. These unknowns consist of four to five intrinsic unknowns, three rotations and three translation extrinsic parameters relative to the world's coordinate. The focal length, f is the distance between the camera coordinates and the image plane, measured in pixels. It determines the magnification and angle of view of the image plane. The focal length, (f_x, f_y) relates according to the camera's aspect ratio,

$a = \frac{f_y}{f_x}$. This is usually set to one (1) and it is the relationship between the width and the height of the image plane [263]. The principal point, (p_x, p_y) describes where the point is in the image plane. It is the pixel in the image plane through which the optical axis of the camera passes through.

To account for radial and tangential lens distortion, the radial coefficients k_1, k_2, k_3 and the tangential coefficients p_1, p_2 are introduced and it is expressed mathematically as:

$$\begin{bmatrix} x_{corrected} \\ y_{corrected} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix} \quad (4.3)$$

Where: $r = \sqrt{x^2 + y^2}$, (x, y) are the old pixel coordinates of the point in the image, and $(x_{corrected}, y_{corrected})$ are the corrected pixel coordinates.

The difference between different camera models is in their methodologies for deriving the intrinsic parameters within the intrinsic matrix, denoted as K . These differences pertain to whether the model accounts for lens distortion and the specific approach adopted to rectify this distortion. The common distortions are shown in Figure 4-1. The radial distortion manifests in the form of the ‘barrel’ or ‘fish-eye’ effect, while the decentering distortion can also be referred to as tangential distortion occurs when the lenses are not perfectly aligned with the imaging plane. Prominent methodologies for estimating camera calibration parameters include Tsai's, Zhang's, Heikkila's, and Bouguet's algorithms, all designed to address common lens distortions, i.e. radial and/or tangential.

For instance, Tsai's calibration algorithm [255] is based on the pinhole model and operates under the assumption of first-order radial distortion in the camera. Under this premise, image pixels experience distortion solely along the radial direction, allowing the employment of the radial alignment constraint (RAC) model to linearise the calibration process.

Zhang's calibration algorithm [253] requires the camera's observation of a planar pattern from multiple (at least two) orientations. This algorithm models radial lens distortion. The estimation of calibration parameters involves a closed-form linear solution through homography, followed by a nonlinear optimisation technique utilising the maximum likelihood criteria.

Heikkila's calibration algorithm [256,264] also adopts the pinhole camera model and accommodates both radial and tangential lens distortions. Initial parameter estimation relies on the direct linear transformation (DLT), disregarding lens distortions. Parameter refinement occurs through nonlinear estimation, considering both noise and lens distortions. The ultimate parameters are derived by minimising the total squared error between the model and observed data.

Bouguet's approach [254], as implemented in the CalTech Calibration Toolbox, builds upon Heikkila's intrinsic camera model. In the initialisation phase, homography is computed for all points across a series of images, and intrinsic parameters are set to 0 to initialise distortion. Subsequently, extrinsic parameters are established for each pattern image. The final nonlinear optimisation, akin to Zhang's method, integrates Maximum Likelihood estimation, complemented by tangential distortion coefficients.

Decentering distortion, originally modelled by Conrady [265], was later redefined by Brown [266] shortly after proposing a camera model for radial distortion [267]. The primary objective of calibration involves determining external parameters like position and orientation relative to a world coordinate system and internal camera parameters like the principal point or image centre, focal length, and distortion coefficients. While some manufacturers pre-specify intrinsic camera parameters, they can also be ascertained through well-established calibration techniques. Since the introduction of pinhole and Brown camera models [267], numerous calibration techniques have been introduced [253,255,256,258,259,264,268,269] to address diverse lens distortions. To explain camera calibration techniques, this discussion will underscore a set of general prerequisites and

expound upon the Direct Linear Transform (DLT) [270] calibration estimation method. Furthermore, the most common lens distortion like the radial and tangential lens distortion equation for correction will also be presented. No matter the approach used for calibration the target pattern objects must have sharp, detectable edges and corners.

To estimate the calibration parameters of a camera the following requirements are needed:

- The target object should consist of two orthogonal planes, for Tsai's calibration method i.e. 3D objects, see Figure 4-2b. For Zhang's calibration method, the target object should consist of a 2D target object.
- There should be prior knowledge of the 3D coordinates, size and shape of the target object.
- For DLT, at least six (6) control points on a 3D calibration target object and their corresponding 2D coordinates in the image should be known. It is however recommended to have above 20 control points.

4.2.1.2 Direct Linear Transform (DLT)

To determine the internal and external parameters of the camera the direct linear transform (DLT) algorithm can be used. This is represented as a homogeneous linear equation similar to Equation 4.2. The idea of DLT is to represent the perspective projection equation as a homogeneous linear equation and solve it by standard methods. Therefore, the pinhole camera model equation:

$$x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = KR[I_3 | -X_0]X = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (4.4)$$

Where:

- x is the observed image point
- K is the intrinsic camera matrix

- I is an Identity matrix
- R is the rotation matrix
- X_0 is the translation matrix.
- $X = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$ is the control point in the world coordinate.

Let $M = KR[I_3 | -X_0]$, therefore;

$$x = MX = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (4.5)$$

The Matrix M can be decomposed into the intrinsic and extrinsic parameters of the camera.

$$\text{Let } P = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

To decompose M to pixel coordinates would result in:

$$u = \frac{wu}{w} = \frac{m_1^T \cdot P}{m_3^T \cdot P} \text{ and } v = \frac{wv}{w} = \frac{m_2^T \cdot P}{m_3^T \cdot P} \quad (4.6)$$

$$(m_1^T - u_i m_3^T) \cdot P_n = 0 \text{ and } (m_2^T - v_i m_3^T) \cdot P_n = 0 \quad (4.7)$$

This can then be represented as:

$$\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (4.8)$$

Equation 4.7 represents the DLT for conversion of the Homogenous matrix M to pixel coordinate for a single point. A stack of n points would become:

$$\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ & \vdots & \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad (4.9)$$

This can be rewritten as $Q \cdot M = 0$, where matrix Q is the known matrix and matrix M is the unknown matrix. With more points, with a recommended value of above 20, it becomes a lot easier to solve for the unknown matrix using the least square solution, which minimises the sum of square residuals, $\|QM\|^2$, subject to the constraint $\|M\|^2 = 1$. This can be solved through Singular Value Decomposition (SVD). The solution is the eigenvector corresponding to the smallest eigenvalue of the matrix $Q^T Q$. Once M has been determined, it can be decomposed into the intrinsic and extrinsic parameters of the camera. To set the orthogonal constraint for R the QR Decomposition with Gram-Schmidt can be used. To minimise the lens distortion the non-linear optimisation algorithm, LM algorithm can be used, this is called minimising the reprojection error. The above mathematical expression does not model the lens distortion.

4.3 AUTOMATIC EXTRINSIC CALIBRATION

For this sensor fusion pipeline, the automatic extrinsic calibration problem is modelled as a unified bird's eye view (BEV) problem as presented in section 2.4.4 of chapter two of this report. To achieve this the following subsections explain the steps taken.

4.3.1 Remove ground points

The removal of ground points is achieved using the FastSeg3D algorithm. For visualisation purposes, Figure 4-4 shows the point cloud of non-ground points only.

4.3.2 Transform object points to the image camera frame and projection to 2D BEV

Using the camera's intrinsic matrix parameters K which can be obtained using Zhang's or Tsai's calibration method, the point cloud objects P^{ob} are converted to the camera frame.

For the point cloud object P^{ob} , with the expression:

$$P^{ob} = \{\{x_1^{ob}, y_1^{ob}, z_1^{ob}, i_1^{ob}\}, \{x_2^{ob}, y_2^{ob}, z_2^{ob}, i_2^{ob}\}, \{x_3^{ob}, y_3^{ob}, z_3^{ob}, i_3^{ob}\}, \dots, \{x_n^{ob}, y_n^{ob}, z_n^{ob}, i_n^{ob}\}\}.$$

The equivalent transformed point cloud P^{cam} can be expressed as:

$$\begin{aligned} P^{cam} &= K \times P^{obj} \quad \text{for } P^{obj} \\ &= \{x_1^{ob}, y_1^{ob}, z_1^{ob}\}, \{x_2^{ob}, y_2^{ob}, z_2^{ob}\}, \dots, \{x_n^{ob}, y_n^{ob}, z_n^{ob}\} \end{aligned} \quad (4.10)$$

$$\begin{aligned} P^{cam} \\ &= \begin{bmatrix} x_1^{norm} & y_1^{norm} & z_1 & i_1 \\ x_2^{norm} & y_1^{norm} & z_2 & i_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^{norm} & y_1^{norm} & z_n & i_n \end{bmatrix} \end{aligned} \quad (4.11)$$

The x and y spatial coordinates of the point cloud objects transformed to the camera frame are normalised to account for possible large focal lengths and principal point values. Since the third row of the camera intrinsic matrix is homogenous i.e. $[0 \ 0 \ 1]$, the z-axis is multiplied by 1 and would not need normalisation. Furthermore, the intensity is concatenated and so does not need to be normalised.

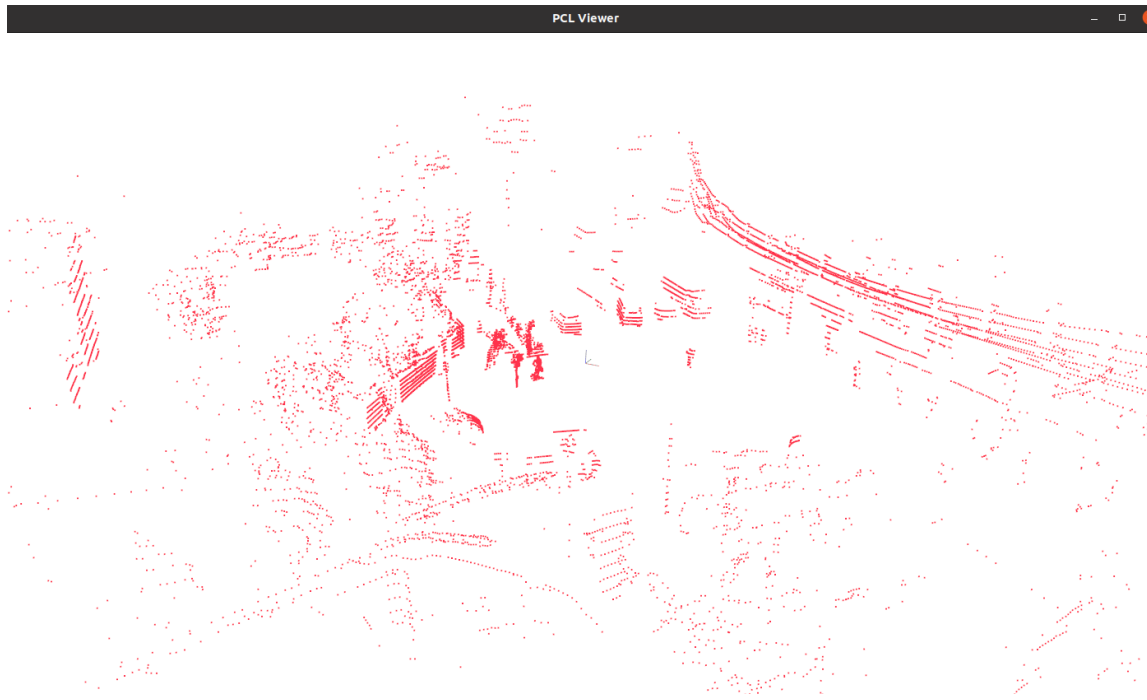


Figure 4-4 Point cloud of object points only after FastSeg3D

With the point cloud transformed into the camera frame, both images and the point cloud are seen by the robot to be taken by the same sensor. The only difference is the extrinsic matrix, which is addressed using the image preprocessing technique (augmentation) for deep learning models. Furthermore, the need to transform point cloud objects to the camera frame would no longer be necessary once the inverse perspective mapping (IPM) for unified BEV is introduced in Chapter 5 of this report.

4.3.3 Project 3D object points in the camera frame to BEV

To address the differences in the extrinsic matrix, both the image and point cloud data need to be transformed into a unified BEV. To begin, the 3D object points in the camera frame are projected onto a 2D BEV RGB image map. This is achieved using a variant of the MV3D [271], and Complex-Yolo [272] RGB-map techniques. Inspired by Simon, *et al* (Complex-Yolo) [272], the RGB map is encoded by height, intensity and density. The modified size of the grid map is defined with $n = 512$ and $m = 512$ as compared to Complex-Yolo of $n = 1024$ and $m = 512$. This is to suit the preferred feature extractor network,

EfficientDdet [273] for this research. These 3D point clouds are then projected and discretised into a 2D grid with a resolution of 0.08m. The RGB feature channels (z_r, z_g, z_b with $z_{r,g,b} \in \mathbb{R}^{m \times n}$) are calculated for the input point cloud $P \in \mathbb{R}^3$ of the captured environment. This can be expressed as:

$$P = \{[x, y, z]^T \text{ for } x \text{ and } y \in [-70m, 70m], z \in [-2.73m, 3.27m]\} \quad (4.12)$$

The x and y are set to a 70m radius just as it was for the FastSeg3D, and the z-axis is set at a total height is 6m to account for most objects in the environment, including common buildings, z – axis above 6m are capped at 6m.

Therefore, the equivalent pixel values for the RGB-map image are calculated as follows:

$$z_g = \max(P_{i \rightarrow j} \cdot [0, 0, 1]^T) \quad (4.13)$$

$$z_b = \max(I(P_{i \rightarrow j})) \quad (4.14)$$

$$z_r = \min(1.0, \frac{\log(N + 1)}{64}) \text{ for } N = |P_{i \rightarrow j}| \quad (4.15)$$

where:

$P_{i \rightarrow j}$: is the mapping function of points with index i to it corresponding cell value j

$I(P_{i \rightarrow j})$: is the intensity of points in the given point cloud.

In addition to the Complex-Yolo approach the derived z_r, z_g, z_b are multiplied by 255 and clamped using the unsigned integer (*uint8*) datatype (i.e. 0 to 255) to account for the normalisation performed during the RGB-map calculation. This is expressed as (also see Figure 4-5):

$$z_{r,g,b} = \max(0, \min(255, z_{r,g,b} \times 255)) \quad (4.16)$$

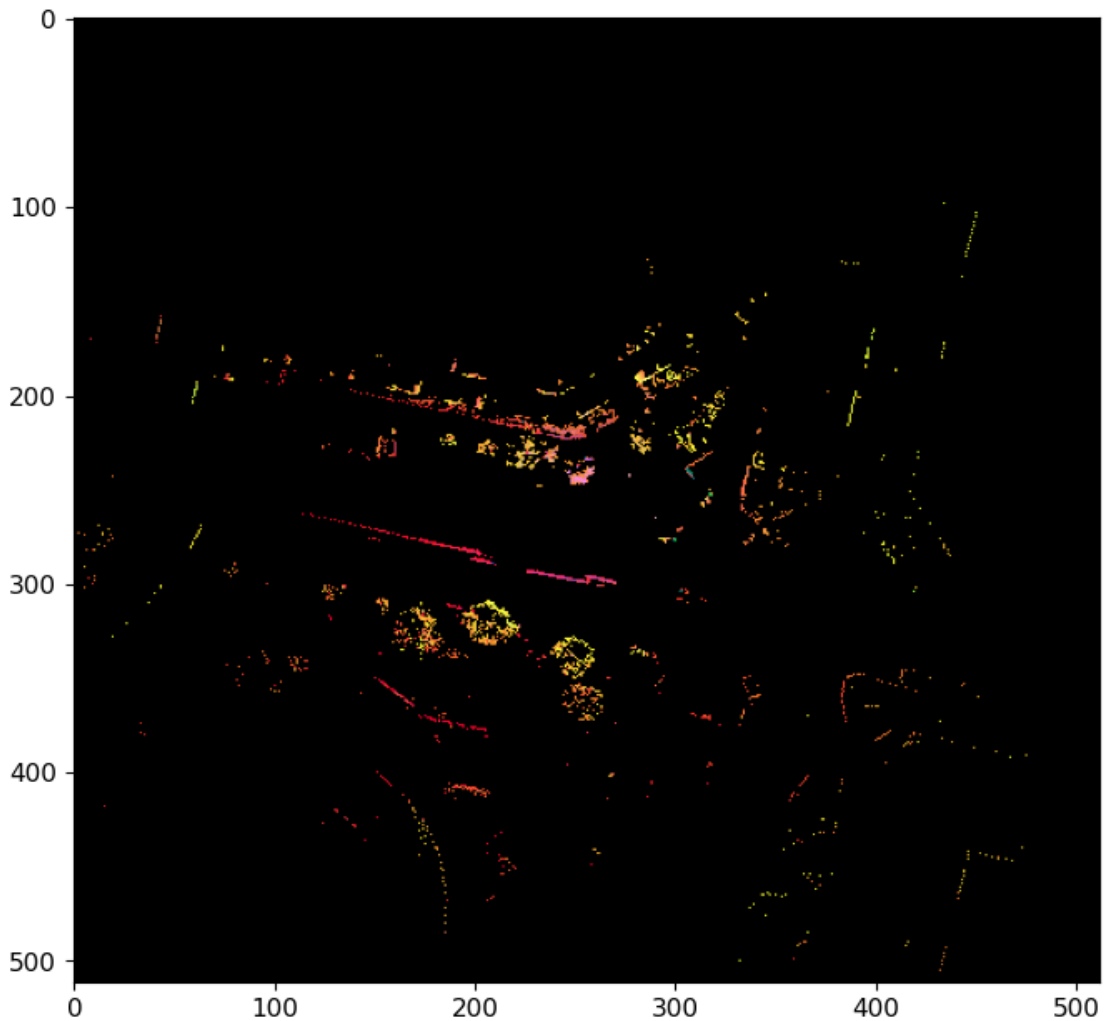


Figure 4-5 3D Point cloud of object points in camera frame projected to 2D BEV image

4.3.4 Perform image processing to make model translation and rotation invariant

The fusion pipeline, which will be discussed in detail, in Chapter 5, is a hybrid architecture that combines deterministic algorithms (FastSeg3D) with deep learning to give a robust multi-fusion pipeline. To achieve robustness for the architecture, an image preprocessing step is necessary as discussed in section 2.2.2 of Chapter 2. This is crucial for achieving automatic extrinsic calibration.

Deep learning models, like the convolutional neural network (CNN) models by default are not translation and rotation invariant. Since the filter slides from left to right and top to

bottom across an input image and activates when it comes across a particular edge-like region, corner, or colour blob, the pooling technique over spatial position helps to make it moderately translation invariant [274]. To further improve the translation invariant capability of CNN models it has to learn translations as highlighted by Biscione, *et al* [275].

With the help of image augmentation techniques, CNN models can learn to be translation and rotation invariant [276]. This is crucial in making the model robust and efficient for multi-sensor fusion. To achieve this the following algorithm is implemented as presented in Table 4-1.

Table 4-1 Image preprocessing algorithm for robust rotation and translation invariant sensor fusion network model

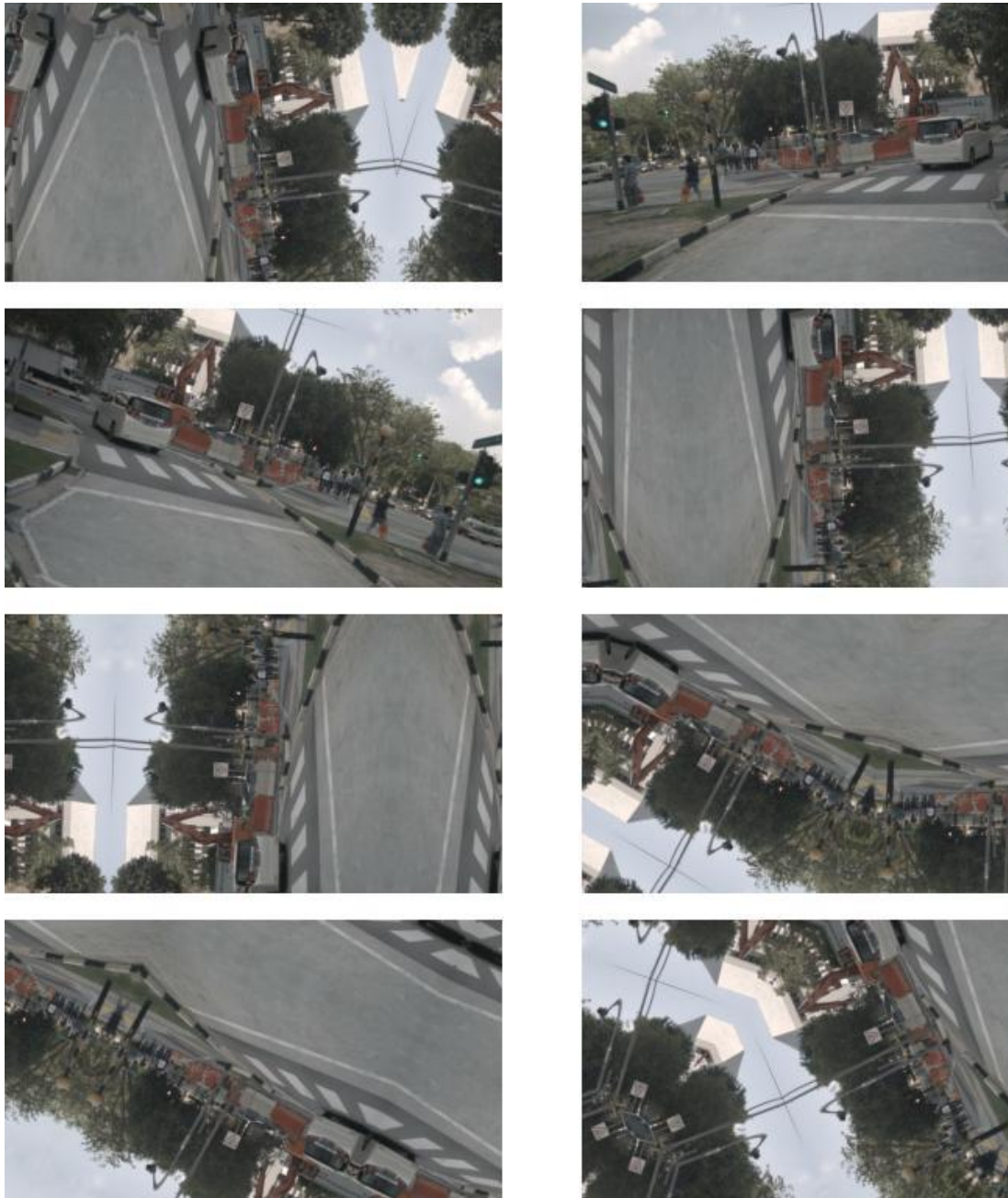
Image Augmentation Algorithm	
Input: Camera Images (<i>IM</i>)	
Output: List of Camera and Augmented Images	
Params:	
1	R_{limit} : Rotation limit in degree. Maximum Degree of rotation
2	R_{prob} : Rotation Probability
3	F_{prob} : Flip Probability
4	S_{limit} : Shift limit
5	for each <i>IM</i> in the dataset do
6	Rotate ($R_{limit} = 360^0$, $R_{prob} = 0.9$);
7	Horizontal Flip ($F_{prob} = 0.5$);
8	Vertical Flip ($F_{prob} = 0.5$);
9	ShiftRotate ($S_{limit} = 0.2$, $R_{limit} = 360^0$, $R_{prob} = 0.5$);
10	Append augmented image to dataset list.

The algorithm in Table 4-1 defines the image augmentation technique for training a robust rotation and translation invariant multi-sensor fusion architecture. This algorithm is

inspired by existing studies on addressing the translation and rotation invariant problem of CNN and deep learning models using image preprocessing techniques [275,277] and was fine-tuned during training to suit the sensor fusion architecture. This algorithm implements four transformation techniques: `Rotate`, HorizontalFlip, VerticalFlip, and ShiftRotate. The Rotate transformation rotates the image by a random angle between -360 and 360 degrees and a rotation probability of 0.9. A 0.9 probability means that every time the code runs, there is a 90% chance that it will be rotated randomly. The HorizontalFlip transformation flips the image horizontally with a probability of 0.5. The VerticalFlip transformation flips the image vertically with a probability of 0.5. Finally, the ShiftRotate transformation applies random shifts with a shift limit of 0.2 and rotations to the image with a probability of 0.5. A 0.2 shift limit means that the image can be shifted up or down the height or width by up to 20% of the original image. Samples of the augmented images are presented in Figure 4-6.



(a) Original Image



(b) Transformed images via image augmentation

Figure 4-6 Sample Image Augmentation

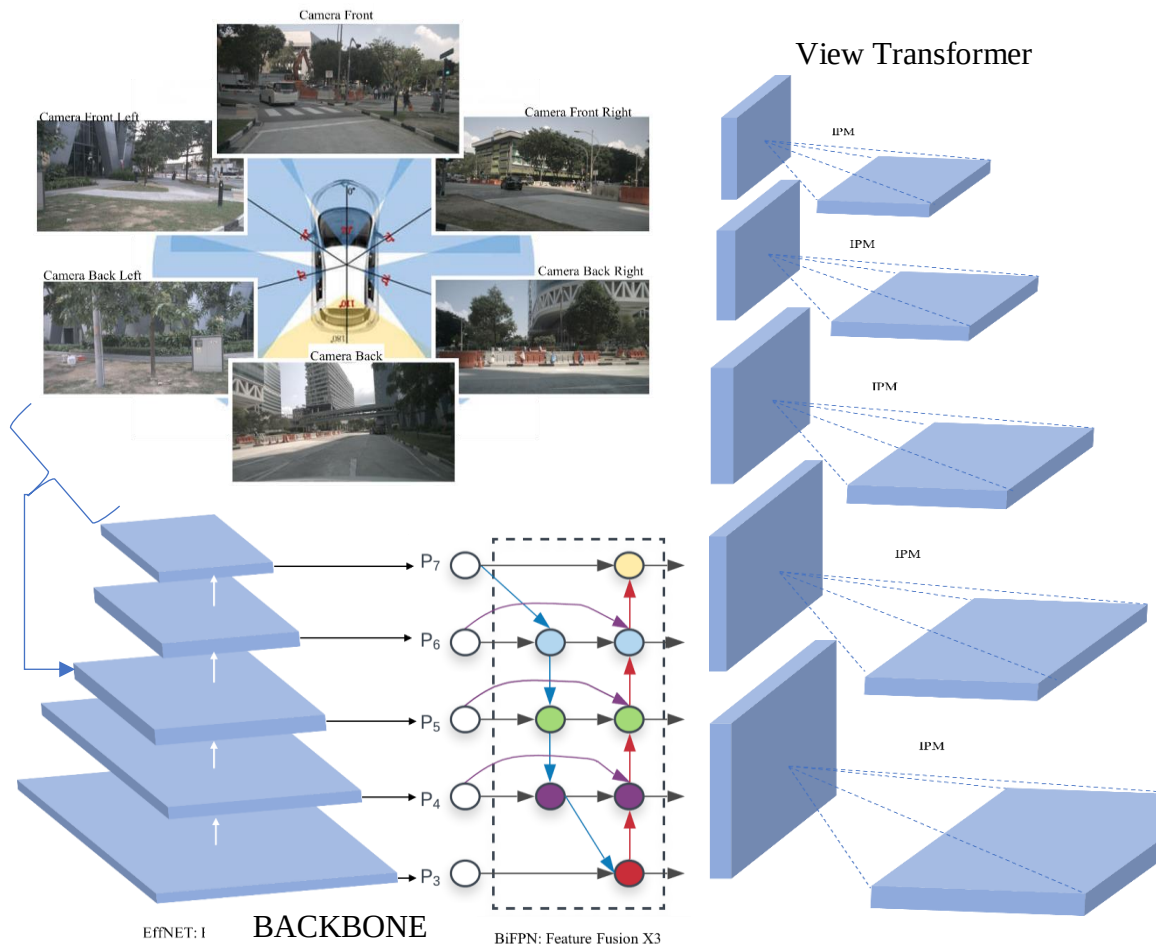
4.4 CONCLUSION

In this chapter, the common calibration processes for image sensors were presented using the camera model concept and the Direct Linear Transform technique. An automatic extrinsic calibration technique was then proposed for the multi-sensor fusion architecture for this report. This technique was achieved using image augmentation and normalisation techniques to make multi-sensor fusion model rotation and translation invariant. With a rotation and translation invariant model, the model becomes robust enough to learn the unified BEV perspective mapping (see Chapter 5) of features from different camera inputs fused with the LiDAR BEV features. This becomes achievable for the model after the intrinsic matrix of the camera sensor has been predetermined using any of the common calibration (Zhang's and Tsai's) techniques highlighted in this chapter.

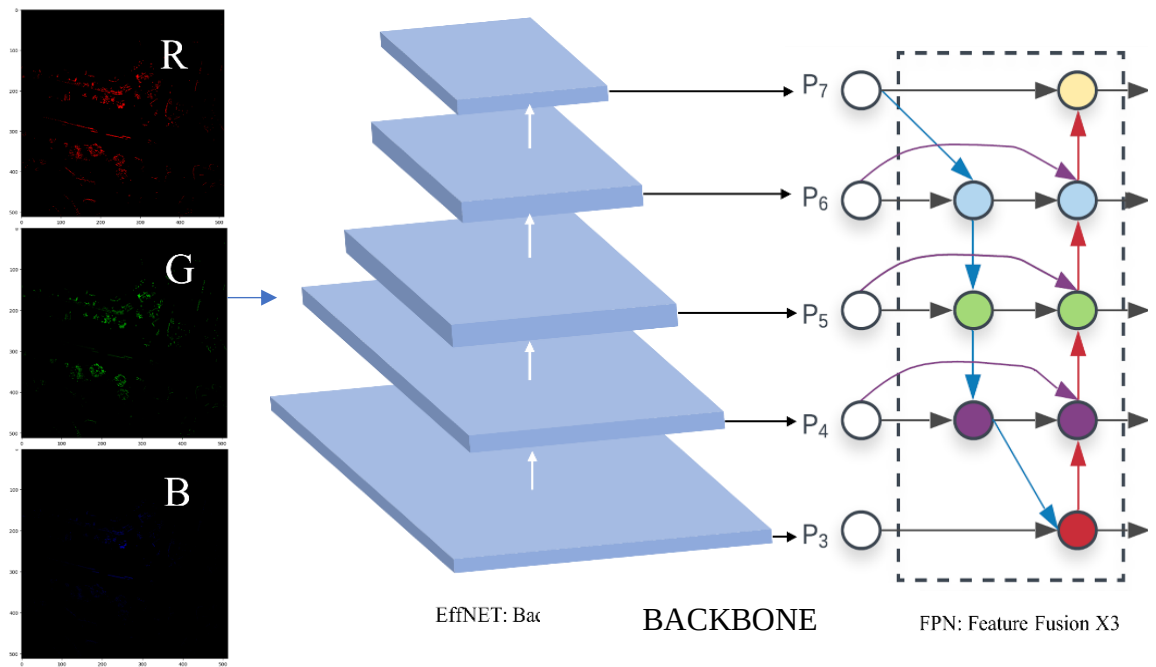
CHAPTER 5: BEVMSF3D AND IOT FRAMEWORK

5.1 INTRODUCTION

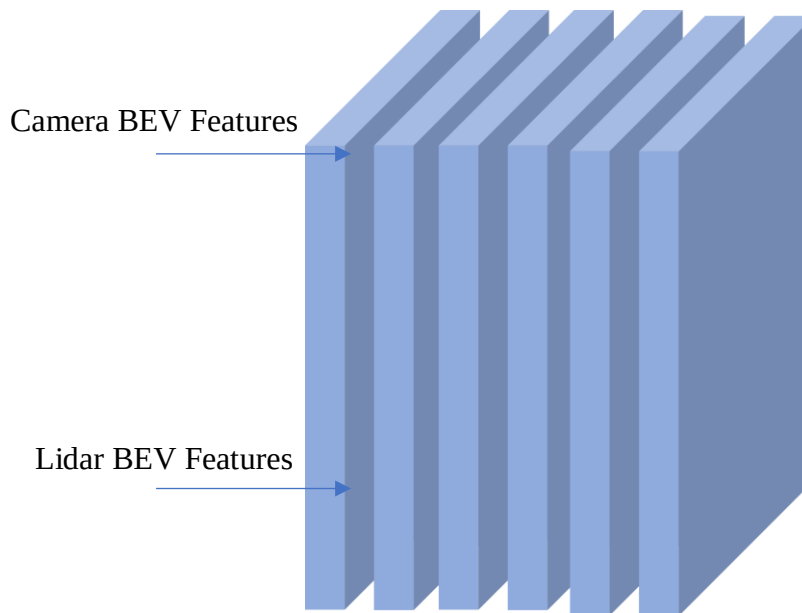
In this chapter, the unified BEV for single or multiple camera(s) and lidar fusion is presented. The sensor fusion pipeline consists of four modules which are the feature extraction module, the view transformation module, the feature fusion module and the classifier module. Figure 5-1 presents an overview of BEVMSF3D.



(a) Each Camera image passes through the backbone and extracted features are passed through the View transformer to its equivalent BEV



(b) The single Lidar Point cloud image is also passed through the Backbone to obtain its BEV features



(c) The BEV Lidar and Camera features are Concatenated to give a single BEV Feature

Figure 5-1 The sensor Fusion Pipeline

5.2 MULTI-SENSOR FUSION ARCHITECTURE

5.2.1 Sensor data preprocessing

To begin the BEVMSF3D architecture, the image augmentation technique is employed to enhance the robustness of the trained fusion model, during the training phase. This technique is aimed at ensuring the model's ability to handle rotations and translations effectively, as elaborated in section 4.3.4. The novel FastSeg3D algorithm, as detailed in section 3.3, is employed on the raw point cloud data to effectively remove ground points. The 3D point cloud object is then converted to its equivalent BEV image in RGB format, as presented in section 4.3.3.

As emphasised in section 2.4.3 of chapter two, it is worth noting that most ground truth data used in training deep learning models for computer vision tasks, such as object detection, heavily relies on depth information, primarily obtained from LiDAR sensors. Therefore, the incorporation of LiDAR information plays a significant role in enhancing the efficiency of our model for assisted navigation.

5.2.2 Feature extraction: The backbone module

To extract the features of the camera image and LiDAR image, the EfficientNet [278] and BiFPN modules were employed as the multi-sensor fusion backbone. EfficientNet is a family of convolutional neural networks that scales up convolutional neural networks by balancing network depth, width, and resolution to achieve better performance. The architecture uses depthwise separable convolutions which are a combination of depthwise convolutions and pointwise convolutions. Depthwise convolutions apply a single filter to each input channel separately while pointwise convolutions apply a 1x1 filter to combine the output of depthwise convolutions, see Figure 5-2. The EfficientDet [279] model uses EfficientNet as the backbone network and a newly proposed multi-scale features fusion block called Bidirectional Feature Pyramid Network (BiFPN).

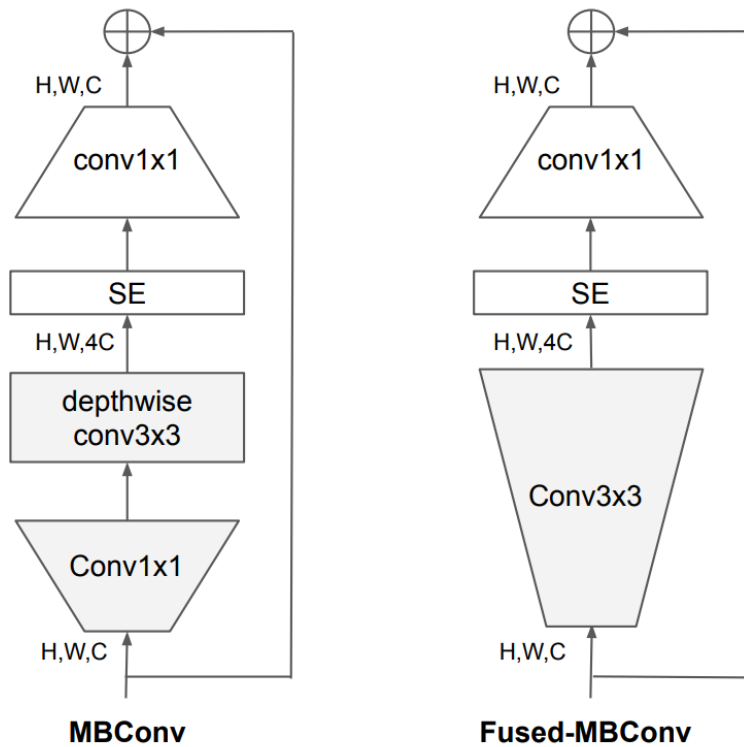
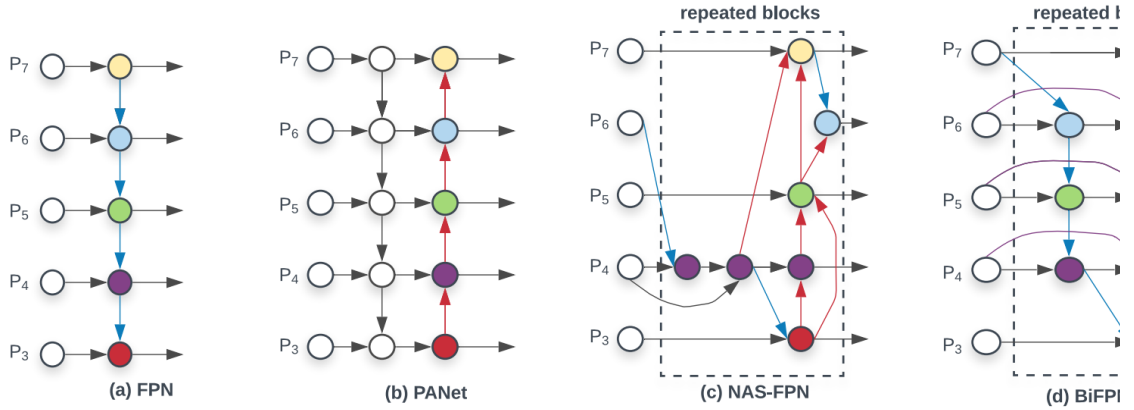
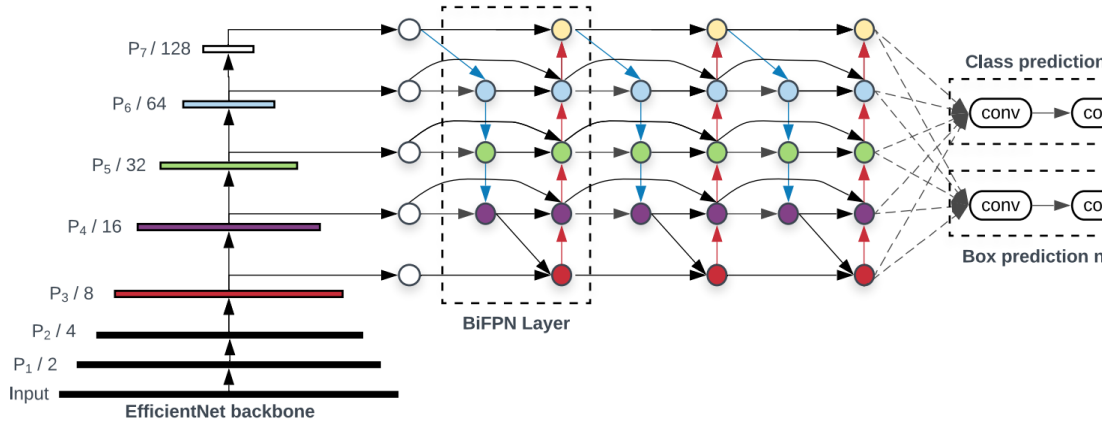


Figure 5-2 A Single EfficientNet Mobile Inverted Bottle Neck Convolution (MBConv) and its improvement for EfficientNet version 2 [278]

The BiFPN is part of the family of FPN, which was proposed first by [280] to address the problem of detecting small objects by fusion different levels of feature extracted sizes. This is an improved version of the Neural Architecture Search Feature Pyramid Network (NAS-FPN) [281] that allows for bidirectional feature fusion between different levels of the feature pyramid. BiFPN achieves bidirectional feature fusion by allowing features from the low-resolution P3 levels to high-resolution P7 levels to repeatedly flow in both top-down and bottom-up ways, using a weighted feature fusion strategy to combine features based on their importance. This makes it more efficient than previous feature networks, see Figure 5-3.



(a) Family of FPN



(b) BiFPN

Figure 5-3 BiFPN and the FPN family [279]

The EfficientNet and BiFPN have shown better accuracy and efficiency than existing convolutional neural networks and FPNs. This forms the backbone of BEVMSF3D for this report, and it is used to extract camera images and point cloud BEV images for further processing.

Given an input tensor X_i , which could be the camera image X_i^{cam} or point cloud image X_i^{pc} , with tensor shape (H_i, W_i, C_i) , where H_i and W_i are the spatial dimensions and C_i is the

channel dimension and a function $Y_i = F_i(X_i)$, where F_i is the operator and Y_i is the output tensor of the input X_i . A ConvNet N representing a list of convolution layers can be expressed as:

$$N = F_k * \dots * F_2 * F_1(X_{(H_i, W_i, C_i)}) = \ast_{j=1 \dots k} F_j(X_{(H_i, W_i, C_i)}) \quad (5.1)$$

Where $*$ is the symbol for convolution, k is the total number of times the convolution is performed and j is the current convolution.

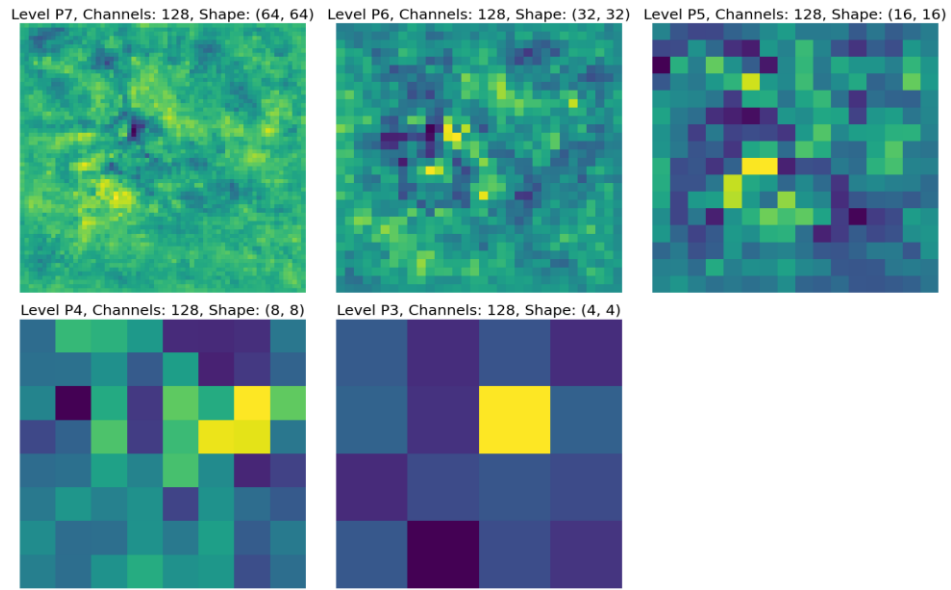
Given a set of extracted features P_i^{in} (where i is the extracted feature level number) from input tensor X_i , the BiFPN uses a set of weights w_i and w'_i to perform feature fusion across feature levels. This can be expressed as:

$$P_6^{td} = Conv \left(\frac{w_1 \cdot P_6^{in} + w_2 \cdot Resize(P_7^{in})}{w_1 + w_2 + \epsilon} \right) \quad (5.2)$$

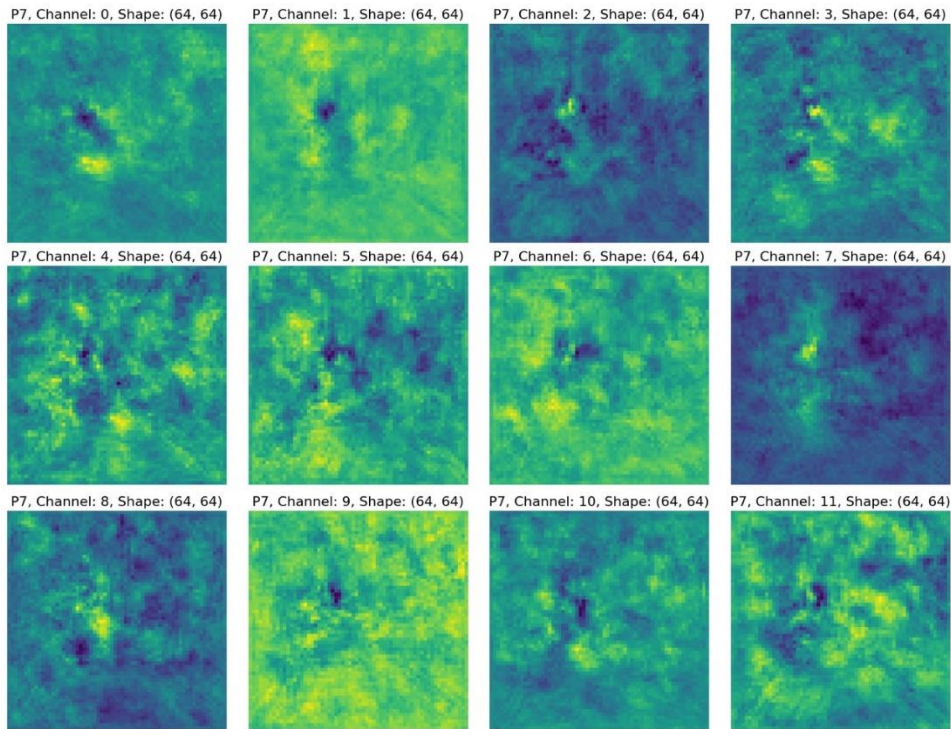
$$P_6^{out} = Conv \left(\frac{w'_1 \cdot P_6^{in} + w'_2 \cdot P_6^{td} + w_2 \cdot Resize(P_5^{out})}{w'_1 + w'_2 + w'_3 + \epsilon} \right) \quad (5.3)$$

Although the point cloud BEV features are already in BEV format, this is not the case for the camera feature image. Therefore, this has to be addressed using camera intrinsic parameters to perform a perspective mapping as discussed in the next subsection.

The features extracted for a single camera image are presented in Figure 5-4. Each feature level is set to a depth (channel size) of 128. Therefore, the features for each level are 128 depth each and shape (w, h) varying from $(64, 64)$ for the highest resolution features, P_7 to $(4, 4)$ for the lowest resolution, P_3 .



(a) A condensed view of extracted features for Levels P7 to P3



(b) The first 8 of 128 channels of the level P7 of the extracted features

Figure 5-4 Features of a single camera image

5.2.3 Inverse perspective mapping (IPM) and the view transform module

The inverse perspective mapping is a well-known concept in computer vision (CV). It is a transformation technique used to map a 3D object which has been projected onto a 2D image plane to a new position to create a new 2D image, like a top-down or bird's-eye view (BEV) image. It is often applied to remove the perspective effect from a vehicle's front-facing camera and to remap its images into a 2D domain, resulting in a top-down view [282,283]. The BEV image attempts to preserve distance and parallel lines, correcting the perspective effect. IPM involves using a homography matrix H that relates the coordinates (x, y) of a point P_{src} in the image plane to the coordinates (X, Y) of the same point P_{dst} in the BEV plane. The matrix can be computed from the camera intrinsic matrix K and the ground plane equation M , or learned from data using a deep neural network. The homography projection matrix H can be expressed as $K[R|t]$ where K is the intrinsic matrix of the camera and $[R|t]$ is the extrinsic matrix of the camera [223]:

$$H = K[R|t] = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \quad (5.4)$$

In OpenCV, this can be estimated without using the camera intrinsic and extrinsic matrices using a function called `getPerspectiveTransform` once four points in the source plane M_{src} and destination plane M_{dst} have been defined. This known or estimated Homography matrix (perspective transform) is then used to warp the source image plane to the destination plane (i.e. BEV image). Warping a source point P_{src} to a destination point P_{dst} can be expressed as:

$$P_{dst} = HP_{src} \quad (5.5)$$

$$P_{src} = H^{-1}P_{dst} \quad (5.6)$$

The destination plane is formulated using the plane mathematical equation:

$$M_{dst} = aX + bY + cZ + d \quad (5.7)$$

This plane mathematical equation is the same as equation 3.11. This model is used to create a mesh grid for the BEV plane with $Z = 0$.

Figure 5-5 shows an example of an image transformed from its front camera to a top-down view (BEV).

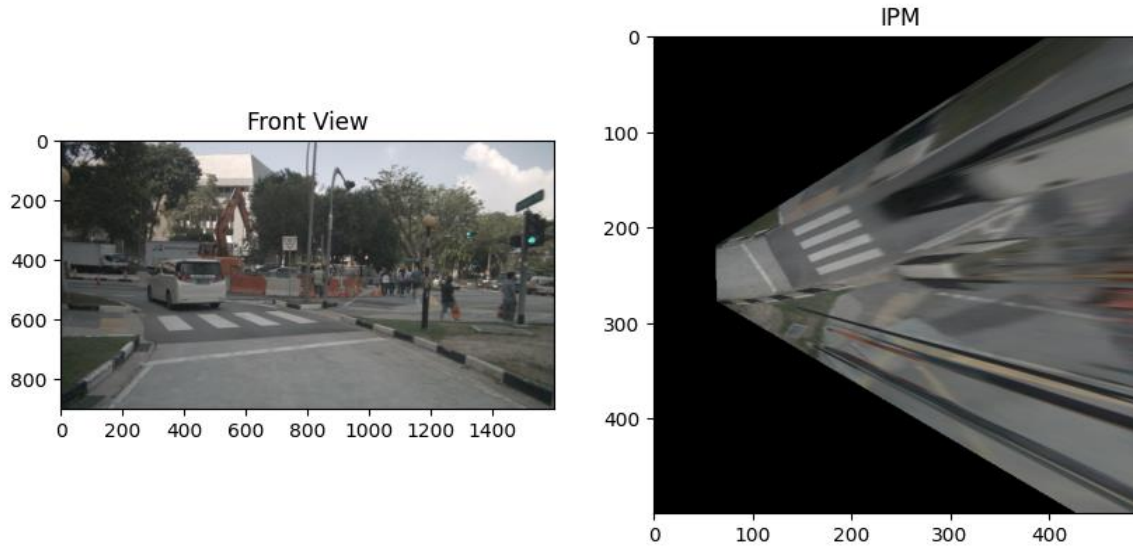


Figure 5-5 Traditional IPM from Front to Top-down View

Using deep learning (supervised learning) the IPM technique can be trained to properly represent a front-view image to a top-down image. This process is achieved using the following steps:

5.2.3.1 Resample features

The camera image features are resampled using the camera's intrinsic matrix and the created BEV mesh grid, see equation 5.7. First, the extracted features from the image are transformed into the camera coordinates using the camera intrinsic matrix. This camera coordinates are repositioned to look like the camera is facing down in a top-down manner. Figure 5-6 shows the relationship between the image plane and the BEV plane. Therefore only the columns of the intrinsic matrix that are responsible for representing the features in

the BEV plane are used for resampling. The algorithm to accomplish this step is shown in Table 5-1.

Table 5-1 Resample features algorithm

Algorithm: Resampling Image Features to Bird's Eye View (BEV)	
Camera Coordinate	
Input:	
features:	Image Features
calib:	Calibration Matrix (3 × 3)
resolution:	BEV Resolution
dimensions:	BEV Grid dimensions in the format $[x_{min}, z_{min}, x_{max}, z_{max}]$
Output:	
	Resampled Features in BEV Camera Coordinate
Initialisation:	
near	$\leftarrow \text{dimensions}[1]$
far	$\leftarrow \text{dimensions}[3]$
grid	$\leftarrow \text{_create_bev_grid}(\text{resolution}, \text{dimensions})$
1 Transform to Camera Coordinates:	
2	$\text{Cam}_{coord} = \text{calib}_{y=0} \times \text{grid};$
3 Calculate bev x-coordinates and z-coordinates:	
4	$x_{bev} = \frac{\text{cam}_{coord}(x)}{\text{cam}_{coord}(y)};$
5	$x_{bev_norm} = \left(\frac{x_{bev}}{\text{features}_{size}} \times 2 \right) - 1;$
6	$z_{bev} = \left(\frac{\text{cam}_{coord}(y) - \text{near}}{\text{far} - \text{near}} \times 2 \right) - 1;$
7	$\text{grid}_{coord} = \max \left(-1.1, \min \left(1.1, \text{stack}(x_{bev_norm}, z_{bev}) \right) \right);$
8	<i>grid_sample(features, grid_{coord})</i>

The resampler algorithm begins by initialising parameters such as the **near** and **far** **dimensions** and creating a BEV grid for resampling based on the specified **resolution** and **dimensions**. This grid is created using PyTorch's mesh grid function, which creates a rectangular grid from an array of $z_{max} - z_{min}$ with an increment of the specified *resolution* and an array of $x_{max} - x_{min}$ with an increment of the specified *resolution*. It

then transforms the grid into camera coordinates using the provided calibration matrix. Subsequently, it calculates x-coordinates and z-coordinates to form grid coordinates. These grid coordinates are clamped to be between -1.1 and 1.1. This resolution of approximately 2m is set to strike a balance between accuracy and speed. Finally, the algorithm performs grid sampling, using PyTorch's grid sample function, on the input image features using the calculated BEV grid coordinates, resulting in resampled features in the Bird's Eye View (BEV) camera coordinate system as the output.

The grid sample function performs a spatial transformation on the given features to the specified grid coordinate, the input and output shape remains the same, only a transformation of features to the specified new coordinates.

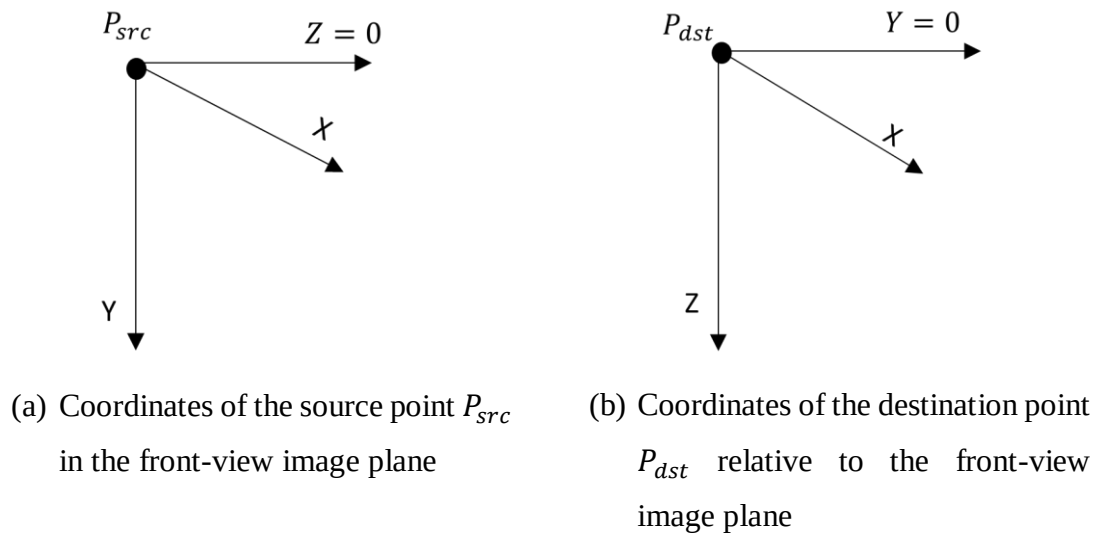


Figure 5-6 Relationship between the image plane and the BEV plane

5.2.3.2 View transform module

Although features have been resampled to the top-down view, the resampler only performs a mathematical transformation, see equations 5.4 to 5.6. This is prone to errors and distortion, as shown in Figure 5-5. To address this, the model would need to learn to make corrections to any wrongly positioned or transformed features. The view transformer

module is designed to address this. To achieve this the algorithm in Table 5-2 shows how learnable convolution layers are incorporated to improve the view transformation.

Table 5-2 Algorithm of view transform features

Algorithm: View Transform Features
Input: features: Image Features calib: Calibration Matrix (3 x 3) resolution, res: BEV Resolution dimensions: BEV Grid dimensions in the format $[x_{min}, z_{min}, x_{max}, z_{max}]$ in_channels, ch_{in}: Number of Input Channels channels, ch: Number of Intermediate Channels ymin, ymax: BEV Y-coordinates. See Figure 5.6. This is set to (-1.0, 1.0). focal_length, f_{bev}: Specified the camera <i>focal length</i> in the top-down view (BEV) position.
Output: Resampled Features in BEV Camera Coordinate
Initialisation: $conv \leftarrow Conv2D(ch_{in}, ch, kernel = 1)$ $bn \leftarrow GroupBatchNorm(16, ch)$ $resampler \leftarrow _resampler(res, dimensions)$ $in_height, h \leftarrow roundup\left(f_{bev} \times \frac{y_{max} - y_{min}}{z_{min}}\right)$ $y_{mid} \leftarrow \frac{y_{max} + y_{min}}{2}$ $depth \leftarrow roundup\left(\frac{z_{max} - z_{min}}{res}\right)$ $fc \leftarrow Conv1D(ch \times h, ch \times depth, kernel = 1)$. This is the fully convolution layer. $out_channels \leftarrow channels$
1 for $pair(feats, cal)$ in $pair(features, calib)$: 2 focal length $f = f_x$; 3 offset, $p = p_x$; 4 vertical middle, $v_{mid} = \left(\frac{y_{mid} \times f}{z_{min} + p}\right)$;

```

5   vertical minimum,  $v_{min} = \left(\frac{v_{mid}-h}{2}\right)$ ;
6   vertical maximum,  $v_{max} = \left(\frac{v_{mid}+h}{2}\right)$ ;
7    $feat_{pad} = pad(feat, [0, 0, -v_{min}, v_{max} - feat_{size\ of\ the\ x-axis}])$ ;
8   stack the padding features:  $feat_{stack} = \sum_{i=0,...,n} feat_{pad(i)}$ ;
9   Conv2D, bn, and ReLU features,  $feat_{conv}$ ;
10  flatten the convolved features:  $feat_{flat} = flat(feat_{conv})$ ;
11  fc the flattened features:  $feat_{bev} = fc(feat_{flat})$ ;
12  resampler( $feat_{bev}, calib$ );

```

The algorithm begins by initialising several parameters. These parameters are; the convolution layer, batch normalisation type, and an instance of the resampler algorithm. The algorithm refines the features before they are passed to the resampler. This is major because the resampler only focuses on resampling input features, there is a need to properly define the BEV image plane (X, Y). To define the BEV image plane, the Y -coordinate is determined to be a total of 2m i.e. $(-1m, 1m)$, see input. This Y coordinate is used to pad the initially resampled features to adjust its column (i.e. Y -axis) size, see line 7. The final BEV would have its x-axis unchanged, see line 7 (0,0 represents x_{min} and x_{max}). To achieve this the set Y -coordinate, a set focal length, f_{bev} and the Z -coordinate of the predefined grid extent or dimension is used to determine the height (h), which is the input height. Also, the y_{mid} and $depth$ of the BEV camera coordinates about the BEV plane are initialised.

After initialising the relevant attributes, the features are padded to suit the new BEV set environment. This is achieved using the focal length and principal point of the x-axis of the camera's intrinsic parameters, remember the y-axis is zero, as shown in Figure 5-6. A new vertical minimum and maximum value are determined and used to pad the input features. These padded features and stacked up together and convolved. This convolution layer is responsible for learning and correcting features. This is then flattened and passed through the fully convolution layer which serves as a decoder to decode the input features that have been adjusted to be in line with the BEV image plane.

5.2.3.3 Applying the view transform to all feature levels

So far, the view transform features and resampler algorithms are designed to transform the features for a given level e.g. P_6 to BEV image plane, see section 5.2.2 and Figure 5-4. To apply this to all feature levels a simple algorithm is used to organise features into a list, each position holding the features for a level. Then iteratively, the view transform and resampler are performed on each level and all the features for each level are concatenated (i.e. merged) to return a single merged feature for that particular level. This algorithm is presented in Table 5-3.

Table 5-3 Algorithm to apply view transform to all feature levels (also called Pyramid features)

Algorithm: Transform Pyramid Features to BEV.	
Input:	
features:	List of Image Features
calib:	Calibration Matrix (3 x 3)
resolution, res:	BEV Resolution
dimensions:	BEV Grid dimensions in the format $[x_{min}, z_{min}, x_{max}, z_{max}]$
in_channels, ch_{in}:	Number of Input Channels
channels, ch:	Number of Intermediate Channels
ymin, ymax:	BEV Y-coordinates. See Figure 5-6. This is set to $(-1.0, 1.0)$.
focal_length, f_{bev}:	Specified the camera <i>focal length</i> in the top-down view (BEV) position.
Output:	
List of Resampled Features for each level in BEV Camera Coordinate	
Initialisation:	
view_transform list, Tx_{list} $\leftarrow$ <i>view transform</i>	
for i in size of Tx_{list} :	
f_{level}	$\leftarrow \frac{f_{bev}}{(i+3)^2}$
$z_{level(max)}$	$\leftarrow \min(\text{rounddown}(f_{level} \times 2) \times res, z_{max})$
$z_{level(min)}$	$\leftarrow \text{rounddown}(f_{level}) \times res$
$dimensions_{level}$	$\leftarrow [x_{min}, z_{level(min)}, x_{min}, z_{level(max)}]$
Tx_{level}	$\leftarrow \text{view_transform}(ch_{in}, ch, res, dimensions_{level}, ymin, ymax, f_{level})$
Tx_{list}	$\leftarrow \text{append}(Tx_{level})$

```

1 BEV features list,  $bev_{list}$ 
2 for  $i, feats$  in  $features$ :
3      $zoom\_in = 4 \times 2^i$ ;
4      $calib_{feats} = \frac{calib}{zoom\_in}$ ;
5      $bev_{list} \leftarrow append(Tx_{list(i)}(feats, calib_{feats}))$ ;
6  $bev_{level} = concat(bev_{list})$ ;

```

As stated earlier, the View Transform Pyramid Algorithm helps to implement the previously presented algorithms compactly. The feature level, f_{level} and z-coordinates, (z_{min}, z_{max}) are adjusted to account for the different resolutions of each feature level in the pyramid. This is then used to instantiate the view transform algorithm for that level, Tx_{level} . Recall that levels are from P7 to P3. After this, the calibration for the current level is then adjusted (zoomed in) to account for the differences in channel values for each level. The BEV feature list, bev_{list} is used to collect all transformed features for a current level and then these features are concatenated to give a single merged feature for a given level.

5.2.4 Feature fusion

After the camera image features have been successfully transformed to BEV, the resulting output comprises a set of transformed features one for each level from P7 to P3. This is the same with the 2D RGB image map for the point cloud, whose features have also been extracted. These features can now be concatenated to give a single set of BEV features representing each level from highest to lowest resolution. This concatenation is done using the PyTorch's concatenate function. The concatenated features are then passed through the Fused-MBConv layer three times and the output is sent to the classifier, see code 7 in the appendix.

5.2.5 Classifier module and loss function

A Bayesian classifier is used to generate semantic occupancy grid probabilities $p(m_i^c | z_t)$, a concept used in robotics to represent objects' position in the discrete grid (BEV plane). This is accomplished using a sigmoid function to generate a prediction of the desired output

classes and their corresponding position in the BEV image plane. During training two loss functions, see equations 5.8 to 5.10. are used to train the model. This Bayesian classifier algorithm is presented in Table 5-4.

Table 5-4 Algorithm for Bayesian classifier with dropout sampling

Algorithm: Bayesian Classifier with Dropout Sampling	
Input:	
in_channels, ch_{in}:	Number of Input Channels
num_class, cls:	Number of output classes.
num_samples, s:	Number of dropout samples for test-time prediction.
prior, p:	Prior probability distribution over classes.
Output:	
Class logits (probability) for each input feature.	
Initialisation:	
conv	$\leftarrow \text{Conv2D}(ch_{in}, cls, \text{kernel} = 1)$
prior, p	$\leftarrow \text{tensor}(p)$
conv.weights, w	$\leftarrow (0 + \epsilon)$
conv.bias, b	$\leftarrow \log\left(\frac{p}{1-p}\right)$
1 if training:	
2	$\text{features} = \text{dropout2D}(\text{features}, 0.5, \text{training} = \text{True});$
3	$\text{pred} = \text{conv}(\text{features});$
4 else:	
5	$\text{feats} = \text{dropout2D}(\text{features}, 0.5, \text{training} = \text{True});$
6	$\text{get_mean_score} = \text{sigmoid}(\text{conv}(\text{feats}));$
6	$\text{pred} = \text{get_mean_score};$

The Bayesian Classifier with Dropout Sampling algorithm performs a simple 2D convolution, with its weight and bias initialised to be close to zero. The initialised bias of the log ensures that the bias is between 0 and 1, just like the prior which is a probability between 0 and 1. Also, the predicted output probability, pred is between 0 and 1. The

random dropout of features is a regularisation technique used to prevent overfitting of the model to the training dataset, this helps for better generalisation.

The loss function plays an important role in the success of the model. It helps to calculate the difference between the ground truth and the predicted probability of classes. The weighted binary cross entropy loss function, L_{bce} is combined with an uncertainty loss function, L_{uncert} to discourage the multi-sensor fusion model from making assumptions in regions with high uncertainty. This can be expressed as follows:

$$L_{bce} = \frac{1}{N} \sum_{i=1}^N [m_i^c \log(p(m_i^c|z_t)) + (1 - m_i^c) \log(1 - p(m_i^c|z_t))] \quad (5.8)$$

$$L_{uncert} = \frac{1}{N} \sum_{i=1}^N (1 - p(m_i^c|z_t)) \log(p(m_i^c|z_t)) \quad (5.9)$$

$$L_{total} = L_{bce} + \lambda L_{uncert} \quad (5.10)$$

Where:

- m_i^c is the true label for the i -th sample.
- $p(m_i^c|z_t)$ is the predicted occupancy grid probability.
- N is the number of samples in the dataset.
- L_{uncert} is the uncertainty loss function.
- L_{bce} is the binary cross entropy loss function.
- L_{total} is the total loss function for the multi-sensor fusion model.
- λ is a constant weight factor for the L_{uncert} . This is set to 10^{-3} .

5.3 THE PROPOSED IOT-FRAMEWORK

5.3.1 IoT-framework

One of the cooperative abilities that a network of assistive mobility robots (AMRs) can benefit from is the reference to the mapping of the visited location of other assistive mobility robots. When multiple sensors are fused successfully, the SLAM technique generates a map as an outcome. This map can be useful information to other AMRs accessing or visiting the same location, referred to as data sharing amongst AMRs in the IoT network [284–286]. Combining these maps, which would need to be retrieved by the AMR in real-time, with GPS coordinates and the real-time multi-sensor fusion and SLAM algorithm, the current AMR in the environment can significantly improve safety, efficiency, and speed in navigation tasks. This can also significantly reduce the bandwidth and storage requirements for the cloud.

Research shows that many application layer protocols designed for video transmission over low-power lossy networks (LLNs) in IoT have been proposed [287–289]. However, extensive analysis and comparison by Sultana *et al.* [289] showed that Message Queuing Telemetry Transport (MQTT) [290,291] is a promising solution for the Internet of Video Things (IoVT) at the edge node with the consumption of less bandwidth and short latency. Another report confirmed that although the UDP-based transport protocol is a faster protocol for communication, the lack of packet guarantee and QoS models can extremely affect video quality transmitted over LLNs.

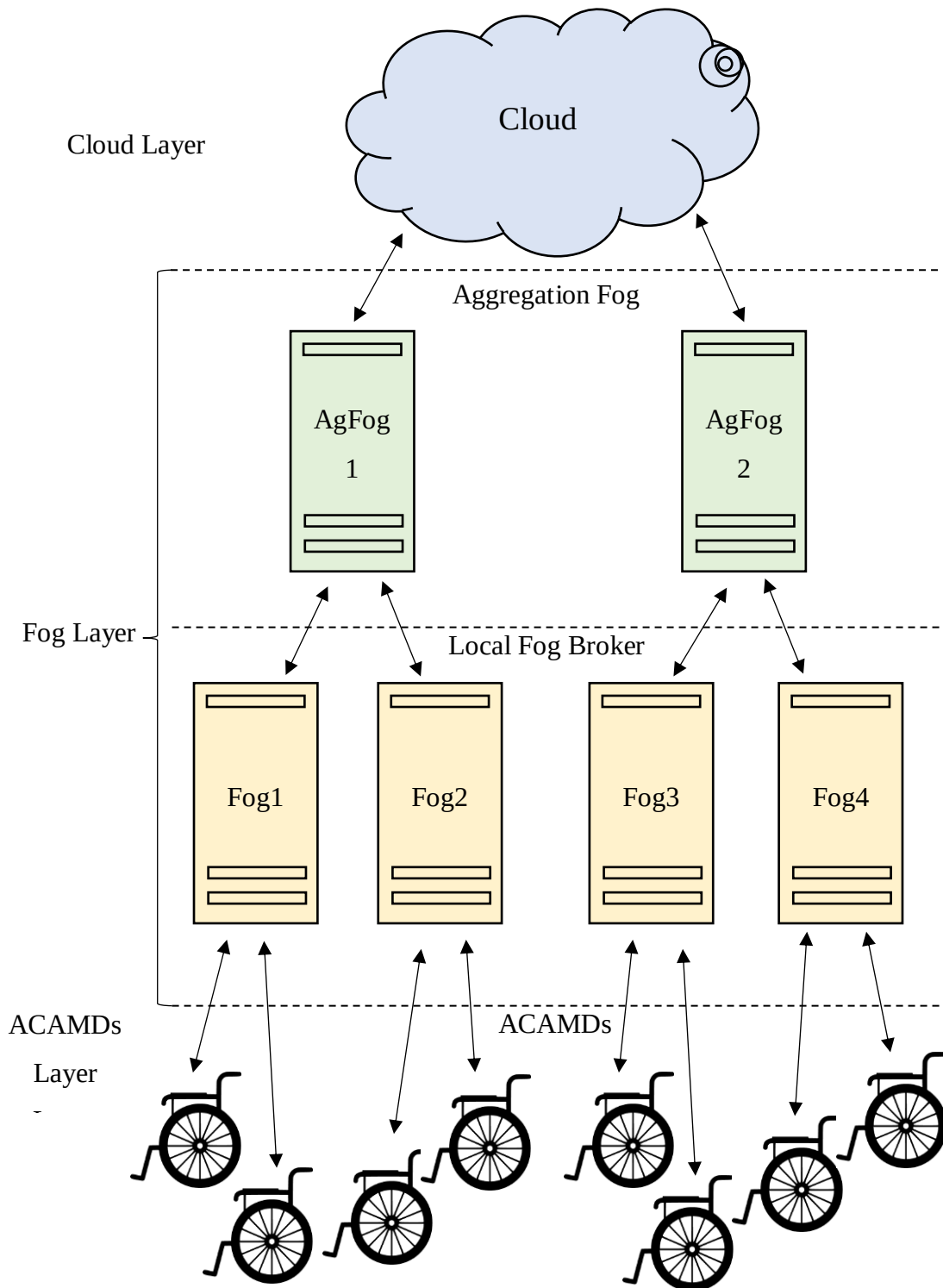


Figure 5-7 IoT Framework for multi-sensor fusion for ACAMDs

Figure 5.7 presents an overview of the recommended IoT framework. This framework utilises fog computing and the MQTT application protocol to achieve efficient and reliable data sharing. The goal of this framework is to enable AMRs to securely upload and share mapping data of their surrounding environment to the cloud, creating a collaborative, real-time reference map for all AMRs in the network.

5.3.1.1 Data latency: Fog computing and the MQTT application protocol

As presented in Figure 5.7, the idea of the proposed IoT framework is to group the ACAMDs to establish real-time interactive communication with the nearest fog server, called the local broker. At the ACAMDs layer, sensor fusion takes place and environments' mappings are developed. Here the ACAMDs leverage the subscriber/publisher model of the MQTT clients to send and receive video and text messages of the BEV mapping of a particular location from the closest local fog broker. Topics are used to properly organise mappings with accompanying topics indicating the ACAMD's start and end location/position coordinates per time (in longitude and latitude), duration of achieving navigational tasks, and other possible messages like the users' state. For instance, wheelchair user's pulse rate, fall cases and possible collisions, and panic button pressed.

To ensure real-time communication, the use of a BEV video mapping to JavaScript Object Notation (JSON) coordinates mapping system is recommended. Table 5-5 shows a high-level algorithm of how this can be achieved. The main goal is to use the SLAM/BEV mapping of the environment with the help of a GPS to establish the positions of the ACAMDs and all detected objects relative to the ACAMDs at every given time. These positions and timestamps are then converted to a JSON file sent as a message and stored in the cloud.

One of the main functions of the aggregate fog broker is traffic management and a slave database. This is similar to the function of the load balancer [292–294] which works to ensure publisher/subscriber communication is evenly distributed across the brokers to avoid traffic congestion in the network. The aggregate fog broker is also responsible for

publishing messages for every given session which is a successful navigation task of each ACAMD to the cloud MQTT broker, which is also the master database, and subscribing to the relevant request from the cloud.

Table 5-5 High-level conversion from BEV mappings to JSON

Algorithm 1: Coarse Segmentation (REE)

Input:

- SLAM recorded video (BEV video)
- Robot positions and timestamps
- Detected objects and their positions relative to the robot at each timestamp.

Output: JSON file containing robot positions, object positions, and timestamps

Params:

JSON: Output JSON dictionary

Initialisation:

JSON = { }

```

1 for every frame in the BEV video do
2   timestamp_data = [ ] # initialize an empty list
3   get robots position;
4   for every detected object in the current frame do
5     calculate the relative position to robots position;
6     objects data = {"object_type" : static (as 0) or moving (as 1),
                     "relative_position": relative position value};
7   timestamp_data.append(objects data);
8 JSON[timestamp] = timestamp_data;
```

5.3.1.2 Security: Authentication management system

The lightweight mutual authentication scheme based on a hash function and the XOR operation proposed by [295] is adopted for this IoT Framework. The scheme is designed to address the limitations of current authentication mechanisms in the Industrial Internet of Things (IIoT) and the increased computation, communication, and storage overhead arising from these mechanisms. Each broker in the architecture is deployed with an authentication manager, which helps in securing the IoT framework. The scheme reduces storage and communication overheads by 89% and 23% respectively, as compared to a benchmark scheme introduced by Diro A. *et al* [296].

5.4 CONCLUSION

In this chapter, a sensor fusion architecture and an IoT framework were developed. The sensor fusion architecture consisted of three main modules: the backbone or feature extraction module, the view transformation module and the Bayesian classifier/occupancy grid module. The backbone is responsible for extracting images into its equivalent features. Features are extracted into four levels using a feature fusion pyramid network to improve feature extraction accuracy for small objects. The view transformation module uses the inverse perspective mapping concept and some learnable convolution layer to successfully transform the extracted features of the image into its equivalent bird's eye view (BEV) or top-down view. The Bayesian classifier/occupancy grid module is then used with the binary cross-entropy loss and the uncertainty function to correctly place fused features in their corresponding grid positions in the BEV grid.

An IoT framework is recommended using the fog computing infrastructure and the MQTT protocol to effectively achieve a low power consumption data sharing and scalable IoT framework. To address the constrain of transmitting and receiving large video files in real-time, a BEV mapping video to JSON algorithm was proposed to convert BEV mapping video to a more compressed form (JSON file) consisting of the positions of the mobile robots or ACAMDs relative to all objects detected for every capture timestamp.

CHAPTER 6: RESULTS AND DISCUSSION

6.1 INTRODUCTION

This chapter presents the experimental setup and results for BEVMSF3D pipeline. The goal of this research study was to investigate and develop a suitable algorithm for multi-sensor fusion for ACAMDs. The sensors used for this study were the monocular cameras and lidar sensors.

6.2 FASTSEG3D

To validate the study's contribution and verify the progressiveness of both modules (REE, and WBMF), in this section the performance increase is presented with the introduction of each module and the accuracy to speed tradeoff, Table 6-1. Then FASTSEG3D is compared to some previous works [87,297,298] as shown in Table 6-2 using three (3) of the NuScene scenarios to demonstrate each algorithm's performance on uneven terrain.

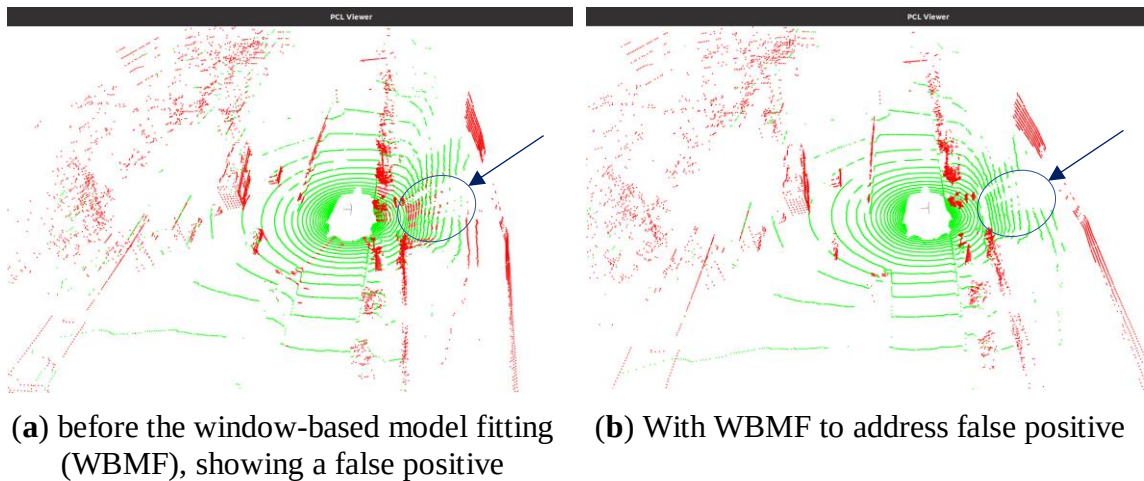
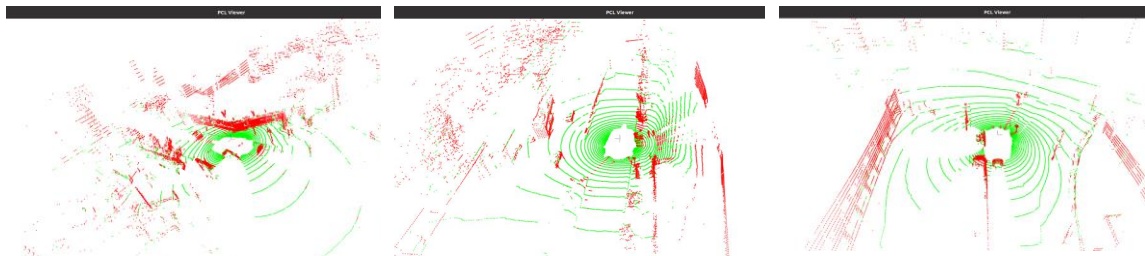


Figure 6-1 FASTSEG3D: Before and after the WBMF



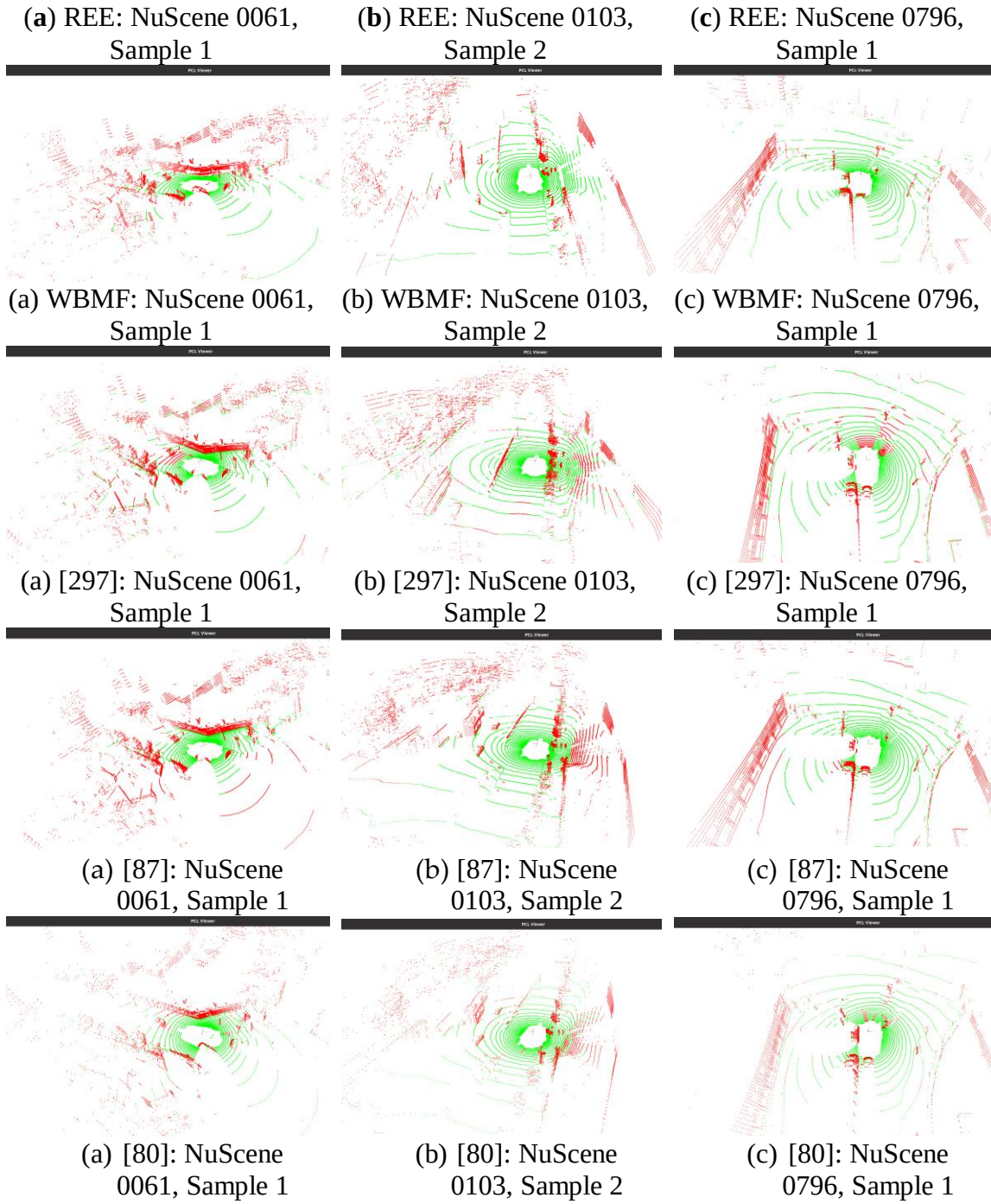


Figure 6-2 Qualitative comparison between FASTSEG3D before WBMF (REE), after WBMF, and some previous works [87,297,298].

6.2.1 Experimental setup:

The experiment for all ground segmentation algorithms was performed using the C++ Standard Library, Point Cloud Library (PCL), OpenCV and Eigen Library in Linux Ubuntu 20.04 Operating System (OS). They were all performed on the same computer, Intel i5-7200U and 8GB RAM, for consistency. Thereafter, the NuScene [245,299] dataset was adopted to evaluate and compare FASTSEG3D proposed ground removal algorithms to other efficient algorithms. Based on the point-wise labels, the points are labelled as *flat.terrain*, *flat.other*, *flat.sidewalk*, *flat.driveable* [245,299], are considered to be ground points, points labelled as *noise* were ignored, and every other point was considered to be a non-ground point.

6.2.2 Experimental parameters for FASTSEG3D

To experiment the following parameters were set as follows: range image dimension $(row, column) = \{(x_{min}, x_{max}), (y_{min}, y_{max})\} = \{(0, 1024), (0, 70m)\}$ for the NuScene dataset. Elevation tolerance $E_t = 0.2m$.

For the WBMF, the window dimension was set to, row i.e. w was set to $w = \frac{1024}{16}$ and column i.e. $c = \frac{70}{5}$. Minimum and maximum seed points per window were set to 5 and 20 respectively. The threshold values for distance to the window's origin, inclination angle and elevation were set to $d_{th} = 1m$, $\alpha_{th} = 1^\circ$, $E_{th} = 0.2m$ respectively.

6.2.3 Evaluation metrics:

To evaluate the proposed approach the recall for ground and non-ground points, frame per second (fps) and the mean IoU (intersection over union) for predicted points for each scene were calculated. To capture the performance of the proposed algorithm in varying scenarios a total of 10 scenes were selected from the nuScene lidar dataset with a total of 404 lidar scans, as shown in Table 6-1. The average recall, fps and mIoU for each nuScene were calculated as follows:

$$Recall = \sum_{i=1}^I \frac{TP}{TP + FN}, \text{ where } i = 1, 2, 3, \dots, I. \quad (6.1)$$

$$IoU = \sum_{i=1}^I \frac{TP}{TP + FP + FN} \quad (6.2)$$

$$mIoU = \frac{IoU_g + IoU_{ng}}{2} \quad (6.3)$$

$$fps = \frac{1}{time_{per \ frame}} \quad (6.4)$$

Where TP , FN , and FP represents the True Positive, False Negative, and False Positive for predicted points, I represent all samples in a given scene and IoU_g, IoU_{ng} represents IoU for ground points and IoU for non-ground points for a given scene. For autonomous driving, recall is important to measure the ability of a proposed algorithm to correctly identify all relevant instances of a target class within a dataset. Also, the mean IoU gives us the measure of how the overall predicted classes (ground and non-ground) overlap with the ground truth classes. Therefore, these metrics were used to evaluate and compare the FASTSEG3D approach to other efficient algorithms.

Table 6-1 Description of selected NuScene dataset for evaluating segmentation methods

NuScene	Description
0061	Parked Truck, Construction, Intersection
0103	Pedestrian Traffic, Bike Rack, Cyclist
0553	Intersection Wait, Large Truck, Pedestrian Crossing
0655	Parking Lot, Jaywalker, Bendy Bus
0757	Busy Intersection, Bus, and Bicycle Traffic
0796	Sidewalk Traffic, Overtaking Vehicles, Construction Zone

0916	Bicycle Parking, Pedestrian Traffic, Parked Vehicles
1077	Night Bus Stop, Construction Zone, High Speed
1094	Rainy Night, Pedestrian Traffic, PMD Traffic
1100	Night Pedestrian Traffic, Scooters, Difficult Lighting

6.2.4 FASTSEG3D: REE and WBMF modular performance increase:

The FASTSEG3D modular evaluation begins by showing the performance-to-speed trade-off in Table 6-2, for the fastseg3d modular setup. As referenced in the methodology section [246], the ring elevation map representation of point clouds addresses the limitations of the traditional polar representation for uneven distribution of points over distance. The success of the REE technique is based on the premise that point clouds are evenly distributed. The REE is therefore able to give a good estimation of ground points which are expected to be all points less than the point with the minimum elevation and a certain slope threshold for all points with similar ranges. Furthermore, the success of the WBMF technique is also based on the premise that point clouds are evenly distributed. Therefore, sliding a fixed window size over the ring elevation map image is guaranteed to capture regions of the point clouds and smooth the estimated ground points. Because points within a similar range could include flat terrains that are above the slope threshold, calculating the surface normal of that region and refining the labels according to the surface normal threshold values for that region is also successful as shown in Figure 6-1.

However, the REE module is good for ground removal because its threshold for elevation is fixed. This leads to a high number of false positives and false negatives which is reflected in the recall and mIoU results.

Generally, it is ideal to have a recall and mIoU of above 90% as a very good score for prediction. Figure 6-1 shows an example of how REE suffers from false negatives in uneven terrain.

Table 6-2 The performance-to-speed result for each FASTSEG3D module

FASTSEG3D MODULES	NuScenes									
	0061	0103	0553	0655	0757	0796	0916	1077	1094	1100
<i>fps</i>										
REE	64.80	68.72	70.99	70.09	70.40	73.64	70.52	68.95	70.37	67.75
WBMF	20.29	19.49	24.04	20.07	22.02	22.27	21.51	20.32	22.24	20.11
<i>Recall_g(%)</i>										
REE	94.08	91.37	89.44	91.65	94.34	94.06	93.09	93.68	95.03	93.07
WBMF	99.25	98.62	99.37	97.58	99.77	99.80	98.79	99.17	99.69	99.51
<i>Recall_{ng}(%)</i>										
REE	82.23	88.72	91.97	87.71	89.68	87.91	86.77	85.83	89.79	86.19
WBMF	89.75	93.87	93.08	95.73	94.88	92.59	91.81	91.53	92.66	95.82
<i>mIoU(%)</i>										
REE	78.39	81.61	82.90	81.38	85.63	83.80	82.04	81.90	86.35	81.68
WBMF	87.11	92.23	92.05	91.96	94.40	91.53	91.56	91.11	93.65	92.81

6.2.5 FASTSEG3D versus other proposed algorithms:

The algorithms selected for the comparison were considered based on their similarities to the FastSeg3D algorithm, performance, FPS and their uniqueness as compared to other existing algorithms. These narrowed the selection to the popular RANSAC by Fischler *et al.* [87], the coarse-to-fine grain algorithm by Shen *et al.*[298], and the recent robust concentric zone model (CZM) algorithm approach by Lee *et al.*[297]. To compare other proposed algorithms with FASTSEG3D the parameters of each algorithm are fine-tuned to get the best results for the nuScene dataset. Table 6-3 shows the performance and speed of each algorithm as compared to FASTSG3D. Since this algorithm focuses on efficiently removing ground points from a given point cloud, the *recall_g*, *precision_g*, *fps*, *IoU_g* and the total number of points (in percentage) processed from a given scanned point cloud is

used to compare algorithms. To calculate the number of points processed the percentage of points processed from the given number of scanned point clouds is calculated after filtering out the points close to the ego vehicle which is classified as *vehicle.ego* on the NuScene dataset. Since all algorithms compared do not include a downsampling technique to evenly downsample point clouds a lower value of processed points increases the possibility of omitting important points labelled as objects which could be crucial in object detection or further point processing.

Table 6-3 The performance-to-speed comparison of FASTSEG3D with other algorithms.

Methods	NuScenes									
	0061	0103	0553	0655	0757	0796	0916	1077	1094	1100
<i>Number of points (%)</i>										
[87]	100	100	100	100	100	100	100	100	100	100
[298]	33.44	32.43	38.09	33.20	32.15	36.34	33.72	32.92	32.39	32.99
[297]	100	100	100	100	100	100	100	100	100	100
FASTSEG3D	89.41	88.72	87.65	82.48	82.58	84.96	81.02	81.22	81.50	81.94
<i>fps</i>										
[87]	43.76	48.19	46.81	48.46	52.21	61.52	51.72	46.06	47.45	46.39
[298]	27.21	27.01	25.79	27.20	29.61	28.08	27.19	27.47	28.13	24.49
[297]	5.30	5.71	7.40	5.43	5.91	7.02	6.10	5.63	5.93	5.81
FASTSEG3D	25.29	24.49	29.04	25.07	27.02	27.27	26.51	25.32	27.24	25.11
<i>Recall_g(%)</i>										
[87]	94.32	94.60	98.06	93.60	97.68	98.61	97.48	94.68	98.06	93.02
[298]	95.79	95.19	97.07	90.93	97.26	90.30	95.11	95.72	97.96	95.19
[297]	91.77	91.20	90.62	91.17	96.40	79.71	92.40	93.44	95.73	90.87
FASTSEG3D	99.25	98.62	99.37	97.58	99.77	99.80	98.79	99.17	99.69	99.51
<i>Recall_{ng}(%)</i>										
[87]	88.64	93.32	90.22	97.15	95.51	94.32	93.20	94.16	90.29	94.64
[298]	84.67	91.07	88.25	91.52	91.71	87.22	89.62	89.40	88.58	88.11

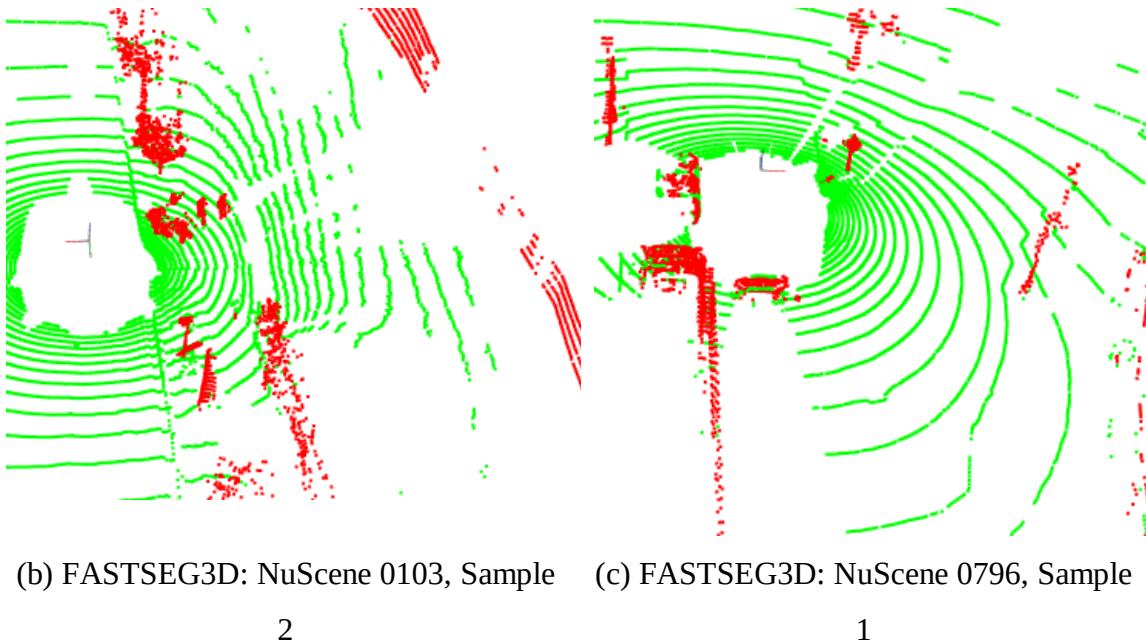
[297]	87.27	93.25	92.03	93.28	94.68	91.37	91.61	90.97	92.12	90.26
FASTSEG3D	89.75	93.87	93.08	95.73	94.88	92.59	91.81	91.53	92.66	95.82
<i>mIoU(%)</i>										
[87]	83.50	88.10	88.72	90.24	93.44	93.74	91.32	89.27	89.71	87.59
[298]	81.07	86.94	84.02	83.47	89.97	79.86	85.88	86.22	87.89	84.87
[297]	80.37	84.89	84.16	84.92	91.45	72.42	85.04	85.46	88.76	82.30
FASTSEG3D	87.11	92.23	92.05	91.96	94.40	91.53	91.56	91.11	93.65	92.81

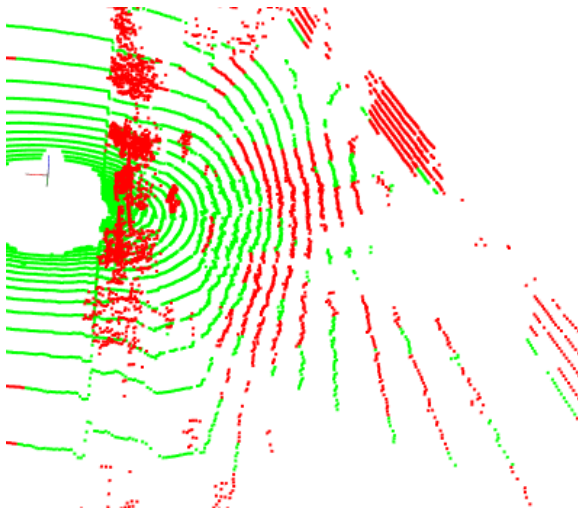
The results from Table 6-3 show how each algorithm compares to FASTSEG3D in effectively predicting ground and non-ground points. Because the REM with REE image was restricted to mapping points with a maximum range of 70 m, this filtered out roughly 15–18% of the input point cloud. The maximum range was set to 70m because this makes up for a reasonable driving range in a real-world scenario. Although all algorithms performed above 90% for the recall of ground points, the recall of non-ground points shows that there is a high possibility of omitting obstacle points which can be a major issue when using these approaches as a preprocessing step towards sensor fusion of 3D reconstruction of fused data.

6.2.6 Discussion:

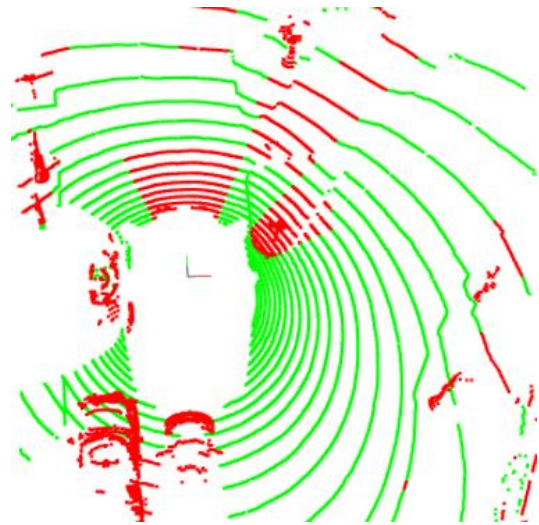
The popular RANSAC by Fischler *et al.* [87] is a good algorithm for the fast removal of ground points but suffers from under-segmentation of uneven terrain. Addressing this problem has been the inspiration for many proposed algorithms. From the compared algorithms, Shen *et al.* [298] proposed a coarse-to-fine grain technique that uses the spatial relationship between regions of low confidence and high confidence to re-classify points in low confidence regions. Although this proposed technique is efficient, its success is dependent on the accuracy of the coarse segmentation. Since the coarse segmentation technique uses a fixed height threshold to classify ground and non-ground points, regions like uneven terrains that have been originally classified as non-ground points with high confidence the fine-grain approach is limited. The proposed robust ground removal approach by Lee *et al.* [297] is also a very efficient technique but to address the limitations

of uneven distribution of points over the polar map representation, the authors proposed a concentric zone model (CZM) to adjust the grid size to compensate for the uneven distribution. The major drawback of this approach is that since the CZM is a handcrafted approach, it has to be fine-tuned when used on a different dataset or different kinds of lidar sensors. For instance, it was developed and tested on the Kitti Data Set which used the Velodyne HDL-64E but when tested on the nuScene dataset which used the Velodyne HDL32E lidar sensor certain regions which were ground points were labelled as non-ground. An example is shown in Figure 6-3.

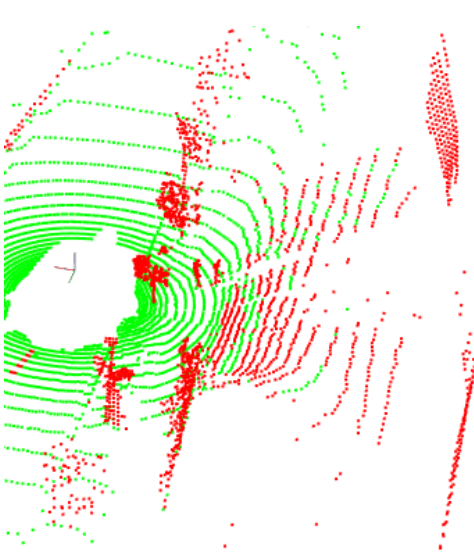




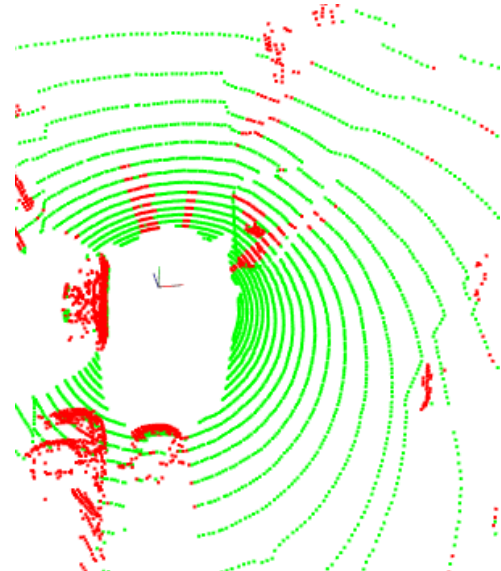
(b) [297]: NuScene 0103, Sample 2



(c) [297]: NuScene 0796, Sample 1



(a) [298]: NuScene 0103, Sample 2



(b) [298]: NuScene 0796, Sample 1

Figure 6-3 Comparison of FASTSEG3D against Shen *et al.*[298] and Lee *et al.*[297] proposed algorithms for ground removal

This explains the low mIoU of less than 90% and why Lee *et al.*[297] had an unusual recall score of ground points of 79% for NuScene 0796 as compared to other recall scores.

6.3 BEVMSF3D PIPELINE:

6.3.1 Experimental setup

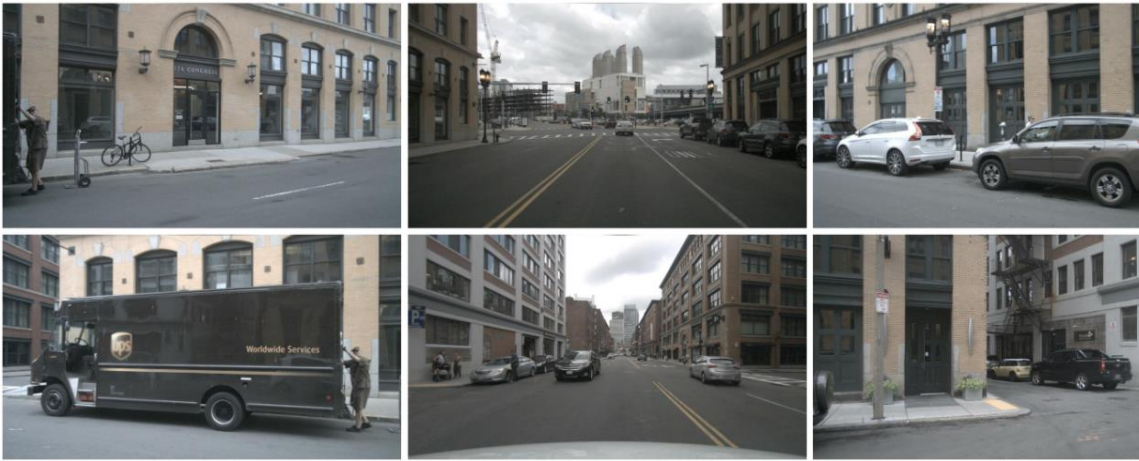
The experiment for BEVMSF3D was performed using the Python standard library, PyTorch library, Albumentation library for the data augmentation, the Timms and efficient-det library, Python-OpenCV, NuScene devkit-tools (for the data loader), Numpy Library, PIL library (for loading image), and matplotlib library for (visualisation) on the Ubuntu 20.04 OS computer. Training and inferencing were conducted using the RTX 4090 graphics processing unit (GPU) and Ryzen 9 5950 central processing unit (CPU).

The experiment was conducted on the large-scale benchmark NuScenes [245,299]. The NuScenes benchmark includes 1000 scenes with images from 6 calibrated surrounding-view cameras and one point cloud. It is the up-to-date popular benchmark for vision-based 3D object detection and BEV semantic segmentation. The scenes are officially split into 700/150/150 scenes for training/validation/testing. There are up to 1.4M annotated 3D bounding boxes for 10 classes: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, barrier, and traffic cone (T.C). A subset of four map categories were selected which can feasibly be estimated from images.

6.3.2 Experimental parameters

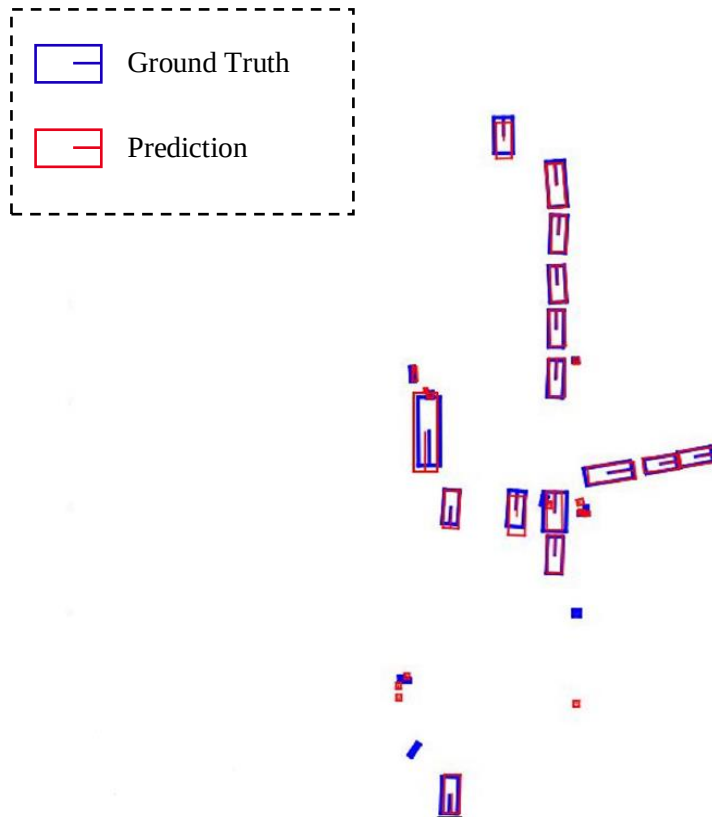
For training, the NuScene dataset was trained using the Timms and efficient-det library along with the coded view transformation algorithm and feature fusion algorithm. The EfficientNet version 1 B3 was used with the BiFPN module [279] as the back backbone. BiFPN was repeated four times for the feature fusion and the feature depth was set to 128. The image size was set to 512 x 512 to suit the training requirements for the EfficientNet-BiFPN backbone. The resampler algorithms BEV dimensions were set to a total size of 70m (i.e. $z_{min}, z_{max} = 1m, 70m$ and $x_{min}, x_{max} = -35m, 35m$ respectively) to conform with that of the FastSeg3D algorithm. To reduce complexity the resolution was set to approximately 0.2m as highlighted in Chapter 5. The focal length for the view transformation algorithm was set to 850 (usually focal length between 600 and 1200 is fine but will determine how close or far the BEV map would look).

The Bayesian Classifier with dropout sampling used was similar to that of [212], the Bayesian decision boundary was set to be greater the 50%, ($p(m_i^c|z_t > 0.5)$). The main difference with [212] is the fact that this study did not include sidewalks and ground classes during training and the Lidar BEV image was included for feature fusion. The uncertainty loss weighting λ was set to 0.001. The network was trained until it converged using SGD with a learning rate of 0.1, batch size of 16, and momentum of 0.9. The predicted occupant (objects) of valid grid cells were used to draw bounding boxes and compared with the ground truth. As shown in Figure 6-4.

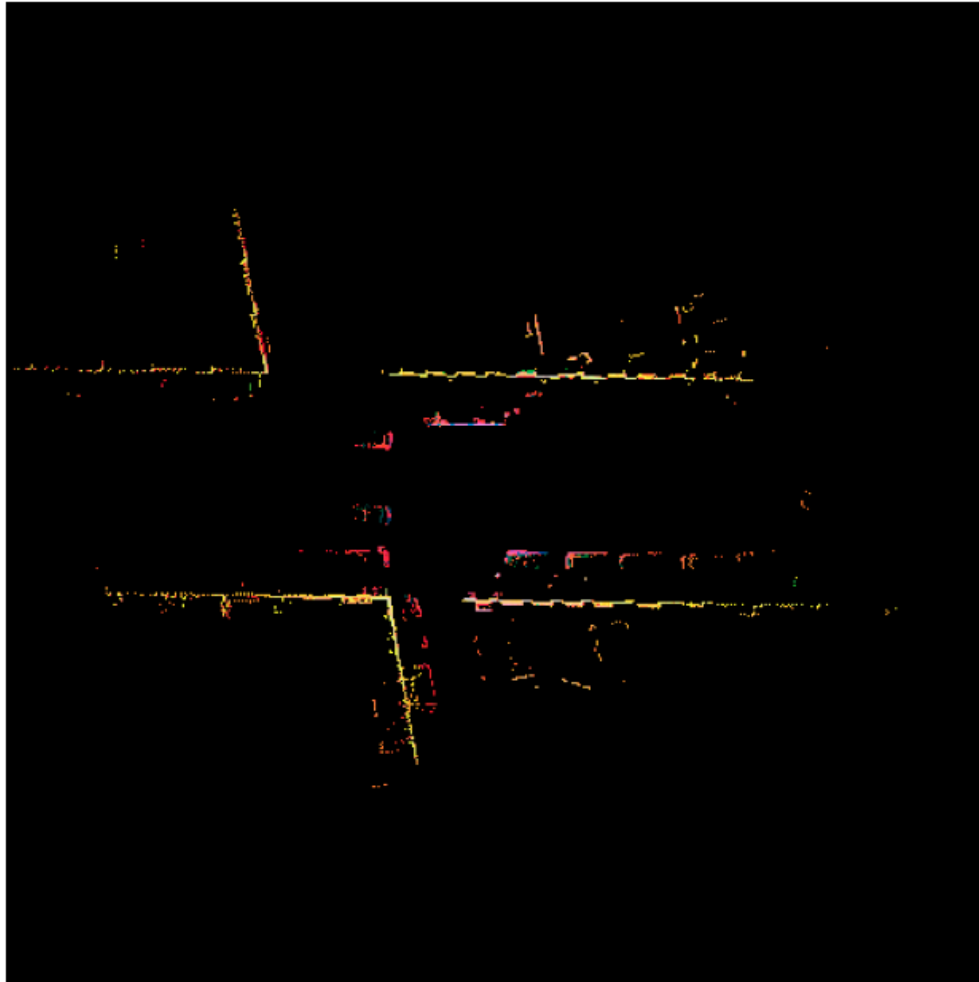


(a) The six (6) viewing cameras from front left to back right.

(b) The equivalent fused BEV semantic



(c) image is represented as prediction boxes.



(d) The lidar BEV image fused with the size camera images.

Figure 6-4 Ground truth (in blue) versus the BEVMSF3D (in red). NuScene Sample

6.3.3 Evaluation metrics

The algorithms contained in the NuScene literature [299] employed baselines focused on each modality. For instance, the Lidar detection baseline employed the PiontPillars algorithm which is an improvement of the PointNet++ architecture [97] focused on Lidar-only detection and segmentation. The image detection baseline employed the Orthographic Feature Transform (OFT) method [300] with a detection head and a non-maximum suppression (NMS) to optimize the combined detection from the 6 camera views. This is an image-only detection algorithm. Furthermore, the panoptic segmentation algorithm

[245] was also based on PointNet++ [97] which is a Lidar-only algorithm and further applied other algorithms like the Hungarian algorithms [301,302] for assigning instances to predefined classes, PointGroup network [303] for clustering of points to achieve a point-wise semantic segmentation of Lidar points in the point cloud. These algorithms are strictly focused on accuracy rather than real-time performance and do not include the fusion of different sensor modalities.

The NuScenes detection score (NDS), mean average precision (mAP) and Latency (ms) were the evaluation metrics used to evaluate the multisensor fusion model, BEVMSF3D. See Table 6-4 for more details.

The results for BEVMSF3D on detection tasks are shown in Table 6-4. The table presents the prediction result for test and validation NDS and mAP and the Latency (ms) for similar algorithms. All modalities compared are for Lidar and Camera to ensure consistency.

Table 6-4 Test and validation results comparison on the NuScene dataset

Method	mAP (test)	NDS (test)	mAP (val)	NDS (val)	Latency (ms)
PointPainting [304]	-	-	65.8	69.6	185.8
MVP [305]	66.4	70.5	66.1	70.0	187.1
TransFusion [306]	68.9	71.6	67.5	71.3	156.6
BEVFusion [208]	70.2	72.9	68.5	71.4	119.2
BEVMSF3D	47.8	49.5	47.2	48.6	56.2
BEVMSF3D + augmentation	62.9	65.4	62.1	64.3	88.7

The mAP test results of all the algorithms for each of the 10 classes: car, truck (tk.), bus, trailer (tr.), construction vehicle (cvh.), pedestrian (ped.), motorcycle (mtc.), bicycle (bike), barrier (bar.), and traffic cone (tc.), was compared with that of BEVMSF3D as shown in table 6-5.

Table 6-5 Comparison of mAP test results across 10 classes

Method	mAP (test)									
	car	tk.	cvh.	Bus	tr.	bar.	mtc.	bike	ped.	tc.
PointPainting [304]	77.9	35.8	15.8	36.2	37.3	60.2	41.5	24.1	73.3	62.4
MVP [305]	86.8	58.5	26.1	67.4	57.3	74.8	70.0	49.3	89.1	85.0
TransFusion [306]	87.1	60.0	33.1	68.3	60.8	78.1	73.6	52.9	88.4	86.7
BEVFusion [208]	88.1	60.9	34.4	69.3	62.1	78.2	72.2	52.2	89.2	85.2
BEVMSF3D	64.3	35.2	11.4	31.7	33.4	42.1	44.1	33.2	63.1	44.6
BEVMSF3D + View Tx	79.4	43.7	14.2	39.1	41.6	51.8	54.3	40.8	78.4	55.3

6.3.4 Discussion

To improve speed and align with the ground-removed BEV lidar image, the ground and walkway classes were not included in the training and testing. Furthermore, semantic objects were represented as bounding boxes for a minimalistic representation of objects around the ego vehicle, see Figure 6-4. The red coloured bounding boxes indicated the predicted boxes and the blue indicated the ground truth boxes. The NuScene map was used as a reference to estimate the distance of the ego vehicle from each object in the BEV semantic image.

To begin the evaluation, the results of NDS and mAP for BEVMSF3D were conducted to show the need for image augmentation (see Section 4.3.4) and the view transformation module (See Section 5.2.3.2). This showed an impressive improvement in the initial results.

As shown in Table 6-4, the BEVMSF3D algorithm was able to achieve a real-time semantic BEV map with an acceptable result of approximately 17.8 fps without image augmentation

and approximately 11.3 fps with augmentation at inferencing time. This is a huge improvement compared to existing algorithms with the best inferencing time of about 8.4 fps [208]. Although compared to [208] the NDS and mAP scores were not as good, there is still room for improvement, as these are the results that show proof of concept. Furthermore, the augmentation of images showed a significant improvement in results with an average of above 30% (percentage) increase in NDS and mAP scores.

The evaluation conducted on the individual class detection showed that on average the BEVMSF3D algorithm performed okay as compared with other state-of-the-art algorithms. Particularly, it performed better than [304] in most of its mAP test scores, this could be due to the strong LiDAR point association with the images. However, BEVMSF3D still struggled in some test scores. This could be due to the drop in performance in environments with poor weather conditions. An impressive aspect of this presented model was its improvement of above 22% (percentage) increase to its initial score without the view transformation module. This is understandable because of the learnable convolution module.

6.4 CONCLUSION

In this chapter, the FastSeg3D and multi-sensor fusion architecture results were presented and discussed. The experimental environment was highlighted, and the results were compared with other algorithms. The FastSeg3D result showed a segmentation speed of 25 fps on average when tested on an Intel single-core i5 7th generation laptop as compared to other algorithms. It recorded an impressive ground recall of over 97 per cent and over 89 per cent for non-ground points recall.

The multi-sensor fusion experiment was conducted on a Ryzen 9 5950 CPU and RTX 4090 GPU. Results showed that the inclusion of the image augmentation preprocessing step greatly improved the mean average precision (mAP) score and NuScenes detection score (NDS) by over 30% as well and the inclusion of the view transformation module improved the mAP by over 22%. BEV fusion and view transformation inferencing speed was at a

maximum frame per second score of 11.3 fps when set up for best performance i.e. when the view transformation module and augmentation steps are included.

CHAPTER 7: CONCLUSION

7.1 CONCLUSION

In this study, a comprehensive literature review was conducted to establish a strong foundation and identify key research gaps in the field of IoT-based multi-sensor fusion for ACAMDs. A novel Fast and Adaptive Ground Segmentation Method of 3D Point Cloud, called FastSeg3D was developed as a first step towards multi-sensor fusion for ACAMDs. This algorithm consisted of two modules: Range Elevation Estimation (REE) and Window-based Model Fitting (WBMF) for a coarse to fine-grain segmentation of 3D point clouds and to effectively remove ground points as a real-time preprocessing step for the multi-sensor fusion architecture. This algorithm was also developed to address the current limitations of the traditional point cloud segmentation techniques which include the segmentation of point cloud for complex terrains in real-time.

The study went further to propose an efficient image augmentation preprocessing step as a technique for achieving automatic extrinsic calibration of image and point clouds in the multi-sensor fusion architecture. The multi-sensor fusion architecture was then proposed to effectively fuse LiDAR point clouds and the images taken from six viewing camera angles in sequence. This was achieved by using a well-known computer vision (CV) technique called inverse perspective mapping (IPM) and learnable (convolutional) layers to transform the front and back viewing cameras into equivalent bird's eye view (BEV). The resulting feature images were then fused with the LiDAR point cloud after the point clouds had been converted into it equivalent BEV image. A Bayesian classifier was used to perform a predictive occupancy grid semantic mapping probability of occurrence using Baye's theorem concept to locate the position of objects relative to the ego vehicle or robot in the BEV grid.

The study went further to propose an IoT framework suitable for ACAMDs for communicating and storing semantic BEV map videos of locations that the ACAMDs in a closed network have visited. This framework was a fog computing network that used the

MQTT protocol for low power consumption and low latency communication. To further improve packet latency, an algorithm that converts semantic BEV videos to its equivalent JSON file consisting of ego vehicle or robot position and orientation and all detected objects' positions relative to the ego vehicle or robot for every timestamp was proposed.

Finally, experimental results of the novel FastSeg3D and multi-sensor fusion architecture were presented, and evaluation metrics were used to highlight its performance and latency. The results were then discussed to show proof of concept and the prospects of the novel algorithm and multi-sensor fusion architecture. The proposed techniques of this study have the potential to significantly improve the accuracy and efficiency of point cloud processing and multi-sensor fusion for various applications, such as assisted mobility, autonomous navigation, SLAM and scene reconstruction.

7.2 LIMITATIONS AND FUTURE WORKS

The limitations and future works of this study are as follows:

- The FastSeg3D approach to REE still requires a fixed threshold to segment ground points and even though there are expected values considered as ground and non-ground points, a more adaptive approach can be investigated.
- FastSeg3D was carried out on a single thread and further optimisation can be investigated to improve the fps result.
- Interpretable machine learning techniques can also be explored to further optimise FastSeg3D.
- Future works may include the evaluation of the performance of the proposed IoT Framework against state-of-the-art literature.
- An extensive analysis of the multi-sensor fusion architecture is needed to test its performance in camera-only and lidar-only BEV representation including its performance under harsh weather conditions to understand the level of contribution offered by each sensor and their limitations. Also, test cases like IPM performance and depth-based unprojection are needed.

- Optimisation of the Multi-sensor fusion to further improve its training and inferencing time as well as achieve deployable versions suitable for electronics.

REFERENCES

1. Oladele, D. A.; Markus, E. D.; Abu-Mahfouz, A. M. Adaptability of Assistive Mobility Devices and the Role of the Internet of Medical Things: Comprehensive Review. *JMIR Rehabil Assist Technol* **2021**, *8*.
2. Sadeghi-Niaraki, A. Internet of Thing (IoT) review of review: Bibliometric overview since its foundation. *Future Generation Computer Systems* **2023**, *143*, 361–377.
3. Luperto, M.; Monroy, J.; Moreno, F. A.; Lunardini, F.; Renoux, J.; Krpic, A.; Galindo, C.; Ferrante, S.; Basilico, N.; Gonzalez-Jimenez, J.; et al. Seeking at-home long-term autonomy of assistive mobile robots through the integration with an IoT-based monitoring system. *Rob Auton Syst* **2023**, *161*, 104346.
4. Mordini, E.; Nierling, L.; Wolbring, G.; Maia, M. J.; Bratan, T.; Capari, L.; Čas, J.; Krieger-Lamina, J.; Kukk, P. Assistive technologies for people with disabilities . Part II, Current and emerging technologies. **2018**.
5. Elmannai, W.; Elleithy, K. Sensor-Based Assistive Devices for Visually-Impaired People: Current Status, Challenges, and Future Directions. *Sensors* **2017**, Vol. 17, Page 565 **2017**, *17*, 565.
6. Cowan, R. E.; Fregly, B. J.; Boninger, M. L.; Chan, L.; Rodgers, M. M.; Reinkensmeyer, D. J. Recent trends in assistive technology for mobility. *J Neuroeng Rehabil* **2012**, *9*, 1–8.
7. Temene, N.; Sergiou, C.; Georgiou, C.; Vassiliou, V. A Survey on Mobility in Wireless Sensor Networks. *Ad Hoc Networks* **2022**, *125*, 102726.
8. Guler, U.; Tufan, T. B.; Chakravarti, A.; Jin, Y.; Ghovanloo, M. Implantable and Wearable Sensors for Assistive Technologies. *Encyclopedia of Sensors and Biosensors: Volume 1-4, First Edition* **2023**, *1–4*, 449–473.
9. Mangal, N. K.; Tiwari, A. K. A review of the evolution of scientific literature on technology-assisted approaches using RGB-D sensors for musculoskeletal health monitoring. *Comput Biol Med* **2021**, *132*, 104316.
10. Daniel, O. A.; Markus, E. D.; Abu-Mahfouz, A. M. Internet of things based multi-sensor fusion for assistive mobility devices. *2021 Conference on Information*

Communications Technology and Society, ICTAS 2021 - Proceedings **2021**, 115–120.

11. Li, N.; Yang, Y.; Li, G.; Yang, T.; Wang, Y.; Chen, W.; Yu, P.; Xue, X.; Zhang, C.; Wang, W.; et al. Multi-Sensor Fusion-Based Mirror Adaptive Assist-as-Needed Control Strategy of a Soft Exoskeleton for Upper Limb Rehabilitation. *IEEE Transactions on Automation Science and Engineering* **2022**.

12. Kim, T. hoon; Ramos, C.; Mohammed, S. Smart City and IoT. *Future Generation Computer Systems* **2017**, 76, 159–162.

13. Taketomi, T.; Uchiyama, H.; Ikeda, S. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications* **2017**, 9, 1–11.

14. Catarinucci, L.; De Donno, D.; Mainetti, L.; Palano, L.; Patrono, L.; Stefanizzi, M. L.; Tarricone, L. An IoT-Aware Architecture for Smart Healthcare Systems. *IEEE Internet Things J* **2015**, 2, 515–526.

15. Alam, M. M.; Malik, H.; Khan, M. I.; Pardy, T.; Kuusik, A.; Le Moullec, Y. A survey on the roles of communication technologies in IoT-Based personalized healthcare applications. *IEEE Access* **2018**, 6, 36611–36631.

16. Liang, M.; Yang, B.; Chen, Y.; Hu, R.; Urtasun, R. Multi-Task Multi-Sensor Fusion for 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2019; pp. 7345–7353.

17. Gravina, R.; Alinia, P.; Ghasemzadeh, H.; Fortino, G. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion* **2017**, 35, 68–80.

18. Cordts, P.; Cotten, S. R.; Qu, T.; Bush, T. R. Mobility challenges and perceptions of autonomous vehicles for individuals with physical disabilities. *Disabil Health J* **2021**, 14, 101131.

19. Pellichero, A.; Kenyon, L. K.; Best, K. L.; Lamontagne, M. E.; Lavoie, M. D.; Sorita, É.; Routhier, F. Relationships between cognitive functioning and powered mobility device use: A scoping review. *Int J Environ Res Public Health* **2021**, 18, 12467.

20. Zhuang, Y.; Sun, X.; Li, Y.; Huai, J.; Hua, L.; Yang, X.; Cao, X.; Zhang, P.; Cao, Y.; Qi, L.; et al. Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches. *Information Fusion* **2023**, *95*, 62–90.
21. Yeong, D. J.; Velasco-hernandez, G.; Barry, J.; Walsh, J. Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. *Sensors* **2021**, *Vol. 21*, Page 2140 **2021**, *21*, 2140.
22. Aguilera, A. A.; Brena, R. F.; Mayora, O.; Molino-Minero-re, E.; Trejo, L. A. Multi-Sensor Fusion for Activity Recognition—A Survey. *Sensors* **2019**, *Vol. 19*, Page 3808 **2019**, *19*, 3808.
23. Kline, A.; Wang, H.; Li, Y.; Dennis, S.; Hutch, M.; Xu, Z.; Wang, F.; Cheng, F.; Luo, Y. Multimodal machine learning in precision health: A scoping review. *npj Digital Medicine* **2022** *5:1* **2022**, *5*, 1–14.
24. Cui, G.; Zhang, W.; Xiao, Y.; Yao, L.; Fang, Z. Cooperative Perception Technology of Autonomous Driving in the Internet of Vehicles Environment: A Review. *Sensors* **2022**, *Vol. 22*, Page 5535 **2022**, *22*, 5535.
25. Kozlov, V. V. Sensor Fusion and Measurement Errors. *Studies in Systems, Decision and Control* **2022**, *419*, 149–156.
26. Lambrecht, S.; Nogueira, S. L.; Bortole, M.; Siqueira, A. A. G.; Terra, M. H.; Rocon, E.; Pons, J. L. Inertial Sensor Error Reduction through Calibration and Sensor Fusion. *Sensors* **2016**, *Vol. 16*, Page 235 **2016**, *16*, 235.
27. Pang, Z. Technologies and Architectures of the Internet-of-Things (IoT) for Health and well-being. Doctoral Thesis, KTH – Royal Institute of Technology: Stockholm, Sweden, 2013.
28. Nebot, E. M. Sensors Used for Autonomous Navigation. *Advances in Intelligent Autonomous Systems* **1999**, 135–156.
29. Bo, F.; Yerebakan, M.; Dai, Y.; Wang, W.; Li, J.; Hu, B.; Gao, S. IMU-Based Monitoring for Assistive Diagnosis and Management of IoHT: A Review. *Healthcare* **2022**, *Vol. 10*, Page 1210 **2022**, *10*, 1210.

30. Murray-Bergquist, L.; Bernauer, F.; Igel, H. Characterization of Six-Degree-of-Freedom Sensors for Building Health Monitoring. *Sensors* **2021**, Vol. 21, Page 3732 **2021**, 21, 3732.
31. Cabric, M. Security Inventions: Security Technologies in Business Analytics. *From Corporate Security to Commercial Force* **2017**, 187–199.
32. Kleusberg, A.; Langley, R. THE LIMITATIONS OF GPS. **1990**.
33. Ayala, R.; Mohd, T. K. Sensors in Autonomous Vehicles: A Survey. *Journal of Autonomous Vehicles and Systems* **2021**, 1.
34. Sun, R.; Yang, Y.; Chiang, K. W.; Duong, T. T.; Lin, K. Y.; Tsai, G. J. Robust I_pGPS/VO Integration for Vehicle Navigation in GNSS Degraded Urban Areas. *IEEE Sens J* **2020**, 20, 10110–10122.
35. Liu, X.; Wen, W.; Hsu, L. T. GLIO: Tightly-Coupled GNSS/LiDAR/IMU Integration for Continuous and Drift-free State Estimation of Intelligent Vehicles in Urban Areas. *IEEE Transactions on Intelligent Vehicles* **2023**.
36. Pandey, M.; Mishra, G. Types of sensor and their applications, advantages, and disadvantages. *Advances in Intelligent Systems and Computing* **2019**, 814, 791–804.
37. Jain, D.; Raman, S. Deep over and Under Exposed Region Detection. *Communications in Computer and Information Science* **2021**, 1378 CCIS, 34–45.
38. Bigas, M.; Cabruja, E.; Forest, J.; Salvi, J. Review of CMOS image sensors. *Microelectronics J* **2006**, 37, 433–451.
39. Li, Y.; Ibanez-Guzman, J. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. *IEEE Signal Process Mag* **2020**, 37, 50–61.
40. Berns, K.; Köpper, A.; Schürmann, B. Sensor data processing. *Lecture Notes in Electrical Engineering* **2021**, 732, 227–253.
41. Tawakuli, A.; Kaiser, D.; Engel, T. Synchronized Preprocessing of Sensor Data. *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020* **2020**, 3522–3531.

42. Kaur, H.; Koundal, D.; Kadyan, V. Image Fusion Techniques: A Survey. *Archives of Computational Methods in Engineering* **2021**, *28*, 4425–4447.
43. Boulahia, S. Y.; Amamra, A.; Madi, M. R.; Daikh, S. Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Mach Vis Appl* **2021**, *32*, 1–18.
44. Singh, A. D.; Sandha, S. S.; Garcia, L.; Srivastava, M. Radhar: Human activity recognition from point clouds generated through a millimeter-wave radar. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM* **2019**, *19*, 51–56.
45. Sun, Y.; Zhang, H.; Huang, Z.; Liu, B. R2P: A Deep Learning Model from mmWave Radar to Point Cloud. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2022**, *13529 LNCS*, 329–341.
46. Yang, L.; Wang, C.; Wang, L. Autonomous UAVs landing site selection from point cloud in unknown environments. *ISA Trans* **2022**, *130*, 610–628.
47. Liu, W.; Sun, J.; Li, W.; Hu, T.; Wang, P. Deep Learning on Point Clouds and Its Application: A Survey. *Sensors* **2019**, *Vol. 19, Page 4188* **2019**, *19*, 4188.
48. Wang, R.; Peethambaran, J.; Chen, D. LiDAR Point Clouds to 3-D Urban Models: A Review. *IEEE J Sel Top Appl Earth Obs Remote Sens* **2018**, *11*, 606–627.
49. Schumann, O.; Hahn, M.; Dickmann, J.; Wöhler, C. Semantic Segmentation on Radar Point Clouds. *2018 21st International Conference on Information Fusion, FUSION 2018* **2018**, 2179–2186.
50. Li, Y.; Ma, L.; Zhong, Z.; Liu, F.; Chapman, M. A.; Cao, D.; Li, J. Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Trans Neural Netw Learn Syst* **2021**, *32*, 3412–3432.
51. Koo, K. M.; Cha, E. Y. Image recognition performance enhancements using image normalization. *Human-centric Computing and Information Sciences* **2017**, *7*, 1–11.

52. Saponara, S.; Elhanashi, A. Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance. *Lecture Notes in Electrical Engineering* **2022**, 866 LNEE, 10–17.
53. Huang, S. C.; Hoang, Q. V.; Le, T. H.; Peng, Y. T.; Huang, C. C.; Zhang, C.; Fung, B. C. M.; Cheng, K. H.; Huang, S. W. An Advanced Noise Reduction and Edge Enhancement Algorithm. *Sensors* **2021**, Vol. 21, Page 5391 **2021**, 21, 5391.
54. Boukhayma, A.; Peizerat, A.; Enz, C. Noise Reduction Techniques and Scaling Effects towards Photon Counting CMOS Image Sensors. *Sensors* **2016**, Vol. 16, Page 514 **2016**, 16, 514.
55. Wang, C.; Dai, D.; Xia, S.; Liu, Y.; Wang, G. One-Stage Deep Edge Detection Based on Dense-Scale Feature Fusion and Pixel-Level Imbalance Learning. *IEEE Transactions on Artificial Intelligence* **2022**.
56. Xiao, G.; Bavirisetti, D. P.; Liu, G.; Zhang, X. Pixel-Level Image Fusion. *Image Fusion* **2020**, 21–101.
57. Sonka, M.; Hlavac, V.; Boyle, R. Image pre-processing. *Image Processing, Analysis and Machine Vision* **1993**, 56–111.
58. Kumar, G.; Bhatia, P. K. A detailed review of feature extraction in image processing systems. *International Conference on Advanced Computing and Communication Technologies, ACCT* **2014**, 5–12.
59. Kaur, H.; Koundal, D.; Kadyan, V. Image Fusion Techniques: A Survey. *Archives of Computational Methods in Engineering* **2021**, 28, 4425–4447.
60. Krig, S. Image Pre-Processing. *Computer Vision Metrics* **2016**, 35–74.
61. Vargas, J.; Alsweiss, S.; Toker, O.; Razdan, R.; Santos, J. An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions. *Sensors* **2021**, Vol. 21, Page 5397 **2021**, 21, 5397.
62. Xu, Y.; Tong, X.; Stilla, U. Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Autom Constr* **2021**, 126, 103675.

63. Han, X. F.; Jin, J. S.; Wang, M. J.; Jiang, W.; Gao, L.; Xiao, L. A review of algorithms for filtering the 3D point cloud. *Signal Process Image Commun* **2017**, 57, 103–112.
64. Liu, W.; Sun, J.; Li, W.; Hu, T.; Wang, P. Deep Learning on Point Clouds and Its Application: A Survey. *Sensors* 2019, Vol. 19, Page 4188 **2019**, 19, 4188.
65. Nguyen, A.; Le, B. 3D point cloud segmentation: A survey. *IEEE Conference on Robotics, Automation and Mechatronics, RAM - Proceedings* **2013**, 225–230.
66. Che, E.; Jung, J.; Olsen, M. J. Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review. *Sensors* 2019, Vol. 19, Page 810 **2019**, 19, 810.
67. Jiang, C.; Meng, Q.; Xu, B.; Gao, W.; Wu, P.; Guan, L.; Li, Z.; Xu, X.; Zhang, L.; Yang, J.; et al. A Review of Multi-Sensor Fusion SLAM Systems Based on 3D LIDAR. *Remote Sensing* 2022, Vol. 14, Page 2835 **2022**, 14, 2835.
68. Placed, J. A.; Strader, J.; Carrillo, H.; Atanasov, N.; Indelman, V.; Carlone, L.; Castellanos, J. A. A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers. *IEEE Transactions on Robotics* **2022**.
69. Su, Y. T.; Bethel, J.; Hu, S. Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications. *ISPRS Journal of Photogrammetry and Remote Sensing* **2016**, 113, 59–74.
70. Vo, A. V.; Truong-Hong, L.; Laefer, D. F.; Bertolotto, M. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* **2015**, 104, 88–100.
71. Zeng, W.; Gevers, T. 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2017**, 11131 LNCS, 314–330.
72. Greenspan, M.; Yurick, M. Approximate k-d tree search for efficient ICP. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM* **2003**, 2003-January, 442–448.

73. Meagher, D. Geometric modeling using octree encoding. *Computer Graphics and Image Processing* **1982**, 19, 129–147.
74. Zeng, W.; Gevers, T. 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*; 2018; pp. 0–0.
75. Pinkham, R.; Zeng, S.; Zhang, Z. QuickNN: Memory and Performance Optimization of k-d Tree Based Nearest Neighbor Search for 3D Point Clouds. *Proceedings - 2020 IEEE International Symposium on High Performance Computer Architecture, HPCA 2020* **2020**, 180–192.
76. Schauer, J.; Nüchter, A. Collision detection between point clouds using an efficient k-d tree implementation. *Advanced Engineering Informatics* **2015**, 29, 440–458.
77. Brown, R. A. Building a Balanced k-d Tree in $O(kn \log n)$ Time. *Journal of Computer Graphics Techniques (JCGT)* **2014**, 4, 50–68.
78. Chen, Y.; Zhou, L.; Tang, Y.; Singh, J. P.; Bouguila, N.; Wang, C.; Wang, H.; Du, J. Fast neighbor search by using revised k-d tree. *Inf Sci (N Y)* **2019**, 472, 145–162.
79. Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; et al. Stanley: The robot that won the DARPA Grand Challenge. *J Field Robot* **2006**, 23, 661–692.
80. Himmelsbach, M.; Hundelshausen, F. V.; Wuensche, H. J. Fast segmentation of 3D point clouds for ground vehicles. *IEEE Intelligent Vehicles Symposium, Proceedings* **2010**, 560–565.
81. Premebida, C.; Sousa, J.; Garrote, L.; Nunes, U. Polar-Grid Representation and Kriging-Based 2.5D Interpolation for Urban Environment Modelling. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* **2015**, 2015-October, 1234–1239.
82. Zhang, Y.; Zhou, Z.; David, P.; Yue, X.; Xi, Z.; Gong, B.; Foroosh, H. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision*

- and Pattern Recognition (CVPR)*; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, 2020; pp. 9601–9610.
83. Ran, H.; Liu, J.; Wang, C. Surface Representation for Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2022; pp. 18942–18952.
 84. Serafin, J.; Olson, E.; Grisetti, G. Fast and robust 3D feature extraction from sparse point clouds. *IEEE International Conference on Intelligent Robots and Systems* **2016**, 2016-November, 4105–4112.
 85. Rummelhard, L.; Paigwar, A.; Negre, A.; Laugier, C. Ground estimation and point cloud segmentation using SpatioTemporal Conditional Random Field. *IEEE Intelligent Vehicles Symposium, Proceedings* **2017**, 1105–1110.
 86. Cheng, Z.; Ren, G.; Zhang, Y. Ground Segmentation Algorithm Based on 3D Lidar Point Cloud. **2018**, 16–21.
 87. Fischler, M. A.; Bolles, R. C. Random sample consensus. *Commun ACM* **1981**, 24, 381–395.
 88. Chen, T.; Dai, B.; Wang, R.; Liu, D. Gaussian-Process-Based Real-Time Ground Segmentation for Autonomous Land Vehicles. *Journal of Intelligent & Robotic Systems* 2013 76:3 **2013**, 76, 563–582.
 89. Chen, T.; Dai, B.; Liu, D.; Song, J. Sparse Gaussian process regression based ground segmentation for autonomous land vehicles. *Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015* **2015**, 3993–3998.
 90. Liu, K.; Wang, W.; Tharmarasa, R.; Wang, J.; Zuo, Y. Ground surface filtering of 3D point clouds based on hybrid regression technique. *IEEE Access* **2019**, 7, 23270–23284.
 91. Zhang, M.; Morris, D. D.; Fu, R. Ground Segmentation Based on Loopy Belief Propagation for Sparse 3D Point Clouds. *Proceedings - 2015 International Conference on 3D Vision, 3DV 2015* **2015**, 615–622.
 92. Moosmann, F.; Pink, O.; Stiller, C. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. *IEEE Intelligent Vehicles Symposium, Proceedings* **2009**, 215–220.

93. Yang, B.; Dong, Z. A shape-based segmentation method for mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* **2013**, *81*, 19–30.
94. Zhang, L.; Zhang, L.; Mao, Q.; Zou, Q.; Zhang, P.; Feng, S.; Ochieng, W. Motion Field Estimation for a Dynamic Scene Using a 3D LiDAR. *Sensors* **2014**, Vol. 14, Pages 16672-16691 **2014**, *14*, 16672–16691.
95. Zermas, D.; Izzat, I.; Papanikolopoulos, N. Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications. *Proc IEEE Int Conf Robot Autom* **2017**, 5067–5073.
96. Goga, S.; Nedevschi, S. An approach for segmenting 3D LiDAR data using multi-volume grid structures. *Proceedings - 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing, ICCP 2017* **2017**, 309–315.
97. Qi, C. R.; Yi, L.; Su, H.; Guibas, L. J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Adv Neural Inf Process Syst* **2017**, 2017-December, 5100–5109.
98. Qi, C. R.; Liu, W.; Wu, C.; Su, H.; Guibas, L. J. Frustum PointNets for 3D Object Detection from RGB-D Data. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2017**, 918–927.
99. Zhu, X. X.; Tuia, D.; Mou, L.; Xia, G. S.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geosci Remote Sens Mag* **2017**, *5*, 8–36.
100. Xie, Y.; Tian, J.; Zhu, X. X. Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation. *IEEE Geosci Remote Sens Mag* **2019**, *8*, 38–59.
101. Qi, C. R.; Su, H.; Mo, K.; Guibas, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* **2016**, 2017-January, 77–85.
102. Pang, Y.; Wang, W.; Tay, F. E. H.; Liu, W.; Tian, Y.; Yuan, L. *Masked Autoencoders for Point Cloud Self-supervised Learning*; 2022.

103. Xu, M.; Zhang, J.; Zhou, Z.; Xu, M.; Qi, X.; Qiao, Y. Learning Geometry-Disentangled Representation for Complementary Understanding of 3D Object Point Cloud. *35th AAAI Conference on Artificial Intelligence, AAAI 2021* **2020**, 4A, 3056–3064.
104. Ma, X.; Qin, C.; You, H.; Ran, H.; Fu, Y. Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework. In *ICLR 2022*; ICLR 2022, 2022.
105. Garnelo, M.; Shanahan, M. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Curr Opin Behav Sci* **2019**, 29, 17–23.
106. Poux, F.; Billen, R. Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised Geometric and Relationship Featuring vs Deep Learning Methods. *ISPRS International Journal of Geo-Information* 2019, Vol. 8, Page 213 **2019**, 8, 213.
107. Poux, F. U. de L.-Ul. > D. de géographie > U. de G.-T. et géométrie; Mattes, C.; Kobbelt, L. Unsupervised segmentation of indoor 3D point cloud: application to object-based classification. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **2020**, XLIV–4, 111–118.
108. Xue, F.; Lu, W.; Chen, Z.; Webster, C. J. From LiDAR point cloud towards digital twin city: Clustering city objects based on Gestalt principles. *ISPRS Journal of Photogrammetry and Remote Sensing* **2020**, 167, 418–431.
109. Maheshan, C. M.; Prasanna Kumar, H. Performance of image pre-processing filters for noise removal in transformer oil images at different temperatures. *SN Appl Sci* **2020**, 2, 1–7.
110. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2001**, 1.

111. Poornachandra, S.; Naveena, C.; Aradhya, M. Intensity normalization—a critical pre-processing step for efficient brain tumor segmentation in MR images. *Advances in Intelligent Systems and Computing* **2018**, 672, 885–893.
112. Friston, K. J.; Ashburner, J.; Frith, C. D.; Poline, J. -B; Heather, J. D.; Frackowiak, R. S. J. Spatial registration and normalization of images. *Hum Brain Mapp* **1995**, 3, 165–189.
113. Reinhold, J. C.; Dewey, B. E.; Carass, A.; Prince, J. L. Evaluating the impact of intensity normalization on MR image synthesis. <https://doi.org/10.1117/12.2513089> **2019**, 10949, 890–898.
114. Konovalenko, I. A.; Kokhan, V. V.; Nikolaev, D. P. Optimal affine image normalization approach for optical character recognition. *Computer Optics* **2021**, 45, 90–100.
115. Monteiro, H. A.; Brito, A. V. de; Melcker, E. U. K. Image normalization in embedded systems. *J Real Time Image Process* **2021**, 18, 2469–2478.
116. Delisle, P. L.; Anctil-Robitaille, B.; Desrosiers, C.; Lombaert, H. Realistic image normalization for multi-Domain segmentation. *Med Image Anal* **2021**, 74, 102191.
117. Gopal, S.; Patro, K.; Kumar Sahu, K. Normalization: A Preprocessing Stage. *IARJSET* **2015**, 20–22.
118. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, 86, 2278–2323.
119. Taylor, L.; Nitschke, G. Improving Deep Learning with Generic Data Augmentation. *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018* **2019**, 1542–1547.
120. Inoue, H. Data Augmentation by Pairing Samples for Images Classification. **2018**.
121. Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. Random Erasing Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence* **2020**, 34, 13001–13008.

122. DeVries, T.; Taylor, G. W. Dataset Augmentation in Feature Space. *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings* **2017**.
123. Shorten, C.; Khoshgoftaar, T. M. A survey on Image Data Augmentation for Deep Learning. *J Big Data* **2019**, 6, 1–48.
124. Moreno-Barea, F. J.; Strazzera, F.; Jerez, J. M.; Urda, D.; Franco, L. Forward Noise Adjustment Scheme for Data Augmentation. *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018* **2019**, 728–734.
125. Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. Random Erasing Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence* **2020**, 34, 13001–13008.
126. Naveed, H.; Anwar, S.; Hayat, M.; Javed, K.; Mian, A. Survey: Image Mixing and Deleting for Data Augmentation. **2021**.
127. Inoue, H. Data Augmentation by Pairing Samples for Images Classification. *arXiv; Computer Vision and Pattern Recognition* **2018**.
128. Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; Yoo, Y. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) 2019*, 6023–6032.
129. DeVries, T.; Taylor, G. W. Dataset Augmentation in Feature Space. *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings* **2017**.
130. Li, B.; Wu, F.; Lim, S. N.; Belongie, S.; Weinberger, K. Q. On Feature Normalization and Data Augmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2020**, 12378–12387.
131. Lemley, J.; Bazrafkan, S.; Corcoran, P. Smart Augmentation Learning an Optimal Data Augmentation Strategy. *IEEE Access* **2017**, 5, 5858–5869.
132. Yarats, D.; Fergus, R.; Kostrikov, I. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. *ICLR 2021 - 9th International Conference on Learning Representations* **2020**.

133. Wong, S. C.; Gatt, A.; Stamatescu, V.; McDonnell, M. D. Understanding Data Augmentation for Classification: When to Warp? *2016 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2016* **2016**.
134. Perez, L.; Wang, J. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. **2017**.
135. Qiu, Z.; Martínez-Sánchez, J.; Arias-Sánchez, P.; Rashdi, R. External multi-modal imaging sensor calibration for sensor fusion: A review. *Information Fusion* **2023**, 97, 101806.
136. Khurana, A.; Nagla, K. S. Extrinsic Calibration Methods for Laser Range Finder and Camera: A Systematic Review. *Mapan - Journal of Metrology Society of India* **2021**, 36, 669–690.
137. Lu, L.; Zhang, C.; Liu, Y.; Zhang, W.; Xia, Y. IEEE 1588-Based general and precise time synchronization method for multiple sensors. *IEEE International Conference on Robotics and Biomimetics, ROBIO 2019* **2019**, 2427–2432.
138. Elson, J.; Römer, K. Wireless sensor networks. *ACM SIGCOMM Computer Communication Review* **2003**, 33, 149–154.
139. Blair, W. D.; Rice, T. R.; Alouani, A. T.; Xia, P. Asynchronous data fusion for target tracking with a multitasking radar and optical sensor. <https://doi.org/10.1117/12.45699> **1991**, 1482, 234–245.
140. Yuan, S.; Li, Y.; Chen, Y.; Wang, C. Time Synchronization Accuracy Verification for Multi-Sensor System. *Proceedings - 2021 International Conference on Artificial Intelligence, Big Data and Algorithms, CAIBDA 2021* **2021**, 21–24.
141. Huang, G.; Zomaya, A. Y.; Delicato, F. C.; Pires, P. F. Long term and large scale time synchronization in wireless sensor networks. *Comput Commun* **2014**, 37, 77–91.
142. Zhang, Q.; Pless, R. Extrinsic calibration of a camera and laser range finder (improves camera calibration). *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* **2004**, 3, 2301–2306.
143. Wang, C. C. Extrinsic Calibration of a Vision Sensor Mounted on a Robot. *IEEE Transactions on Robotics and Automation* **1992**, 8, 161–175.

144. Rodriguez F., S. A.; Frémont, V.; Bonnifait, P. Extrinsic calibration between a multi-layer lidar and a camera. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems* **2008**, 214–219.
145. Gong, X.; Lin, Y.; Liu, J. 3D LIDAR-Camera Extrinsic Calibration Using an Arbitrary Trihedron. *Sensors* **2013**, Vol. 13, Pages 1902-1918 **2013**, 13, 1902–1918.
146. Lenz, R. K. A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration. *IEEE Transactions on Robotics and Automation* **1989**, 5, 345–358.
147. Shiu, Y. C.; Ahmad, S. FINDING THE MOUNTING POSITION OF A SENSOR BY SOLVING A HOMOGENEOUS TRANSFORM EQUATION OF THE FORM AX equals XB . **1987**, 1666–1671.
148. Kim, E. S.; Park, S. Y. Extrinsic Calibration between Camera and LiDAR Sensors by Matching Multiple 3D Planes. *Sensors* **2020**, Vol. 20, Page 52 **2019**, 20, 52.
149. Bergou, E. H.; Diouane, Y.; Kungurtsev, V. Convergence and Complexity Analysis of a Levenberg–Marquardt Algorithm for Inverse Problems. *J Optim Theory Appl* **2020**, 185, 927–944.
150. Carrera, G.; Angeli, A.; Davison, A. J. SLAM-based automatic extrinsic calibration of a multi-camera rig. *Proc IEEE Int Conf Robot Autom* **2011**, 2652–2659.
151. Sim, S.; Sock, J.; Kwak, K. Indirect Correspondence-Based Robust Extrinsic Calibration of LiDAR and Camera. *Sensors* **2016**, Vol. 16, Page 933 **2016**, 16, 933.
152. Guindel, C.; Beltran, J.; Martin, D.; Garcia, F. Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* **2018**, 2018-March, 1–6.
153. Beltrán, J.; Guindel, C.; de la Escalera, A.; García, F. Automatic Extrinsic Calibration Method for LiDAR and Camera Sensor Setups. *IEEE Transactions on Intelligent Transportation Systems* **2021**, 23, 17677–17689.
154. Davison, A. J.; Reid, I. D.; Molton, N. D.; Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Trans Pattern Anal Mach Intell* **2007**, 29, 1052–1067.

155. Moghadam, P.; Bosse, M.; Zlot, R. Line-based extrinsic calibration of range and image sensors. *Proc IEEE Int Conf Robot Autom* **2013**, 3685–3691.
156. Zhou, L.; Li, Z.; Kaess, M. Automatic Extrinsic Calibration of a Camera and a 3D LiDAR Using Line and Plane Correspondences. *IEEE International Conference on Intelligent Robots and Systems* **2018**, 5562–5569.
157. Pandey, G.; McBride, J. R.; Savarese, S.; Eustice, R. M. Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information. *J Field Robot* **2015**, 32, 696–722.
158. Yang, Z.; Shen, S. Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration. *IEEE Transactions on Automation Science and Engineering* **2017**, 14, 39–51.
159. Iyer, G.; Ram, R. K.; Murthy, J. K.; Krishna, K. M. CalibNet: Geometrically Supervised Extrinsic Calibration using 3D Spatial Transformer Networks. *IEEE International Conference on Intelligent Robots and Systems* **2018**, 1110–1117.
160. Lv, X.; Wang, S.; Ye, D. CFNet: LiDAR-Camera Registration Using Calibration Flow Network. *Sensors* **2021**, Vol. 21, Page 8112 **2021**, 21, 8112.
161. Alatisse, M. B.; Hancke, G. P. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access* **2020**, 8, 39830–39846.
162. Durrant-Whyte, H.; Henderson, T. C. Multisensor Data Fusion. *Springer Handbook of Robotics* **2008**, 585–610.
163. Tang, Q.; Liang, J.; Zhu, F. A comparative review on multi-modal sensors fusion based on deep learning. *Signal Processing* **2023**, 213, 109165.
164. Chatila, R.; Laumond, J. P. Position referencing and consistent world modeling for mobile robots. *Proc IEEE Int Conf Robot Autom* **1985**, 138–145.
165. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot Autom Mag* **2006**, 13, 99–108.
166. Bailey, T.; Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robot Autom Mag* **2006**, 13, 108–117.

167. Mur-Artal, R.; Montiel, J. M. M.; Tardos, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* **2015**, *31*, 1147–1163.
168. Taketomi, T.; Uchiyama, H.; Ikeda, S. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSI Transactions on Computer Vision and Applications* **2017**, *9*, 1–11.
169. Zhang, H. Y.; Lin, W. M.; Chen, A. X. Path Planning for the Mobile Robot: A Review. *Symmetry* **2018**, *Vol. 10, Page 450* **2018**, *10*, 450.
170. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path planning and trajectory planning algorithms: A general overview. *Mechanisms and Machine Science* **2015**, *29*, 3–27.
171. Durrant-whyte, H.; Henderson, T. C. Multisensor Data Fusion. In *Springer Handbook of Robotics*; Siciliano, B.; Khatib, O., Eds.; Springer, Cham, 2016; pp. 867–896.
172. Mor, B.; Garhwal, S.; Kumar, A. A Systematic Review of Hidden Markov Models and Their Applications. *Archives of Computational Methods in Engineering* **2021**, *28*, 1429–1448.
173. Levinson, S. E.; Rabiner, L. R.; Sondhi, M. M. An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition. *Bell System Technical Journal* **1983**, *62*, 1035–1074.
174. Rabiner, L. R.; Juang, B. H. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine* **1986**, *3*, 4–16.
175. Sutton, C.; McCallum, A. An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning* **2010**, *4*, 267–373.
176. Lafferty, J.; McCallum, A.; Pereira, F. C. N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data 2001.
177. Wan, E. A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC 2000* **2000**, 153–158.

178. Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* **1960**, 82, 35–45.
179. Julier, S. J.; Uhlmann, J. K. New extension of the Kalman filter to nonlinear systems. <https://doi.org/10.1117/12.280797> **1997**, 3068, 182–193.
180. Basar, T. A New Approach to Linear Filtering and Prediction Problems. *Control Theory* **2010**.
181. Handschin, J. E.; Mayne, D. Q. Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering†. <http://dx.doi.org/10.1080/00207176908905777> **2007**, 9, 547–559.
182. Gordon, N. J.; Salmond, D. J.; Smith, A. F. M. Novel approach to nonlinear/non-gaussian Bayesian state estimation. *IEE Proceedings, Part F: Radar and Signal Processing* **1993**, 140, 107–113.
183. Wan, E. A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC 2000* **2000**, 153–158.
184. Del Moral, P.; Doucet, A.; Jasra, A. Sequential Monte Carlo Samplers. *J R Stat Soc Series B Stat Methodol* **2006**, 68, 411–436.
185. Del Moral, P. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* **1997**, 325, 653–658.
186. Elfring, J.; Torta, E.; van de Molengraft, R. Particle Filters: A Hands-On Tutorial. *Sensors 2021, Vol. 21, Page 438* **2021**, 21, 438.
187. Lykou, R.; Tsaklidis, G. Particle Filtering: A Priori Estimation of Observational Errors of a State-Space Model with Linear Observation Equation. *Mathematics 2021, Vol. 9, Page 1445* **2021**, 9, 1445.
188. He, X.; Aloï, D. N.; Li, J. Probabilistic Multi-Sensor Fusion Based Indoor Positioning System on a Mobile Device. *Sensors 2015, Vol. 15, Pages 31464-31481* **2015**, 15, 31464–31481.
189. Yang, M.; Yang, E. Two-stage multi-sensor fusion positioning system with seamless switching for cooperative mobile robot and manipulator system. *Int J Intell Robot Appl* **2023**, 7, 275–290.

190. Jia, Y.; Luo, H.; Zhao, F.; Jiang, G.; Li, Y.; Yan, J.; Jiang, Z.; Wang, Z. Lvio-Fusion: A Self-adaptive Multi-sensor Fusion SLAM Framework Using Actor-critic Method. *IEEE International Conference on Intelligent Robots and Systems* **2021**, 286–293.
191. Li, S.; Li, X.; Wang, H.; Zhou, Y.; Shen, Z. Multi-GNSS PPP/INS/Vision/LiDAR tightly integrated system for precise navigation in urban environments. *Information Fusion* **2023**, 90, 218–232.
192. Obando-Ceron, J. S.; Romero-Cano, V.; Monteiro, S. Probabilistic multi-modal depth estimation based on camera–LiDAR sensor fusion. *Mach Vis Appl* **2023**, 34, 1–16.
193. Dong, X.; Zhuang, B.; Mao, Y.; Liu, L. Radar Camera Fusion via Representation Learning in Autonomous Driving. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* **2021**, 1672–1681.
194. Zhuang, Z.; Li, R.; Jia, K.; Wang, Q.; Li, Y.; Tan, M. Perception-Aware Multi-Sensor Fusion for 3D LiDAR Semantic Segmentation. *Proceedings of the IEEE International Conference on Computer Vision* **2021**, 16260–16270.
195. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3D object detection network for autonomous driving. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* **2017**, 2017-January, 6526–6534.
196. Kim, A.; Osep, A.; Leal-Taixé, L. EagerMOT: 3D Multi-Object Tracking via Sensor Fusion. *Proc IEEE Int Conf Robot Autom* **2021**, 2021-May, 11315–11321.
197. Drews, F.; Feng, D.; Faion, F.; Rosenbaum, L.; Ulrich, M.; Glaser, C. DeepFusion: A Robust and Modular 3D Object Detector for Lidars, Cameras and Radars. *IEEE International Conference on Intelligent Robots and Systems* **2022**, 2022-October, 560–567.
198. Prakash, A.; Chitta, K.; Geiger, A. Multi-Modal Fusion Transformer for End-to-End Autonomous Driving. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2021**, 7073–7083.

199. Chen, X.; Zhang, T.; Wang, Y.; Wang, Y.; Zhao, H. FUTR3D: A Unified Sensor Fusion Framework for 3D Detection. **2022**.
200. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR 2021 - 9th International Conference on Learning Representations* **2020**.
201. Touvron, H.; Cord, M.; El-Nouby, A.; Verbeek, J.; Jégou, H. Three things everyone should know about Vision Transformers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2022**, 13684 LNCS, 497–515.
202. Chen, S.; Cheng, T.; Wang, X.; Meng, W.; Zhang, Q.; Liu, W. Efficient and Robust 2D-to-BEV Representation Learning via Geometry-guided Kernel Transformer. **2022**.
203. Yang, C.; Chen, Y.; Tian, H.; Tao, C.; Zhu, X.; Zhang, Z.; Huang, G.; Li, H.; Qiao, Y.; Lu, L.; et al. BEVFormer v2: Adapting Modern Image Backbones to Bird's-Eye-View Recognition via Perspective Supervision. **2022**.
204. Liu, Y.; Yan, J.; Jia, F.; Li, S.; Gao, A.; Wang, T.; Zhang, X.; Sun, J. PETRv2: A Unified Framework for 3D Perception from Multi-Camera Images. **2022**.
205. Li, Y.; Huang, B.; Chen, Z.; Cui, Y.; Liang, F.; Shen, M.; Liu, F.; Xie, E.; Sheng, L.; Ouyang, W.; et al. Fast-BEV: A Fast and Strong Bird's-Eye View Perception Baseline. **2023**.
206. Huang, J.; Huang, G.; Zhu, Z.; Ye, Y.; Du, D.; Robotics, P. BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View. **2021**.
207. Liang, T.; Xie, H.; Yu, K.; Xia, Z.; Lin, Z.; Wang, Y.; Tang, T.; Wang, B.; Tang, Z. BEVFusion: A Simple and Robust LiDAR-Camera Fusion Framework. **2022**.
208. Liu, Z.; Tang, H.; Amini, A.; Yang, X.; Mao, H.; Rus, D.; Han, S. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. **2022**.

209. Huang, J.; Huang, G. BEVDet4D: Exploit Temporal Cues in Multi-camera 3D Object Detection. **2022**.
210. Li, Y.; Ge, Z.; Yu, G.; Yang, J.; Wang, Z.; Shi, Y.; Sun, J.; Li, Z. BEVDepth: Acquisition of Reliable Depth for Multi-view 3D Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence* **2022**, *37*, 1477–1485.
211. Phillion, J.; Fidler, S. Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2020**, *12359 LNCS*, 194–210.
212. Roddick, T.; Cipolla, R. Predicting semantic map representations from images using pyramid occupancy networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2020**, 11135–11144.
213. Li, Z.; Yu, Z.; Wang, W.; Anandkumar, A.; Lu, T.; Alvarez, J. M. FB-BEV: BEV Representation from Forward-Backward View Transformations. **2023**.
214. Li, Z.; Yu, Z.; Austin, D.; Fang, M.; Lan, S.; Kautz, J.; Alvarez, J. M. FB-OCC: 3D Occupancy Prediction based on Forward-Backward View Transformation. **2023**.
215. Reiher, L.; Lampe, B.; Eckstein, L. A Sim2Real Deep Learning Approach for the Transformation of Images from Multiple Vehicle-Mounted Cameras to a Semantically Segmented Image in Bird's Eye View. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020* **2020**.
216. Yin, R.; Cheng, Y.; Wu, H.; Song, Y.; Yu, B.; Niu, R. FusionLane: Multi-Sensor Fusion for Lane Marking Semantic Segmentation Using Deep Neural Networks. *IEEE Transactions on Intelligent Transportation Systems* **2022**, *23*, 1543–1553.
217. Zhou, B.; Krahenbuhl, P. Cross-view Transformers for real-time Map-view Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2022**, *2022-June*, 13750–13759.
218. Liu, J.; Wang, G.; Duan, L. Y.; Abdiyeva, K.; Kot, A. C. Skeleton-Based Human Action Recognition with Global Context-Aware Attention LSTM Networks. *IEEE Transactions on Image Processing* **2018**, *27*, 1586–1599.

219. Liu, J.; Wang, G.; Hu, P.; Duan, L.-Y.; Kot, A. C. Global Context-Aware Attention LSTM Networks for 3D Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017; 2017; pp. 1647–1656.
220. Tucker, R.; Snavely, N. Single-View View Synthesis with Multiplane Images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2020**, 548–557.
221. Srinivasan, P. P.; Mildenhall, B.; Tancik, M.; Barron, J. T.; Tucker, R.; Snavely, N. Lighthouse: Predicting Lighting Volumes for Spatially-Coherent Illumination. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2020**, 8077–8086.
222. Gosala, N.; Valada, A. Bird’s-Eye-View Panoptic Segmentation Using Monocular Frontal View Images. *IEEE Robot Autom Lett* **2022**, 7, 1968–1975.
223. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library*; “O’Reilly Media, Inc.,” 2008.
224. Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; Geiger, A. Occupancy networks: Learning 3D reconstruction in function space. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2019**, 2019-June, 4455–4465.
225. Mallot, H. A.; Bülthoff, H. H.; Little, J. J.; Bohrer, S. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biol Cybern* **1991**, 64, 177–185.
226. Elfes, A. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer (Long Beach Calif)* **1989**, 22, 46–57.
227. Adame, T.; Carrascosa-Zamacois, M.; Bellalta, B. Time-Sensitive Networking in IEEE 802.11be: On the Way to Low-Latency WiFi 7. *Sensors* **2021**, Vol. 21, Page 4954 **2021**, 21, 4954.
228. Derhamy, H.; Eliasson, J.; Delsing, J. IoT Interoperability - On-Demand and Low Latency Transparent Multiprotocol Translator. *IEEE Internet Things J* **2017**, 4, 1754–1763.

229. Mishra, Z.; Acharya, B. High throughput and low area architectures of secure IoT algorithm for medical image encryption. *Journal of Information Security and Applications* **2020**, *53*, 102533.
230. Li, Y.; Liu, X.; Chi, Z. Chiron: Concurrent high throughput communication for IoT devices. *MobiSys 2018 - Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services* **2018**, 204–216.
231. Rajab, H.; Cinkler, T.; Bouguera, T. IoT scheduling for higher throughput and lower transmission power. *Wireless Networks* **2021**, *27*, 1701–1714.
232. Morabito, R.; Cozzolino, V.; Ding, A. Y.; Beijar, N.; Ott, J. Consolidate IoT Edge Computing with Lightweight Virtualization. *IEEE Netw* **2018**, *32*, 102–111.
233. De Donno, M.; Tange, K.; Dragoni, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access* **2019**, *7*, 150936–150948.
234. Cao, K.; Liu, Y.; Meng, G.; Sun, Q. An Overview on Edge Computing Research. *IEEE Access* **2020**, *8*, 85714–85728.
235. Gusev, M.; Dustdar, S. Going back to the roots - The evolution of edge computing, an IoT perspective. *IEEE Internet Comput* **2018**, *22*, 5–15.
236. Hassan, N.; Gillani, S.; Ahmed, E.; Yaqoob, I.; Imran, M. The Role of Edge Computing in Internet of Things. *IEEE Communications Magazine* **2018**, *56*, 110–115.
237. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J* **2016**, *3*, 637–646.
238. Pang, Z.; Sun, L.; Wang, Z.; Tian, E.; Yang, S. A Survey of Cloudlet Based Mobile Computing. *Proceedings - 2015 International Conference on Cloud Computing and Big Data, CCBBD 2015* **2016**, 268–275.
239. Iorga, M.; Feldman, L.; Barton, R.; Martin, M. J.; Goren, N. S.; Mahmoudi, C. Fog Computing Conceptual Model. **2018**.
240. Yi, S.; Li, C.; Li, Q. A survey of fog computing: Concepts, applications and issues. *Mobidata 2015 - Proceedings of the 2015 Workshop on Mobile Big Data, co-located with MobiHoc 2015* **2015**, 2015-June, 37–42.

241. Stojmenovic, I.; Wen, S.; Huang, X.; Luan, H. An overview of Fog computing and its security issues. *Concurr Comput* **2016**, *28*, 2991–3005.
242. Li, Y.; Ibanez-Guzman, J. Lidar for Autonomous Driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Process Mag* **2020**, *37*, 50–61.
243. Mehendale, N.; Neoge, S. Review on Lidar Technology. *SSRN Electronic Journal* **2020**.
244. Raj, T.; Hashim, F. H.; Huddin, A. B.; Ibrahim, M. F.; Hussain, A. A Survey on LiDAR Scanning Mechanisms. *Electronics* **2020**, *Vol. 9, Page 741* **2020**, *9*, 741.
245. Fong, W. K.; Mohan, R.; Hurtado, J. V.; Zhou, L.; Caesar, H.; Beijbom, O.; Valada, A. Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. *IEEE Robot Autom Lett* **2021**, *7*, 3795–3802.
246. Huang, W.; Liang, H.; Lin, L.; Wang, Z.; Wang, S.; Yu, B.; Niu, R. A Fast Point Cloud Ground Segmentation Approach Based on Coarse-To-Fine Markov Random Field. *IEEE Transactions on Intelligent Transportation Systems* **2022**, *23*, 7841–7854.
247. Foix, S.; Alenyà, G.; Torras, C. Lock-in time-of-flight (ToF) cameras: A survey. *IEEE Sens J* **2011**, *11*, 1917–1926.
248. Mirzaei, F. M. Extrinsic and intrinsic sensor calibration. **2013**.
249. Deen, J.; Mukhopadhyay, S.; Chen, Y.; Morais, S.; Cennamo, N.; Lee, J.; Sun, N.; Qiu, Q.; Fan, Z.; Li, T.; et al. Intrinsic Calibration of Multi-Beam LiDARs for Agricultural Robots. *Remote Sensing* **2022**, *Vol. 14, Page 4846* **2022**, *14*, 4846.
250. Li, C.; Lubecke, V. M.; Boric-Lubecke, O.; Lin, J. A review on recent advances in doppler radar sensors for noncontact healthcare monitoring. *IEEE Trans Microw Theory Tech* **2013**, *61*, 2046–2060.
251. Hasch, J.; Topak, E.; Schnabel, R.; Zwick, T.; Weigel, R.; Waldschmidt, C. Millimeter-wave technology for automotive radar sensors in the 77 GHz frequency band. *IEEE Trans Microw Theory Tech* **2012**, *60*, 845–860.
252. Toa, M.; Whitehead, A. Ultrasonic Sensing Basics. **2021**.

253. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans Pattern Anal Mach Intell* **2000**, 22, 1330–1334.
254. Bouguet, J. Y.; Perona, P. 3D photography on your desk. *Proceedings of the IEEE International Conference on Computer Vision* **1998**, 43–50.
255. Tsai, R. Y. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal on Robotics and Automation* **1987**, 3, 323–344.
256. Heikkila, J. Sensitivity analysis of camera calibration. <https://doi.org/10.1117/12.58797> **1992**, 1615, 29–38.
257. Hemayed, E. E. A survey of camera self-calibration. *Proceedings - IEEE Conference on Advanced Video and Signal Based Surveillance, AVSS 2003* **2003**, 351–357.
258. Maybank, S. J.; Faugeras, O. D. A theory of self-calibration of a moving camera. *Int J Comput Vis* **1992**, 8, 123–151.
259. Faugeras, O. D.; Luong, Q. T.; Maybank, S. J. Camera self-calibration: Theory and experiments. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **1992**, 588 LNCS, 321–334.
260. Richard Hartley; Andrew Zisserman Multiple View Geometry in Computer Vision - Richard Hartley, Andrew Zisserman - Google Books <https://books.google.co.za/books?id=si3R3Pfa98QC&pg=PA153&dq=pinhole+in+title:%22Multiple+View+Geometry+in+computer+vision%22&redir_esc=y#v=one+page&q=pinhole%20intitle%3A%22Multiple%20View%20Geometry%20in%20computer%20vision%22&f=false> (accessed Aug 29, 2023).
261. Kirkpatrick, L. D.; Francis, G. E. Physics : a world view. **2007**, 11.
262. Fernandez, L.; Avila, V.; Goncalves, L. A Generic Approach for Error Estimation of Depth Data from (Stereo and RGB-D) 3D Sensors. **2017**.
263. Camera calibration With OpenCV — OpenCV 2.4.13.7 documentation <https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html> (accessed Aug 30, 2023).

264. Heikkila, J.; Silven, O. Four-step camera calibration procedure with implicit image correction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **1997**, 1106–1112.
265. Conrady, A. E. Decentred Lens-Systems. *Mon Not R Astron Soc* **1919**, 79, 384–390.
266. Brown, D. . C. Close-range camera calibration. *Photogrammetric Engineering* **1971**, 37, 855–866.
267. Brown, D. . C. Decentering Distortion of Lenses. *Photogrammetric Engineering and Remote Sensin* **1966**, 32.
268. Willson, R. G. Modeling and calibration of automated zoom lenses. <https://doi.org/10.1117/12.189130> **1994**, 2350, 170–186.
269. Kumar, R.; Hanson, A. R. Robust Methods for Estimating Pose and a Sensitivity Analysis. *CVGIP: Image Understanding* **1994**, 60, 313–342.
270. Sutherland, I. E. Three-Dimensional Data Input by Tablet. *Proceedings of the IEEE* **1974**, 62, 453–461.
271. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-View 3D Object Detection Network for Autonomous Driving. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* **2016**, 2017-January, 6526–6534.
272. Simon, M.; Milz, S.; Amende, K.; Gross, H.-M. Complex-YOLO: Real-time 3D Object Detection on Point Clouds. **2018**.
273. Tan, M.; Pang, R.; Le, Q. V. EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2019**, 10778–10787.
274. Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning. In *Deep Learning*; MIT Press, 2016.
275. Biscione, V.; Bowers, J. Learning Translation Invariance in CNNs. **2020**.
276. Quiroga, F.; Ronchetti, F.; Lanzarini, L.; Bariviera, A. F. Revisiting data augmentation for rotational invariance in convolutional neural networks. *Advances in Intelligent Systems and Computing* **2020**, 894, 127–141.

277. Pei, S.-C.; Lin, C.-N. Image normalization for pattern recognition.
278. Tan, M.; Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *36th International Conference on Machine Learning, ICML 2019* **2019**, 2019-June, 10691–10700.
279. Tan, M.; Pang, R.; Le, Q. V. EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2019**, 10778–10787.
280. Lin, T. Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* **2017**, 2017-January, 936–944.
281. Ghiasi, G.; Lin, T. Y.; Le, Q. V. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2019**, 2019-June, 7029–7038.
282. Mallot, H. A.; Bülthoff, H. H.; Little, J. J.; Bohrer, S. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biol Cybern* **1991**, 64, 177–185.
283. Bertozzi, M.; Broggi, A.; Fascioli, A. Stereo inverse perspective mapping: theory and applications. *Image Vis Comput* **1998**, 16, 585–590.
284. Wei, X.; Yan, Y.; Guo, S.; Qiu, X.; Qi, F. Secure Data Sharing: Blockchain-Enabled Data Access Control Framework for IoT. *IEEE Internet Things J* **2022**, 9, 8143–8153.
285. Figueredo, K.; Seed, D.; Wang, C. A Scalable, Standards-Based Approach for IoT Data Sharing and Ecosystem Monetization. *IEEE Internet Things J* **2022**, 9, 5645–5652.
286. Kasrin, N.; Benabbas, A.; Elmamooz, G.; Nicklas, D.; Steuer, S.; Sünkel, M. Data-sharing markets for integrating IoT data processing functionalities. *CCF Transactions on Pervasive Computing and Interaction* **2021**, 3, 76–93.

287. Herrero, R. MQTT-SN, CoAP, and RTP in wireless IoT real-time communications. *Multimed Syst* **2020**, 26, 643–654.
288. Ghotbou, A.; Khansari, M. Comparing application layer protocols for video transmission in IoT low power lossy networks: an analytic comparison. *Wireless Networks* **2021**, 27, 269–283.
289. Sultana, T.; Wahid, K. A. Choice of Application Layer Protocols for Next Generation Video Surveillance Using Internet of Video Things. *IEEE Access* **2019**, 7, 41607–41624.
290. Mishra, B.; Kertesz, A. The use of MQTT in M2M and IoT systems: A survey. *IEEE Access* **2020**, 8, 201071–201086.
291. Quincozes, S.; Emilio, T.; Kazienko, J. MQTT protocol: Fundamentals, tools and future directions. *IEEE Latin America Transactions* **2019**, 17, 1439–1448.
292. Jafarnejad Ghomi, E.; Masoud Rahmani, A.; Nasih Qader, N. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications* **2017**, 88, 50–71.
293. Eisenbud, D. E.; Yi, C.; Contavalli, C.; Smith, C.; Kononov, R.; Mann-Hielscher, E.; Cilingiroglu, A.; Cheyney, B.; Shang, W.; Eisenbud, D. E.; et al. *Maglev: A Fast and Reliable Software Network Load Balancer*; 2016.
294. Rahman, M.; Iqbal, S.; Gao, J. Load balancer as a service in cloud computing. *Proceedings - IEEE 8th International Symposium on Service Oriented System Engineering, SOSE 2014* **2014**, 204–211.
295. Kurdi, H.; Thayananthan, V. A Multi-Tier MQTT Architecture with Multiple Brokers Based on Fog Computing for Securing Industrial IoT. *Applied Sciences* **2022**, Vol. 12, Page 7173 **2022**, 12, 7173.
296. Diro, A. A.; Chilamkurti, N.; Veeraraghavan, P. Elliptic curve based cybersecurity schemes for publish-subscribe internet of things. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST* **2017**, 199, 258–268.
297. Lee, S.; Lim, H.; Myung, H. Patchwork++: Fast and Robust Ground Segmentation Solving Partial Under-Segmentation Using 3D Point Cloud. *IEEE*

International Conference on Intelligent Robots and Systems **2022**, 2022-October, 13276–13283.

298. Shen, Z.; Liang, H.; Lin, L.; Wang, Z.; Huang, W.; Yu, J. Fast Ground Segmentation for 3D LiDAR Point Cloud Based on Jump-Convolution-Process. *Remote Sensing* **2021**, Vol. 13, Page 3239 **2021**, 13, 3239.

299. Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A multimodal dataset for autonomous driving. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2019**, 11618–11628.

300. Roddick, T.; Kendall, A.; Cipolla, R. Orthographic Feature Transform for Monocular 3D Object Detection. *30th British Machine Vision Conference 2019, BMVC 2019* **2018**.

301. Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* **1955**, 2, 83–97.

302. Kuhn, H. W. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly* **1956**, 3, 253–258.

303. Jiang, L.; Zhao, H.; Shi, S.; Liu, S.; Fu, C. W.; Jia, J. PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2020**, 4866–4875.

304. Vora, S.; Lang, A. H.; Helou, B.; Beijbom, O. Pointpainting: Sequential fusion for 3D object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2020**, 4603–4611.

305. Yin, T.; Zhou, X.; Krahenbuhl, P. Multimodal Virtual Point 3D Detection. *Adv Neural Inf Process Syst* **2021**, 20, 16494–16507.

306. Bai, X.; Hu, Z.; Zhu, X.; Huang, Q.; Chen, Y.; Fu, H.; Tai, C. L. TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* **2022**, 2022-June, 1080–1089.

APPENDIX

Code 1 Range elevation estimation (REE) snippet

```
FASTSEG3D::FASTSEG3D(pcl::PointCloud<pcl::PointXYZIRL>::Ptr cloud_in, bool nusc) {
    // Timer timer("Initialize FastSeg");
    std::cout << "Initializing FastSeg ..... \n";

    _cloud = cloud_in;
    if(nusc) { _img_height = 32, _img_width = 1024; }
    else { _img_height = 64, _img_width = 2048; }

    Elevation th;
    th.Nz = 100., th.cloud_id = -1, th.label = 2, th.ring = -1;
    th.e_th = 100., th.mean_z = 0., th.min_svd = 0., th.dist_to_plane = 100.;
    th.slope = -M_PI, th.Ndist_to_origin = 100.;
    _elevationMap.resize(_img_width, std::vector<Elevation>(int(_max_r), th));

    _ringImage = cv::Mat::zeros(_img_height, _img_width, CV_8UC3);

    // map cloud to polar ring
    for (size_t i = 0; i < _cloud->points.size(); ++i) {
        float r = getXYRange(_cloud->points[i].x, _cloud->points[i].y);
        if(r < _min_r || r > _max_r)
            continue;

        int angle = round((_img_width - 1) * getAngle(_cloud->points[i].x, _cloud->points[i].y));
        int ring = _cloud->points[i].ring;
        // set _elevationMap
        int range = int(_max_r - r); // int(r - _max_r);
        th.label = 0, th.ring = ring, th.cloud_id = i;
        // set a elevation value for range image
        th.e_th = std::min(_elevationMap[angle][range].e_th, _cloud->points[i].z);
        _elevationMap[angle][range] = th;
        // label all value ring image pixels as ground
        _ringImage.at<cv::Vec3b>(ring, angle) = cv::Vec3b(0,255,0);
    }

    // estimate minimum ground height for each pixel
    float temp_e = 0.;
    for (size_t row = 0; row < _elevationMap.size(); ++row) {
        // adjacent points
        for (size_t col = 0; col < _elevationMap[0].size(); ++col) {
            if (col != 0)
                _elevationMap[row][col].e_th = std::min(_elevationMap[row][col].e_th, temp_e +
_elev_th);
            temp_e = _elevationMap[row][col].e_th;
        }
    }

    // update _ringImage labels and _elevationMap labels.
    for (size_t row = 0; row < _elevationMap.size(); ++row) {
        for (size_t col = 0; col < _elevationMap[0].size(); ++col) {
            Elevation cur = _elevationMap[row][col];
            float ring = cur.ring;
            int id = cur.cloud_id;
            float elev_th = cur.e_th + _elev_th;
            if(id == -1)
                continue;
            if((_cloud->points[id].z >= (elev_th))) {
                _elevationMap[row][col].label = 1;
                _ringImage.at<cv::Vec3b>(ring, row) = cv::Vec3b(0,0,255);
            }
        }
    }
}
```

Code 2 Window-based model fitting (WBMF) snippet

```
void FASTSEG3D::getWindowBaseParams(const int &wind_size) {
    setWindows(wind_size);
    int reg_idx = 0;
    std::vector<WindowIndex> region_vals(0);
    std::vector<WindowIndex> temp(0);
    std::vector<WindowIndex> candidate_vals(0);
    Eigen::Vector3f normals, mean;
    float min_svd, dist;
    std::map<float, WindowIndex> temp_candidate_vals;
    for (int row = 0; row < _img_width / _window_row; row++) {
        for (int col = 0; col < int(_max_r) / _window_col; col++) {
            int start_row = row * _window_row;
            int start_col = col * _window_col;
            int end_row = start_row + _window_row;
            int end_col = start_col + _window_col;
            std::map<float, WindowIndex> map_candidate_vals; // 10 lowest window z.
            for (int i = start_row; i < end_row; i++) {
                for (int j = start_col; j < end_col; j++) {
                    int id = _elevationMap[i][j].cloud_id;
                    if (id >= 0) { // window cloud_id >= 0
                        float z_id = (_cloud->points[id].z > -_gr_height) ? _cloud->points[id].z
: _gr_height;

                        WindowIndex reg;
                        reg.range = i;
                        reg.angle = j; // intensity, vert_angle and min_z
                        reg.label = _elevationMap[i][j].label;
                        if (reg.label == 0)
                            map_candidate_vals[z_id] = reg;
                        region_vals.push_back(reg);
                    }
                }
            }
            if (temp_candidate_vals.size() > 0) {
                for (auto &[min_z, info] : temp_candidate_vals)
                    map_candidate_vals[min_z] = info;
                temp_candidate_vals.clear();
            }
            if (map_candidate_vals.size() >= _window_min_pts) {
                size_t ls = 0;
                for (auto &[min_z, info] : map_candidate_vals) {
                    if (ls < _window_min_pts) // for all min_z -> info.vert_angle should be < 1.8
                        candidate_vals.push_back(info);
                    ls++;
                }
            } else {
                temp_candidate_vals = map_candidate_vals;
            }
            if (candidate_vals.size() >= (_window_min_pts/2)) {
                calSurfaceNormals(candidate_vals, normals, mean, min_svd, dist);
                candidate_vals.clear();
                setWindowsParams(region_vals, normals, mean, min_svd, dist);
                region_vals.clear();
            }
            if (reg_idx == _window - 1) { // if its the last region
                if (temp_candidate_vals.size() > 0) {
                    for (auto &[min_z, info] : temp_candidate_vals)
                        map_candidate_vals[min_z] = info;
                    temp_candidate_vals.clear();
                }
                size_t ls = 0;
                for (auto &[min_z, info] : map_candidate_vals) {
                    if (ls < _window_min_pts)
                        candidate_vals.push_back(info);
                    ls++;
                }
                if (candidate_vals.size() > 5) {
                    calSurfaceNormals(candidate_vals, normals, mean, min_svd, dist);
                    candidate_vals.clear();
                }
                setWindowsParams(region_vals, normals, mean, min_svd, dist);
                region_vals.clear();
            }
            reg_idx++;
        }
    }
}
```


Code 3 Project 3D point cloud to BEV image

```
def pointCloudBEVFeature(PointCloud_, Discretization, bc):
    Height = BEV_HEIGHT + 1
    Width = BEV_WIDTH + 1

    # Discretize Feature Map
    PointCloud = np.copy(PointCloud_)
    # PointCloud[:, 0] = np.int_(np.floor(PointCloud[:, 0] / Discretization))
    # To show ego vehicle at center for full
    PointCloud[:, 0] = np.int_(
        np.floor((PointCloud[:, 0] - bc['minX']) / Discretization))
    PointCloud[:, 1] = np.int_(
        np.floor(PointCloud[:, 1] / Discretization) + Width / 2)

    # sort-3times. sort by z, then y then x.
    indices = np.lexsort(
        (-PointCloud[:, 2], PointCloud[:, 1], PointCloud[:, 0]))
    PointCloud = PointCloud[indices]

    # Height Map
    heightMap = np.zeros((Height, Width))

    # selects the unique x, y points i.e (0:2) and saves the index in an array
    _, indices = np.unique(PointCloud[:, 0:2], axis=0, return_index=True)
    PointCloud_frac = PointCloud[indices]
    # some important problem is image coordinate is (y,x), not (x,y)
    max_height = float(np.abs(bc['maxZ'] - bc['minZ']))
    heightMap[np.int_(PointCloud_frac[:, 0]),
              np.int_(PointCloud_frac[:, 1])] = PointCloud_frac[:, 2] / max_height

    # Intensity Map & DensityMap
    intensityMap = np.zeros((Height, Width))
    densityMap = np.zeros((Height, Width))

    _, indices, counts = np.unique(
        PointCloud[:, 0:2], axis=0, return_index=True, return_counts=True)
    PointCloud_top = PointCloud[indices]

    normalizedCounts = np.minimum(1.0, np.log(counts + 1) / np.log(64))

    intensityMap[np.int_(PointCloud_top[:, 0]), np.int_(
        PointCloud_top[:, 1])] = PointCloud_top[:, 3]
    densityMap[np.int_(PointCloud_top[:, 0]), np.int_(
        PointCloud_top[:, 1])] = normalizedCounts

    RGB_Map = np.zeros((3, Height - 1, Width - 1))
    RGB_Map[2, :, :] = densityMap[:, BEV_HEIGHT, :BEV_WIDTH] # r_map
    RGB_Map[1, :, :] = heightMap[:, BEV_HEIGHT, :BEV_WIDTH] # g_map
    RGB_Map[0, :, :] = intensityMap[:, BEV_HEIGHT, :BEV_WIDTH] # b_map

    # Scale the RGB_Map values to the range 0-255
    RGB_Map = (RGB_Map * 255).astype(np.uint8)

    return RGB_Map
```

Code 4 Image augmentation

```
# Define Albumentations augmentation pipeline
transform = A.Compose([
    A.Rotate(limit=360, p=1.0),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.ShiftScaleRotate(shift_limit=0.2, scale_limit=0.0, rotate_limit=360, p=0.5)
])
```

Code 5 Resampler

```
class Resampler(nn.Module):
    def __init__(self, resolution, dimensions):
        super().__init__()
        self.near = dimensions[1]
        self.far = dimensions[3]
        self.grid = _make_grid(resolution, dimensions)

    def forward(self, features, calib):
        self.grid = self.grid.to(features.device)

        # Assume world Y-coordinate is zero
        calib = calib[:, [0, 2]][..., [0, 2]].view(-1, 1, 1, 2, 2)
        cam_coords = torch.matmul(calib, self.grid.unsqueeze(-1)).squeeze(-1)

        ucoords = cam_coords[..., 0] / cam_coords[..., 1]
        ucoords = ucoords / features.size(-1) * 2 - 1
        zcoords = (cam_coords[..., 1] - self.near) / (self.far - self.near) * 2 - 1

        grid_coords = torch.stack([ucoords, zcoords], -1).clamp(-1.1, 1.1)
        return F.grid_sample(features, grid_coords)
```

Code 6 View transform module

```
class ViewTransform(nn.Module):
    def __init__(self, in_channels, channels, resolution, dimensions,
                 ymin, ymax, focal_length):
        super().__init__()

        self.conv = nn.Conv2d(in_channels, channels, 1)
        self.bn = nn.GroupNorm(16, channels)
        self.resampler = Resampler(resolution, dimensions)

        self.zmin, zmax = dimensions[1], dimensions[3]
        self.in_height = math.ceil(focal_length * (ymax - ymin) / self.zmin)
        self.ymin = (ymin + ymax) / 2
        self.out_depth = math.ceil((zmax - self.zmin) / resolution)

        self.fc = nn.Conv1d(
            channels * self.in_height, channels * self.out_depth, 1, 1
        )
        self.out_channels = channels

    def forward(self, features, calib, *args):
        features = torch.stack([self._crop_feature_map(fmap, cal)
                                for fmap, cal in zip(features, calib)])

        features = F.relu(self.bn(self.conv(features)))

        B, C, _, W = features.shape
        flat_feats = features.flatten(1, 2)
        bev_feats = self.fc(flat_feats).view(B, C, -1, W)

        return self.resampler(bev_feats, calib)

    def _crop_feature_map(self, fmap, calib):
        focal_length, principal_point = calib[1, 1:]
        vmid = self.ymin * focal_length / self.zmin + principal_point
        vmin = math.floor(vmid - self.in_height / 2)
        vmax = math.floor(vmid + self.in_height / 2)
        return F.pad(fmap, [0, 0, -vmin, vmax - fmap.shape[-2]])
```

Code 7 Camera-LiDAR feature fusion Snippet

```
class FuseFeatures(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(DoubleConvolution, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

    def forward(self, bev_feat, img_feat):
        bev_feat = self.conv(bev_feat) # Apply the first convolution to bev_feat
        combined_feat = torch.cat((bev_feat, img_feat), dim=1) # Concatenate with img_feat
        fused_feat = self.conv(combined_feat) # Apply the second convolution

        return fused_feat
```