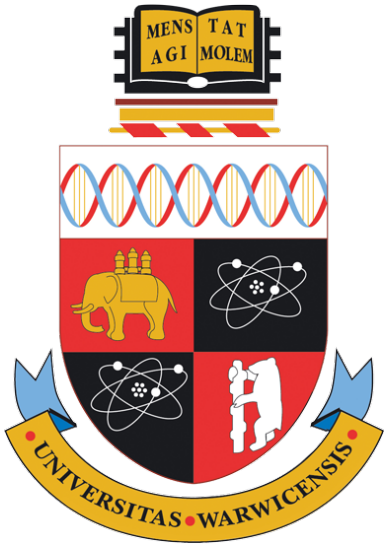
WARWICK
THE UNIVERSITY OF WARWICK



Cooperative Perception for Driving Applications

by

Eduardo Henrique Arnold

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

WMG

December 2021

WARWICK
THE UNIVERSITY OF WARWICK

Contents

List of Tables

List of Figures

Acknowledgments

My PhD journey has been a life changing experience that would not have been possible without the help and support from many people. First, I would like to thank my supervisors Mehrdad Dianati and Paul Jennings for their patience, guidance, support, and insightful discussions. I am thankful to Mehrdad for giving me the opportunity of starting this fully-funded PhD programme at this great university, for his extensive help in revising manuscripts and encouraging me to grow professionally by letting me explore teaching and other research opportunities in external projects. I am also grateful to Paul for asking me the big picture questions which helped to guide the direction of this PhD and for his constant assistance through the writing of this thesis. I would like to thank the colleagues I had the privilege to work with, particularly Sajjad Mozaffari for his support, suggestions and interesting discussions, Omar Y. Al-Jarrah who guided me during my early research stages and Graham Lee for his help and advice regarding data collection.

Moreover, I thank all colleagues in the Intelligent Vehicles group who made my PhD journey a fun and sociable experience with the exciting board-game sessions at the Dirty Duck, gym work-outs and our regular afternoon tea breaks. I also thank my friend Luis Oliveira for his support, guidance, and house plants. I am grateful to all my friends, including the ones I have met through Warwick Salsa, Warwick Tango and Warwick Pride societies and all flatmates for their friendship and hospitality.

Lastly, I am eternally grateful to my family, Sandra, Valmir and Heloisa Arnold, for their love and patience throughout the PhD but particularly during the difficult times. Also during this PhD journey, I was lucky enough to have met my partner in life, Andrew Anzel. I am thankful to Andrew for his constant love, support, and encouragement which has made my life and PhD experience much brighter and happier.

Declarations

1 Publications

Parts of this thesis have been previously published by the author in the following:

[1] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019. doi: 10.1109/TITS.2019.2892405

[2] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. Cooperative Object Classification for Driving Applications. In *IEEE Intelligent Vehicles Symposium*, pages 2484–2489, 2019. doi: 10.1109/IVS.2019.8813811

[3] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative Perception for 3D Object Detection in Driving Scenarios Using Infrastructure Sensors. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2020. doi: 10.1109/TITS.2020.3028424

[4] Eduardo Arnold, Sajjad Mozaffari, Mehrdad Dianati, and Paul Jennings. Visual Sensor Pose Optimisation Using Rendering-based Visibility Models for Robust Cooperative Perception. *Under review, IEEE Transactions on Systems, Man and Cybernetics: Systems*, 2021

[5] Eduardo Arnold, Sajjad Mozaffari, and Mehrdad Dianati. Fast and Robust Registration of Partially Overlapping Point Clouds. *IEEE Robotics and Automation Letters*, 2021. doi: 10.1109/LRA.2021.3137888

Research was performed in collaboration during the development of this thesis, but does not form part of the thesis:

[6] Francesco Bellotti, Riccardo Berta, Ahmad Kobeissi, Nisrine Osman, Eduardo Arnold, Mehrdad Dianati, Ben Nagy, and Alessandro De Gloria. Designing an IoT Framework for Automated Driving Impact Analysis. In *IEEE Intelligent Vehicles Symposium*, pages 1111–1117, 2019. doi: 10.1109/IVS.2019.8813989

[7] Francesco Bellotti, Nisrine Osman, Eduardo H. Arnold, Sajjad Mozaffari, Satu Innamaa, Tyron Louw, Guilhermina Torrao, Hendrik Weber, Johannes Hiller, Alessandro De Gloria, Mehrdad Dianati, and Riccardo Berta. Managing Big Data for Addressing Research Questions in a Collaborative Project on Automated Driving Impact Assessment. *Sensors*, 20(23), 2020. ISSN 1424-8220. doi: 10.3390/s20236773. URL `https://www.mdpi.com/1424-8220/20/23/6773`

[8] Johannes Hiller, Sami Koskinen, Riccardo Berta, Nisrine Osman, Ben Nagy, Francesco Bellotti, Ashfaqur Rahman, Erik Svanberg, Hendrik Weber, Eduardo H. Arnold, Mehrdad Dianati, and Alessandro De Gloria. The L3Pilot Data Management Toolchain for a Level 3 Vehicle Automation Pilot. *Electronics*, 9(5), 2020. ISSN 2079-9292. doi: 10.3390/electronics9050809. URL `https://www.mdpi.com/2079-9292/9/5/809`

[9] Eduardo Arnold, Jiaxin Chen, Ivan Croydon-Veleslavov, Anurag Deshpande, Tanaya Guha, Yixuan He, Ben Moseley, Gim Seng Ng, Luke Prince, Thomas Statham, and Peter Strong. Alan Turing Data Study Group Final Report: Greenvest Solutions, February 2021. URL `https://doi.org/10.5281/zenodo.4534349`

[10] Sajjad Mozaffari, Eduardo Arnold, Mehrdad Dianati, and Saber Fallah. A Comparative Study of Ego-centric and Cooperative Perception for Lane Change Prediction in Highway Driving Scenarios. In *Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems - ROBOVIS,*, pages 113–121. INSTICC, SciTePress, 2021. ISBN 978-989-758-537-1. doi: 10.5220/0010655700003061

[11] Sajjad Mozaffari, Eduardo Arnold, Mehrdad Dianati, and Saber Fallah. Early Lane Change Prediction for Automated Driving Systems Using Multi-Task Attention-based Convolutional Neural Networks. *IEEE Transactions on Intelligent Vehicles*, 2021. doi: 10.1109/TIV.2022.3161785

2 Sponsorships and Grants

This work was supported by Jaguar Land Rover and the U.K.-EPSRC as part of the jointly funded Towards Autonomy: Smart and Connected Control (TASCC) Programme under Grant EP/N01300X/1.

Abstract

An automated vehicle needs to understand its driving environment to operate safely and reliably. This function is performed within the vehicle's perception system, where data from on-board sensors is processed by multiple perception algorithms, including 3D object detection, semantic segmentation and object tracking. To take advantage of different sensor modalities, multiple perception methods fusing the data from on-board cameras and lidars have been devised. However, sensing exclusively from a single vehicle is inherently prone to occlusions and a limited field-of-view that indiscriminately affects all sensor modalities. Alternatively, cooperative perception incorporates sensor observations from multiple view points distributed throughout the driving environment.

This research investigates if and how cooperative perception is capable of improving the detection of objects in driving environments using data from multiple, spatially diverse sensors. Over the course of this thesis, four studies are conducted considering different aspects of cooperative perception.

The first study considers the various impacts of occlusions and sensor noise on the classification of objects in images and investigates how to fuse data from multiple images. This study serves as a proof-of-concept to validate the core idea of cooperative perception and presents quantitative results on how well cooperative perception can mitigate such impairments.

The second study generalises the problem to 3D object detection using infrastructure sensors capable of providing depth information and investigates different sensor fusion approaches for such sensors. Three sensor fusion approaches are devised and evaluated in terms of object detection performance, communication bandwidth and inference time. This study also investigates the impact of the number of sensors in the performance of object detection. The results show that the proposed cooperative 3D object detection method achieves more than thrice the number of correct detections compared to single sensor baselines, while also reducing the number of false positive detections.

Next, the problem of optimising the pose of fixed infrastructure sensors in cluttered driving environments is considered. Two novel sensor pose optimisation methods are proposed, one using gradient-based optimisation and one using integer programming techniques, to maximise the visibility of objects. Both use a novel visibility model, based on a rendering engine, capable of determining occlusions between objects. The results suggest that both methods have the potential to guide the cost effective deployment of sensor networks in cooperative perception applications.

Finally, the last study considers the problem of estimating the relative pose between non-static sensors relying on sensor data alone. To that end, a novel and computationally efficient point cloud registration method is proposed using a bespoke feature encoder and attention network. Extensive results show that the proposed method is capable of operating in real-time and is more robust for point clouds with low field-of-view overlap compared to existing methods.

Acronyms

3DOD 3D Object Detection.

ABS Anti-Lock Brakes.

ADAS Advanced Driver-Assistance Systems.

AP Average Precision.

AV Automated Vehicle.

AWGN Additive White Gaussian Noise.

BB Bounding Box.

BEV Bird's Eye View.

CAD Computer Aided Design.

CDF Cumulative Distribution Function.

CNN Convolutional Neural Network.

ECDF Empirical Cumulative Distribution Function.

FC Fully Connected (Layer/Network).

FCN Fully Convolutional Network.

FP Feature Propagation (layer).

fps frames per second.

FPS Farthest Point Sampling.

GNN Graph Neural Network.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GPU Graphics Processing Unit.

INS Inertial Navigation System.

IOU Intersection over Union.

IP Integer Programming.

LED Light Emitting Diode.

LoS Line of Sight.

MLP Multilayer Perceptron.

MRV Markovian Random Field.

NMS Non-Maxima Suppression.

NN Nearest Neighbour.

PCL Point cloud.

RANSAC Random Sample Consensus.

ReLU Rectified Linear Unit.

ROI Region of Interest.

RPN Region Proposal Network.

RSU Road Side Unit.

SA Set Abstraction (layer).

SGD Stochastic Gradient Descent.

SLAM Simultaneous Localisation and Mapping.

SVD Singular Value Decomposition.

TOF Time-of-Flight.

VFE Voxel Feature Encoding.

Chapter 1

Introduction

Vehicles have enjoyed significant technical developments since their widespread adoption in the twentieth century. Features such as power steering, cruise control, seat belts, Anti-Lock Brakes (ABS) and airbags improved the safety and quality of the driving experience. More recently, Advanced Driver-Assistance Systems (ADAS) have extended vehicles' capabilities by providing new safety features such as traffic alerts, automatic breaking, and lane departure warnings. Furthermore, ADAS are able to automate some driving tasks under specific conditions by providing features such as automatic parking, lane keeping, and adaptive cruise control. ADAS rapid development has promoted a race towards full driving autonomy where the ultimate goal is creating an Automated Vehicle (AV) capable of driving under all conditions without human supervision, which the Society of Automotive Engineers (SAE) classifies as level 5 driving automation [12].

Both ADAS and AVs require an understanding of the driving environment in order to operate safely and reliably. For example, the vehicle must be able to identify other road users, *e.g.* vehicles, cyclists and pedestrians, as well as lane limits and road obstacles. This function is performed within the vehicle's perception system, where data from one or multiple on-board sensors must be processed and converted into a representation of the driving environment. Such representation can have various forms and often use the output of a number of perception algorithms such as 3D object detection and semantic/instance segmentation, as illustrated in Figures 1.1 and 1.2, respectively. For example, objects in the driving environment can be represented by a set of 3D bounding boxes, delimiting the position, size and orientation of objects in the environment, as well as the objects' class. The resulting list of detected objects can then used as input to advanced algorithms in ADAS/AVs to provide safety features and high-level functions such as trajectory planning and control.

The accuracy of the perception algorithms, *e.g.* the accuracy of the detected bounding boxes, depends on a number of factors, including weather and lighting

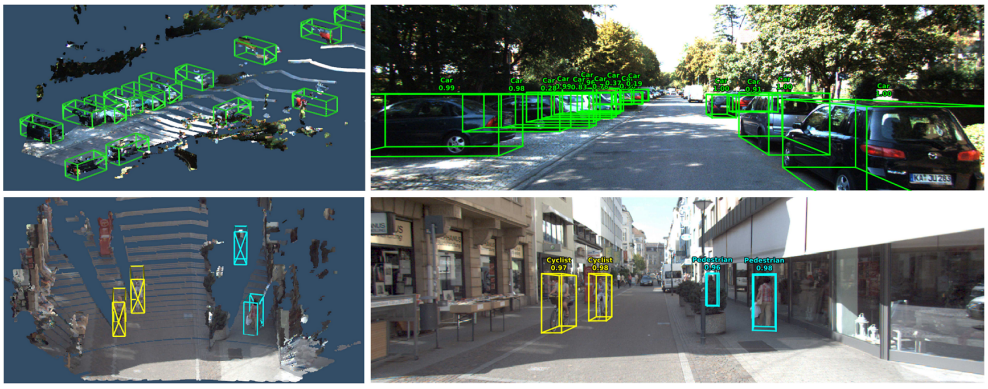


Figure 1.1: 3D Object Detection: lidar and camera sensors are used to detect objects in 3D space. The detections are represented as green 3D boxes in the 3D point cloud (left) or projected back to the image (right). Image obtained from [13].

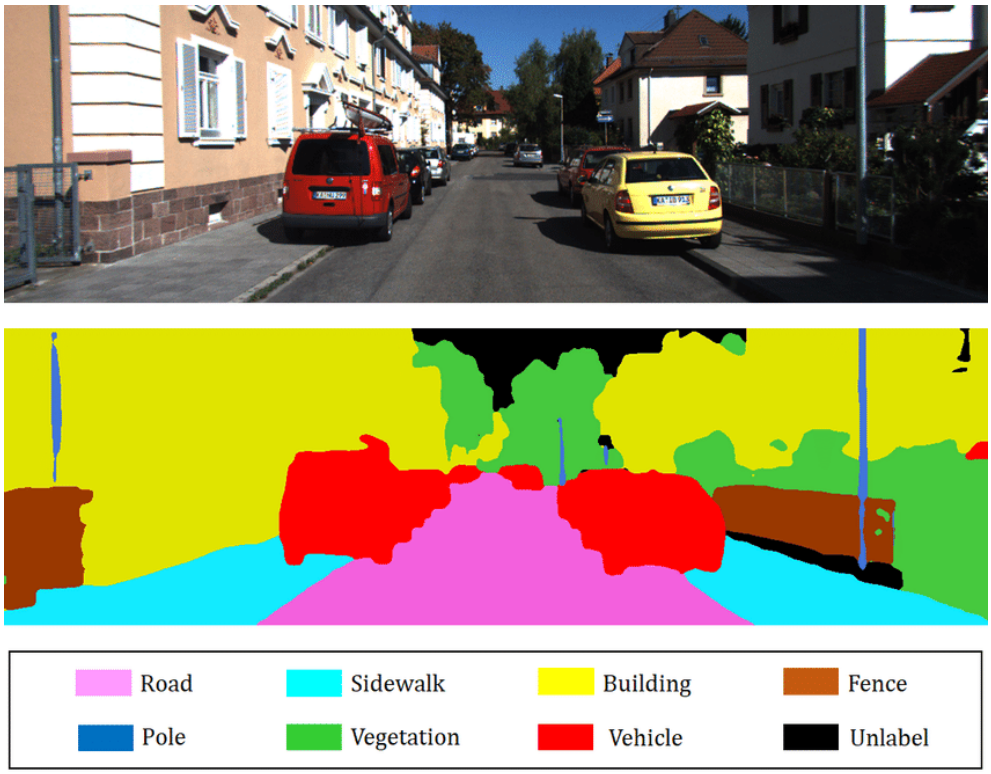


Figure 1.2: Semantic Segmentation: each image pixel is labelled with a corresponding class. This semantic output can be used to determine drivable areas. Image obtained from [14].

conditions, the choice of on-board sensors and the performance of perception algorithms. Camera sensors provide rich texture information that can be used to classify objects, however, they lack depth information to accurately localise objects within the 3D space and are unreliable under poor illumination. On the other hand, lidars provide 3D points with accurate depth and do not rely on external illumination, but the density of points decreases substantially with the distance from the sensor. To take advantage of different sensor modalities, multiple perception methods have been devised to fuse the data from on-board cameras and lidars.

1.1 Motivation

Despite the improvements in perception due to multi-modal sensor fusion, sensing exclusively from a single vehicle is inherently prone to a major category of sensing impairments that indiscriminately affects all sensor modalities. Such impairments include occlusions, restricted perception horizon due to limited field-of-view and low point/pixel density at distant regions. As a result, these impairments may prevent the detection of road users and lead to potentially hazardous situations, particularly in complex road segments. One could hypothesise whether such impairments can be mitigated by incorporating sensor observations from multiple view points distributed along the driving environment. To this end, cooperative perception could enable agents to share sensor data and improve the accuracy of 3D object detection, where the effective field-of-view is enlarged and potentially occluded objects/hazards may be correctly identified. Cooperative perception could increase the perception horizon, which would allow vehicles to become aware of objects that are not within direct line-of-sight, and thus, could not be identified solely by on-board sensors. Consequently, the safety of road users can be increased compared to human driving, where the environmental awareness is limited to the drivers' line-of-sight. Furthermore, knowing the state of the driving environment beyond the vehicle's line-of-sight would allow for long-term planning and lead to more efficient and comfortable driving capabilities.

Ultimately, the fundamental concept of cooperative perception can be employed in a wide range of applications where multiple agents are able to share sensor information to improve their collective understanding of the environment. In the driving context, this concept can be realised considering sensors on-board of vehicles, or fixed infrastructure-based sensors which are strategically installed in challenging driving environments. Leveraging the data from high-end infrastructure sensors would allow vehicles to use lower-end on-board sensors, and thus, reduce the cost of the suite of sensors required in automated vehicles.

Despite the aforementioned benefits, fusing the data from multiple, spatially diverse sensors is still an open problem. A significant part of the problem is defining the fusion algorithm, which dictates how and at which stage of the perception pipeline the data is fused. Furthermore, the data from multiple sensors must be aligned both spatially and temporally before it can be fused. The spatial alignment is achieved by transforming the data from each sensor to a common coordinate system. This requires obtaining the relative pose transformation between the sensors and a common coordinate system, which is another critical part of the problem. The relative pose between fixed infrastructure sensors can be obtained through calibration techniques, however, sensors on-board of vehicles are constantly moving and, thus, cannot be calibrated in the same way. Although global positioning and inertial systems could be used to obtain the relative pose between vehicles, such methods are prone to significant pose errors that could degrade the performance of cooperative perception. The temporal alignment consists of ensuring that data from multiple sources are synchronised in time, which can be challenging due to communication delay and packet drops. When considering the deployment of infrastructure sensors, another problem resides in determining the number, position, and orientation of such sensors. The empirical placement of these sensors to maximise the coverage of road segments is not capable of guaranteeing that all objects of interest will be detected since objects are prone to occlusions.

1.2 Research Objective

The aim of this research is to enhance the detection performance of 3D object detection used in the context of automated vehicles and vehicles with ADAS capabilities. This leads to the main objective of this research:

> Investigate if and how cooperative perception is capable of improving the detection of objects in driving environments using the data from multiple, spatially diverse sensors.

To achieve this objective, several studies considering different aspects of cooperative perception are reported in this thesis. An overview of these studies, including how they interconnect and address gaps in the literature, is presented in Chapter 3. That chapter also presents the research questions investigated in each study.

1.3 Contributions

The main contributions made in this thesis are summarised as follows:

- A system architecture with three fusion schemes is proposed for Cooperative 3D Object Detection using Infrastructure sensors in Chapter 5; a new dataset considering infrastructure-based sensors is created to evaluate the proposed system; a thorough analysis is performed considering how the number of sensors and fusion algorithms impacts detection performance and communication load, providing a guidance to the practical deployment the system;

- A realistic visibility model capable of inferring occlusions between objects is proposed; two novel optimisation methods, both using the aforementioned visibility model, are proposed to optimise the pose of infrastructure sensors in Chapter 6;

- A novel registration method is proposed to obtain the relative pose between a pair of partially overlapping point clouds in real-time in Chapter 7; a new lidar dataset for cooperative perception is created, including a diverse range of driving scenarios with multiple lidar sensors, which is used to evaluate the registration performance and can also be used for 3D object detection and tracking.

1.4 Outline

This section provides an outline of this thesis.

Chapter 2 reviews existing works in the context of perception for autonomous driving applications. The topics include 3D object detection and classification, cooperative perception, sensor pose optimisation, and point cloud registration. The comprehensive review of these works identifies research gaps and opportunities related to cooperative perception for driving applications. This review identifies that understanding how to fuse data from multiple sensors spatially distributed in an environment, how to optimise the pose of infrastructure sensors and how to efficiently obtain accurate relative pose transformation between sensors on-board of vehicles are all still open challenges.

Chapter 3 provides a concise overview of the research activities in this thesis and how they interconnect. This chapter describes the research methodology, contributions and describes how the research gaps identified in the previous chapter are addressed.

In Chapter 4, an initial concept of cooperative perception for object classification is designed and implemented. The images contain a single object and are artificially impaired with sensor noise and occluding objects. Despite these strong assumptions, this study validates the concept of cooperative perception and shows its potential in overcoming sensor noise and occlusions.

Chapter 5 generalises the previous concept of cooperative perception to 3D object detection using lidar sensors. A cooperative perception system is designed and implemented assuming fixed infrastructure sensors. Three sensor fusion algorithms are proposed and evaluated in terms of communication load, inference time and object detection performance. The system is capable of significantly increasing the detection performance compared to non-cooperative baselines and has shown resilience to sensor impairments.

In the previous chapter, the infrastructure sensors are empirically placed in the environment according to heuristics including ground coverage and field-of-view overlap maximisation. In Chapter 6, the problem of optimising the pose of fixed infrastructure sensors for cooperative perception is addressed. Both have the potential to guide the cost effective deployment of sensor networks in cooperative perception applications.

Chapter 7 addresses the problem of estimating the relative pose between sensors on-board of vehicles. A novel point cloud registration method is proposed to efficiently recover the relative pose between two lidar sensors. The evaluation on real and synthetic datasets demonstrates that the proposed method is robust to partially overlapping point clouds and can be used in real-time cooperative perception applications.

In Chapter 8, the results from the previous chapters are discussed in the light of the research objectives. This chapter concludes this work by highlighting the key learnings, strengths and the limitations of this research. In addition, recommendations for future research directions are provided.

Chapter 2

Literature Review

This chapter reviews existing works related to perception problems in driving applications organised in six sections. First, Section 2.1 discusses existing sensor technology and their usage in the context of perception problems. Section 2.2 reviews methods for object classification. Next, methods for 3D object detection and cooperative 3D object detection are reviewed in Section 2.3 and 2.4, respectively. Then, sensor pose optimisation methods and point cloud registration methods are reviewed in Section 2.5 and 2.6, respectively. Finally, a summary of the identified research gaps and opportunities are presented in Section 2.7.

2.1 Sensors

Although humans primarily use their visual and auditory systems while driving, artificial perception methods rely on multiple sensor modalities to overcome shortcomings of individual sensors. There are a wide range of sensors used by autonomous vehicles: passive ones, such as monocular and stereo cameras, and active ones, including lidar, radar and sonar. Although the usage of radar sensors for perception applications have been recently proposed [15, 16], the majority of research on perception for AVs focus on cameras and lidars sensors, and thus these two categories are reviewed below. The advantages and disadvantages of each sensor modality is summarised in Table 2.1.

Monocular cameras provide information in the form of pixel intensities, which at a bigger scale reveal shape and texture details. The shape and texture information can be used to detect lane geometry, traffic signs [17] and objects [18]. One disadvantage of monocular cameras is the lack of depth information, which is required for accurate object size and position estimation. Alternatively, depth cameras can be used to recover the depth of each point relative to the camera. In this category, a stereo camera setup, *i.e.* a pair of monocular cameras separated by a small baseline, uses matching algorithms to

Table 2.1: Sensors Comparison

	Advantages	**Disadvantages**
Monocular Camera	Readily available and inexpensive. Multiple specifications available.	Prone to adverse light and weather conditions. No depth information provided.
Stereo Camera	Higher point density when compared to lidar. Provides dense depth map.	Depth estimation is computationally expensive. Poor performance with textureless regions or during night-time. Limited Field-of-View (FoV). Depth error increases quadratically with distance.
Lidar	360 degrees coverage, precise distance measurements. Not affected by light conditions.	Raw point cloud does not provide texture information. Expensive and large equipment.
Solid-State lidar	No moving mechanical parts, compact size. Large scale production should reduce final cost.	Limited FoV when compared to mechanical scanning lidar. Still under development.

find correspondences in both images and computes the depth of each point based on the disparity between correspondences [19]. Time-of-Flight (ToF) cameras are another modality of depth cameras where depth is inferred by measuring the delay between emitting and receiving modulated infrared pulses [20]. This active sensor technology has been applied for vehicle safety applications [21], but despite the lower integration price and computational complexity it has lower resolution when compared to stereo cameras. A known limitation of camera sensors is their susceptibility to light and weather conditions, challenging cases ranging from low luminosity at night-time to abrupt brightness changes after entering or exiting tunnels. Another issue is the flickering behaviour created by the recent usage of LEDs on traffic signs and vehicles brake lights caused as the camera fails to consistently capture the emitted light due to the LEDs' switching behaviour. Additionally, weather conditions such as rain and snow can degrade the image quality and deteriorate the performance of the algorithms using the image data.

Lidar sensors emit laser beams and measure the time between emitting and detecting the pulse back. The timing information determines the distance of obstacles in any given direction. The sensor readings result in a set of 3D points,

also called a point cloud, and corresponding reflectance values representing the strength of the received pulses. One of the advantage of lidars when compared to stereo-based depth cameras is that the error in the measured depth of each point does not depend on the distance from the point to the sensor, while for stereo cameras this error increases quadratically [22]. As active sensors, external illumination is not required and thus more reliable detection can be achieved considering extreme lighting conditions (*e.g.,* night-time or sun glare scenarios). Still, these sensors are also vulnerable to adverse weather conditions, particularly heavy fog and snow [23]. Standard lidar models, such as the HDL-64L [24], use an array of rotating laser beams to obtain 3D point clouds in 360 degrees and up to 120m radius. This sensor can output 120 thousand points per frame, which amounts to 1,200 million points per second on a 10 Hz frame rate. Velodyne recently announced the VLS-128 model [25] featuring 128 laser beams, higher angular resolution and 300m radius range. The announcement suggests that the increased point density might enhance the recall of methods using this modality but challenges real time processing performance. Although the cost of lidar sensors is still the main cause preventing its widespread adoption, new technology including solid state lidar technology [26] and large scale production are expected to reduce the costs of individual sensors.

2.2 Object Classification

Object classification is a sub-problem within object detection. The problem definition of object detection is "to determine where objects are located in a given image (object localisation) and which category each object belongs to (object classification)" [27]. In other words, while object classification is the problem of classifying an object depicted in an image into one of a set of classes, object detection is the problem of identifying if and where an object is present in an image. In general, the classification of an object in an image is performed within a detection pipeline consisting of two steps. First, obtaining a set of object proposals in the form of a 2D bounding boxes determining the object size on the image plane. Next, classifying each proposal into a set of predefined classes, or, alternatively discarding the proposal as not containing any objects. The set of predefined classes depend on the application, but in the driving domain these typically include vehicles, pedestrians and cyclists. Early works use the selective search [28] algorithm to create class agnostic 2D bounding boxes over potential objects on images. The image patches determined by these boxes can then be classified into the predefined classes using a convolutional neural network-based classifier [29]. More recent methods perform detection and classification simultaneously by considering class-specific object bounding

box proposals [30].

A variation of this problem considers the classification of an object's 3D model encoded by meshes or 3D points (point cloud). In this context, object classification methods can be categorized into two groups. The first one uses 3D shape features derived directly from the 3D model. In contrast, the second group renders the 3D shape into one or multiple images from different viewpoints and then extract features from these images to perform classification.

In the first group, the features can be "hand-engineered" [31–33] or learned directly from the data using voxel [34, 35] or point cloud [36, 37] representations. For example, *3D ShapeNet* [34] uses a convolutional deep belief network to learn the joint distribution of volumetric representation (voxels) of 3D objects and their class labels. In contrast, *PointNet* [36] directly uses point cloud data, *i.e.* 3D points on the surface of the object, to perform object classification and segmentation. Despite improvements in this group, the dimensionality and low resolution of voxelised 3D inputs undermines the classification performance of these methods [38].

In the group considering rendered images of the 3D object, the feature extraction can be either based on "hand-engineered" features, *e.g.* using Scale Invariant Feature Transform (SIFT) [39] and Fisher vectors [40], or learned using Convolutional Neural Network (CNN) models [29, 41]. In the learning category, Su *et al.* [42] render each 3D object model in 12 different views and then propose a CNN architecture to generate a global feature of the object based on the rendered views. This descriptor is used both for object classification and retrieval. Further work by Kanezaki *et al.* [43] leverages the similarities between pose estimation and object classification problems to create a single network that can perform both tasks simultaneously. Their proposed model takes multi-view inputs and predicts both pose and class for each image, selecting the object class that maximizes the overall class likelihood. Furthermore, the object pose is treated as a latent variable, allowing unsupervised training on the pose with an un-aligned dataset.

Considering occlusions, Meger *et al.* [44] train image classifiers on full objects and sub-parts to detect occluded 3D objects in an indoor scenario. They formulate the presence of an object under a Bayesian framework considering size priors, depth and structure-from-motion posteriors. Yilmaz *et al.* [45] explore recurrent connections on convolutional networks to overcome occlusion on image classification tasks. Despite classification performance improvement, a naive occlusion model is used, where black rectangles are drawn upon original images. Chandler *et al.* [46] generate an occluded dataset using a range occlusion models, but limited in the number of object classes and samples. Their method uses an in-painting technique to overcome occlusion, but in turn requires segmentation and annotation of the occlusion degree to be effective.

2.3 3D Object Detection

The problem of 3D object detection consists of detecting objects by estimating their 3D position, size, heading angle and class. This problem is a generalisation of object classification, since the object position, size and orientation must also be provided. In the driving domain, the class of objects is generally limited to vehicles, pedestrians and cyclists [47]. 3D object detection models receive input data from sensors, *e.g.* cameras and/or lidars, and output 3D bounding boxes, which fully describe the 3D position, size and orientation of objects. A traditional pipeline consists of segmentation (*e.g.,* graph-based segmentation [48] and voxel-based clustering methods [49]), hand-engineered feature extraction (*e.g.,* voxel's probabilistic features [49]) and classification stages (*e.g.,* a mixture of bag-of-words classifiers [50]). Unlike traditional pipelines, which optimise each stage individually, end-to-end pipelines optimise the overall pipeline performance. An end-to-end detection method leverages learning algorithms to propose regions of interest and extract features from the data. The shift towards representation learning and end-to-end detection was possible by using deep learning methods, such as deep convolutional networks, which showed a significant performance gain in different applications [30, 51].

The review of 3D object detection methods is divided into three categories according to the input data modality : monocular image, point cloud and fusion based methods. Each category is reviewed in a sub-section below, and an overview of methodology, advantages and limitations for these methods is provided in Table 2.2.

2.3.1 Monocular Image-based Methods

The problem of 2D object detection has been widely explored in the literature and has achieved remarkable results in several datasets [29, 52]. In the context of autonomous driving, datasets such as KITTI [47], Cityscapes [53], nuScenes [54] and Waymo Open [55] offers particular settings that pose challenges to object detection. These settings, common to most driving environments, include scenes with a high density of objects, many occluded or truncated, and highly saturated areas or shadows. Furthermore, 2D detection on the image plane is not enough for reliable driving systems: accurate 3D space localisation and size estimation is required for driving applications where vehicles require a 3D understanding of the environment. This section focuses on methods that are able to estimate 3D bounding boxes based only on monocular images. Since no depth information is available, most approaches first detect 2D candidates before predicting a 3D bounding box that contains the object using neural networks [56], geometrical constraints [57] or 3D model matching [58, 59]. More recent approaches use deep neural networks to estimate a depth map

Table 2.2: Comparison of 3D Object Detection Methods by Category

Category		**Methodology/Advantages**	**Limitations/Drawbacks**
Monocular		Uses a single RGB image. Lift 2D detection on the image plane to 3D through re-projection constraints. Newer methods use depth-map regression to obtain pseudo-point clouds.	The lack of explicit depth information on the input format limits the accuracy of object localization.
Point-cloud	Projection	Projects point clouds into a 2D image and use established architectures for object detection on 2D images with extensions to regress 3D bounding boxes.	Projecting the point cloud data inevitably causes information loss. It also prevents the explicit encoding of spatial information as in raw point cloud data.
	Volumetric	Generates a 3 dimensional representation of the point cloud in a voxel structure and uses Fully Convolutional Networks (FCNs) to predict object detections. Shape information is encoded explicitly.	Expensive 3D convolutions increase model's inference time. The volumetric representation is sparse and computationally inefficient.
	PointNets	Uses feed-forward networks consuming raw 3D point clouds to generate predictions on class and estimated bounding boxes.	Considering the whole point cloud as input can increase run-time. Difficult establishing region proposals considering raw point inputs.
Fusion		Fuses both front view images and point clouds to generate a robust detections. Architectures usually consider multiple branches, one per modality, and rely on region proposals. Allows modalities to interact and complement each other.	Requires calibration between sensors, and depending on the architecture can be computationally expensive.

from a single monocular image [60–63]. The predicted depth map is then back-projected to 3D, creating a pseudo-lidar point cloud that can be processed by state-of-the-art lidar-based object detectors.

Chen *et al.* propose Mono3D [56], which leverages a simple region proposal algorithm using context, semantics, hand-engineered shape features and location priors. For any given proposal, these features can be efficiently computed and scored by an energy model. Proposals are generated by exhaustive search on 3D space and filtered with the Non-Maxima Suppression (NMS) algorithm. The proposals are further scored by a Fast R-CNN [29] model that regresses 3D bounding boxes. The work builds upon the authors' previous work 3DOP [64], which considers depth images to generate proposals in a similar fashion. Despite using only monocular images, the Mono3D model slightly improves the performance obtained by [64], which uses depth images. Pham *et al.* [65] extends the 3DOP proposal generation considering class-independent proposals, then re-ranks the proposals using both monocular images and depth maps. Their method outperforms both 3DOP and Mono3D methods, despite using depth images to refine proposals.

An important characteristic of driving environments is severe occlusion present in crowded scenes where objects may occlude other objects or parts of themselves (self-occlusion). Xiang *et al.* introduce visibility patterns into the model to mitigate occlusion effects through object reasoning. They propose the 3D Voxel Pattern (3DVP) [58] representation that models appearance through RGB intensities and 3D shape as a set of voxels with corresponding occlusion masks. This representation allows to recover which parts of the object are visible, occluded or truncated (partially invisible). They obtain a dictionary of 3DVPs by clustering the patterns observed on the data and training a classifier for each specific pattern given a 2D image segment of the vehicle. During the test phase, the pattern obtained through classification is used for occlusion reasoning and 3D pose and localisation estimation. The 3D position is recovered by minimizing the re-projection error between a fixed-sized, oriented 3D bounding box and the 2D detection on the image plane. Their proposed pipeline is still dependent on the performance of Region Proposal Networks (RPNs).

Although some RPNs were able to improve traditional proposal methods [29] they still fail to handle occlusion, truncation and different object scales. Extending the previous 3DVP framework, the same authors propose SubCNN [66], a CNN that explores class information for object detection at the RPN level. They use the concept of subcategory, which are classes of objects sharing similar attributes such as 3D pose or shape. Candidates are extracted using convolutional layers to predict heat maps for each subcategory at the RPN level. After Region of Interest (ROI) proposal estimation, the network outputs

category classification along with refined 2D bounding box estimates. Using 3DVPs [58] as subcategories for pedestrian, cyclist and vehicle classes, the model recovers 3D shape, pose and occlusion patterns. An extrapolating layer is used to improve small object detection by introducing multi-scale image pyramids.

Although the previous 3DVP representations [58, 66] allow to model occlusion and parts appearance, they are obtained as a classification among an existing dictionary of visibility patterns common in the training set. Thus, may fail to generalize to an arbitrary vehicle pose that differs from the existing patterns. To overcome this, Deep MANTA [59] uses a many-task network to estimate vehicle position, part localization and shape based from a monocular image. The vehicle shape consists of a set of key points that characterize the vehicle 3-dimensional boundaries, *i.e.* external vertices of the vehicle. They first obtain 2D bounding regression and parts localization through a two-level refinement region-proposal network. Next, based on the inferred shape 3D model matching is performed to obtain the 3D pose.

Previous methods performed either exhaustive search on the 3D bounding box space [56], estimated 3D localisation using a cluster of appearance patterns [58] and 3D templates [59]. Mousavian *et al.* [57] first extend a standard 2D object detector with 3D orientation (yaw) and bounding box sizes regression. This is justified by the box dimensions having smaller variance and being invariant with respect to the orientation. The 3D box dimensions and orientations are determined by the network prediction. and the 3D object pose is recovered solving for a translation vector that minimizes the re-projection error of the 3D bounding box w.r.t. the 2D detection box on the image plane.

More recently, a new branch of methods use CNNs to regress a depth map from a single monocular image, which is then back-projected to 3D space, creating pseudo-point clouds. These point clouds can then be processed using state-of-the-art methods for lidar point-clouds. In this category, Weng and Kitani [63] obtain 2D detections and instance segmentation from the image and use a CNN on each instance patch to regress a depth map which is then lifted to a point cloud format and finally processed by a point-cloud based detection model. They propose two innovation to alleviate noise in the depth estimation stage, namely a 2D-3D consistency constraint and using the instance segmentation patch to regress depth instead of the whole 2D bounding box. Further work by Rui *et al.* [61] creates a differentiable end-to-end model that creates a depth representation aligned with the final goal of 3D object detection, which results in a significant detection performance increase for monocular methods. You *et al.* [62] improves upon the previous methods by improving upon the depth prediction of far objects, which is the weakness of the previous methods. They propose to use a stereo depth-regression network adapted to

monocular images only and calibrate the depth map estimates using sparse lidar points to further reduce depth errors. The limitation of this branch of methods include the inaccuracies in the depth estimation, which can lead to significant localisation errors [61, 62] and the poor generalisation to unseen objects [67], which poses safety threats.

All the previous monocular methods detects objects using the front-facing camera, ignoring objects on the sides and rear of the vehicle. In contrast, [68] proposes the first 360 degrees panoramic image based method for 3D object detection. They estimate dense depth maps of panoramic images and adapt standard object detection methods for the equirectangular representation. Due to the lack of panoramic labelled datasets for driving, they adapt the KITTI dataset using style and projection transformations. They additionally provide benchmark detection results on a synthetic dataset.

Monocular 3D object detection methods have been widely researched. Although previous works considered hand-engineered features for region proposals [56], most methods have shifted towards a learned paradigm for Region Proposals and second stage of 3D model matching and re-projection to obtain 3D bounding boxes. The main drawbacks of monocular based methods is the lack of depth cues, which limits detection and localization accuracy specially for far and occluded objects, and sensitivity to lighting and weather conditions, limiting the usage of these methods for the day time. Newer pseudo-lidar methods have shown better results than previous approaches, but rely on depth regression methods which have limited accuracy and may introduce large errors for objects far from the camera. The methodology/contributions and limitations of monocular methods are summarised in Table 2.3.

2.3.2 Point Cloud-based Methods

3D object detection methods based on point-clouds can be divided into three subcategories: projection-based, volumetric representations, raw point clouds and multi-representation methods (using multiple input representations). A summary of point cloud-based methods is presented in Table 2.4. Each category is explained and reviewed below, followed by a summary discussion.

Projection Methods

Image classification and object detection in 2D images is a well-researched topic in the computer vision community. The availability of datasets and benchmarked architectures for 2D images make using these methods even more attractive. For this reason, point cloud (PCL) projection methods first transform the 3D points into a 2D image via plane [81], cylindrical [69], spherical [82] or bird-eye view projections that can then be processed using standard

Table 2.3: Summary of Monocular-based 3D Object Detection Methods

Method	**Methodology/Contributions**	**Limitations**
Mono3D [56]	Improves detection performance over 3DOP that relied on the depth channel.	Poor localization accuracy given the lack of depth cues.
3DVP [58]	Novel 3DVP object representation includes appearance, 3D shape and occlusion information. Classification among an existing set of 3DVPs allows occlusion reasoning and recovering 3D pose and localization.	Fixed set of 3DVPs extracted during training limits generalisation to arbitrary object poses.
SubCNN [66]	Uses 3DVP representation to generate occlusion-aware region proposals. The proposals are refined and classified within the object representations (3DVP). Improves RPN model refinement network using CNNs.	Since the 3DVP representation is employed, this method has the same limitations as the previous one.
Deep3DBox [57]	Simplified network architecture by independently regressing bounding box size and angle. Then using image reprojection error minimization to obtain 3D localization.	The reprojection error is dependent on the BB size and angle regressed by the network. This dependence increases localization error.
360Panoramic [68]	Estimates depth for 360 degrees panoramic monocular images. Then adapt a CNN to predict 3D object detections on the recovered panoramic image. The only method capable of using images to detect objects at any angle around the vehicle.	Limited to vehicle detection and fails when the vehicle is too close to the camera. The resolution of the camera limits the range of detection.
Monocular Pseudo-lidar [61–63]	Estimates depth for monocular images, lifting the depth maps into 3D point clouds which are then processed by lidar-based 3D object detectors.	Significant detection performance increase compared to previous methods, however still perform poorly for far-away objects where the depth estimates from a monocular image are poor.

Table 2.4: Summary of Point Cloud-based 3D Object Detection Methods

SubCategory	Method	Methodology/Contributions	Limitations
Projection	VeloFCN [69]	Uses fully convolutional architecture with lidar point cloud bird-eye view projections. Output maps represent 3D bounding box regressions and "objectness" score, the likelihood of having an object at that position.	Detects vehicles only. Limited performance on small or occluded objects due to the loss of resolution across feature maps.
	C-YOLO [70]	Uses a YOLO based single-shot detector extended for 3D BB and orientation regression. The proposed architecture achieves 50 fps runtime, more than any previous method.	There is a tradeoff between inference time and detection accuracy. Single-shot networks underperform networks that use a second stage for refinement.
	TowardsSafe [71]	Uses variational dropout inference to quantify uncertainty in class and bounding box predictions. Aleatoric noise modelling allows the network to generalise better by reducing the impact of noisy samples in the training process.	The uncertaity estimation requires several forward passes of the network. This limits the temporal performance of this method, preventing real-time results.
	BirdNet [72]	Normalizes point cloud representation to allow detection generalisation across different lidar models and specifications.	Input image with only 3 channels encoding height, density and intensity information looses detailed information, which degrades performance.
Volumetric	3DFCN [73]	Extension of the FCN architecture to voxelised lidar points clouds. Single shot detection method.	Requires 3D convolutions, limiting temporal performance to 1 fps.
	Vote3Deep [74]	Proposes an efficient convolutional algorithm to exploit the sparsity of volumetric point cloud data. Uses L1 regularisation and Rectified Linear Unit (ReLU) to maintain sparsity.	Assumes fixed sizes for all detected objects, limiting the detection performance.
	VoxelNet [75] SECOND [76]	Uses raw 3D points to learn a volumetric representation through Voxel Feature Encoding layers. The volumetric features are used for 3D region proposal.	Expensive 3D convolutions limits time performance. Models are class specific, thus multiple models must be run in parallel at run time.
Raw Point Cloud	PointRCNN [77]	Uses raw 3D points to learn point-wise feature vectors through Pointnet++ backbone. Segment background/foreground masks and generate 3D bounding box proposals based on point-wise features.	Models are class specific and struggle with smaller objects such as pedestrians.
	3DSSD [78]	Efficient point cloud sampling through Farthest Point Sampling (FPS) based on Euclidian and point feature distance. The proposed sampling scheme reduces the number of background points and increases recall of objects while reducing the processing cost of the system.	Model architecture requires tuning according to the complexity of the data, *e.g.* different models for KITTI and nuScenes.
Multi-Representation	PV-RCNN [79] M3DETR [80]	Exploits the strengths of multiple input representations (projection, volumetric and raw point cloud) to improve the detection performance.	Increased computational cost due to multiple input branches and multi-scale features, which may limit practical usage in real-time applications.

2D object detection models such as [83]. The 3D bounding box detections are obtained using position and dimensions regression.

Li *et al.* [69] uses a cylindrical projection mapping and a Fully Convolutional Network (FCN) to predict 3D bounding boxes around vehicles only. The input image resulting from the projection has channels encoding the points' height and distance from the sensor. This input is fed to a 2D FCN which down-samples the input for three consecutive layers and then uses transposed convolutional layers to up-sample these maps into point-wise "objectness" score and bounding box (BB) prediction outputs. The first output defines if a given point is part of a vehicle or the background, effectively working as a weak classifier. The second output encodes the vertices of the 3D bounding box delimiting the vehicle conditioned by the first output. Since there will be many BB estimates for each vehicle, an NMS strategy is employed to reduce overlapping predictions based on score and distance. The authors train this detection model in an end-to-end fashion on the KITTI dataset with loss balancing to avoid bias towards negative samples or near cars, which appear more frequently. Meyer *et al.* propose LaserNet [84], a network that maps cylindrical projection input point clouds into a multi-modal distribution over 3D bounding boxes. The authors show that estimating a distribution over 3D bounding boxes leads to increased detection performance when compared to the standard deterministic single bounding box estimate. Furthermore, the network uses a fully convolutional architecture which provides faster inference times than competing methods [75, 85]. More recently, Fan *et al.* [86] show that there is a performance gap between cylindrical and bird-eye (top-down) view models, arising from two challenges. First, scale variations between near and far objects; and second, the inconsistency between the cylindrical input coordinate system and cartesian coordinate system of the output. The authors propose new components in the architecture to address these challenges and obtain comparable performance to multi-view-based methods (discussed in Section 2.3.2 – Multi-representation Methods).

While previous methods used cylindrical and spherical projections, [70, 72, 87] use the bird-eye view projection to generate 3D proposals. They differ regarding the input representation: the first encodes the 2D input cells using the minimum, median and maximum height values of the points lying inside the cell as channels, while the last two use height, intensity and density channels. The first approach uses a Faster R-CNN [30] architecture as a base with an adjusted refinement network that outputs oriented 3D bounding boxes. Despite their reasonable bird-eye view results, their method performs poor orientation angle regression. Most lidar base methods use sensors with high point density, which limits the application of the resulting models on low-end lidar sensors. Beltran *et al.* [72] propose a novel encoding that normalizes the density channel

based on the parameters of the lidar being used. This normalization creates a uniform representation and allows to generalise the detection model to sensors with different specifications and number of beams.

One fundamental requirement of safety-critical systems deployed on autonomous vehicles, including object detection, is real-time operation capability. These systems must be able to perform real-time inference to allow the vehicle to respond to changes in the environment. Complex-YOLO [70] focus on efficiency using a YOLO [88] based architecture, with extensions to predict the extra dimension and yaw angle. While classical RPN approaches further process each region for finer predictions, known as two-stage detectors, this architecture is categorized as a single-shot detector, obtaining detections with a single forward step. This allows Complex-YOLO to achieve a runtime of 50 fps, up to five times more efficient than previous methods, despite inferior, but comparable detection performance. The follow up work by the same authors, Complexer-YOLO [89], extends the previous architecture to incorporate multi-target tracking and visual features obtained from camera-based semantic segmentation. Similarly, PIXOR [85] devises a Bird Eye View (top-down) projection that encodes the binary occupancy of a voxel grid but uses 2D convolutions and Pyramid-based upsampling [90] to aggregate the features across different scales. They then output object predictions in a single stage, without refinement or anchors, which improves the computational performance while reducing the number of hyper-parameters such as anchor sizes. Luo *et al.* [91] explore the same Bird Eye View projection as previous methods and incorporate temporal information by concatenating the voxel grids over multiple frames. This allows them to both increase the object detection performance as well as forecast objects' future positions based on the historical input data.

Quantifying the confidence of predictions made by an AV's object detection system is fundamental for the safe operation of such vehicle. As with human drivers, if the system has low confidence on its predictions, it should enter a safe state to avoid risks. Although most detection models offer a score for each prediction, they tend to use *softmax* normalization to obtain class distributions. Since this normalization forces the sum of probabilities to unity, it does not necessarily reflect the absolute confidence on the prediction. Feng *et al.* [71] uses a Bayesian Neural Network to predict the class and 3D bounding box after ROI pooling, which allows to quantify the network confidence for both outputs. The authors quantify epistemic and aleatoric uncertainties. While the former measures the model uncertainty to explain the observed object, the latter relates to observation noises in scenarios of occlusion and low point density. They observed an increase in detection performance when modelling aleatoric uncertainty by adding a constraint that penalizes noisy training samples.

Volumetric Convolutional Methods

Volumetric methods assume that the object or scene is represented in a 3D grid, or a voxel representation, where each unit has attributes, which may include a binary occupancy state, a continuous point density or a learned embedding. One advantage of such methods is that they encode shape information explicitly. However, as a consequence, they require a significant amount of memory to store the environment, which results in reduced efficiency since most of the volume is empty, due to the sparsity of lidar point clouds. Additionally, some methods in this category use 3D convolutions over the volumetric voxel representation, drastically increasing the computational cost of such models.

Seminal work [73, 74] address the problem of object detection on driving scenarios using a one-stage FCN on the entire scene volumetric representation. This one-stage detection differs from two-stage where region proposals are first generated and then refined on a second processing stage. Instead, one-stage detectors infer detection predictions in a single forward pass. Li *et al.* [73] uses a binary volumetric input and detects vehicles only. The model's output maps represent the probability of a vehicle being present in a particular location and the corresponding BB vertices predictions, similarly to the authors' previous work [69]. The model relies on expensive 3D convolutions which limits temporal performance. Aiming at a more efficient implementation, [74] fixes BB sizes for each class but detects cars, pedestrians and cyclists. This assumption simplifies the architecture and together with a sparse convolution algorithm greatly reduces the model's complexity at the cost of detection performance.

In VoxelNet [75], Zhou *et al.* uses raw point subsets to generate voxel-wise features, creating a uniform representation of the point cloud, as obtained in volumetric methods. The first step randomly selects a fixed number of points from each voxel, reducing evaluation time and enhancing generalization. Each set of points is used by a voxel-feature-encoding (VFE) layer to generate a voxel-wise feature vector that is aggregated into a 4D volumetric representation. This representation is fed to 3D convolutional layers, followed by a 3D region proposal network to predict the objects' location, size and class. In further work Zhou *et al.* [92] propose a dynamic voxelisation process which prevents information loss due to the sampling process when creating the voxel grid in VoxelNet. They also propose a multi-view representation: the standard BEV is used as it preserves objects' dimensions in real coordinates while avoiding occlusions and the perspective view (spherical projection) is used to provide a dense map in the immediate surroundings of the vehicle. The two-stage network using the multi-view representation shows a significant detection performance improvement when compared to the previous models. The multi-view representation significantly increase the detection performance

of the model compared to the VoxelNet baseline.

SECOND [76] leverages the VoxelNet architecture but proposes an efficient sparse 3D convolutional implementation that takes advantage the sparsity of the voxel representation to avoid computing the convolution over empty voxel grids. They enjoy an increase of temporal performance of four-fold for training and three-fold for inference when compared to VoxelNet.

PointPillars [93] provide another encoding strategy: they first voxelise the point cloud over the x-y plane (ignoring the height dimension), then aggregate mean statistics of the points in each voxel and compute a pillar feature using a PointNet [36] model. They use the name pillar to denote a voxel without constraint on the height dimension. Each pillar feature is concatenated in a dense tensor encoding the position and features of each pillar. This dense tensor has approximately 97% sparsity, so the encoding process is very efficient. The final detection uses 2D convolutions over the pillar representation and computes the objects' bounding boxes using a single-shot detection network, which allows for a significantly faster inference speed when compared to previous methods.

HotSpotNet [94] processes a voxelised point cloud into a set of hotspots, each representing a non-empty voxel within the bounds of an object. The network then leverages an anchor-free detection head based on the set of detected hotspots and their spatial relationships. This approach is a bottom-up approach - deriving objects from its parts (hotspots) - as opposed to the traditional anchor-based approach where proposals are first estimated and then refined in a top-down manner. The results show that this bottom-up method has significant advantages in the detection performance for small objects, such as cyclists and pedestrians, in the KITTI dataset, when compared to previous methods [75, 76, 93].

Raw Point Cloud Methods

Point clouds consist of a variable number of 3D points sparsely distributed in space. Therefore, incorporating their structure to traditional feed-forward deep neural networks pipelines that assume fixed input data sizes is not a trivial task. Previous methods attempted to either transform the point cloud raw points into images using projections or into volumetric structures using voxel representations. A third category of methods handle the irregularities by using the raw points as input to a neural network in an attempt to reduce information loss caused by either projection or quantization in 3D space. In this category multiple ways have been proposed to incorporate the data from the variable sized point clouds, for example, by employing Graph Neural Networks (GNNs) [95]. First, the seminal work within the raw point cloud category is revised; next its usage within driving applications is reviewed.

The seminal work in the category is introduced by PointNet [36]. Point clouds of single objects are used as input to perform object classification and part-segmentation. The network performs point-wise transformations using Fully-Connected (FC) layers and aggregates a global feature through a max-pooling layer, ensuring independence on point order. Experimental results show that this approach outperforms volumetric methods on the same tasks [35, 96]. This model is further extended in PointNet++ [97], where each layer progressively encode more complex features in a hierarchical structure, similar to the connectivity of convolutional layers on image data. The model generate overlapping sets of points and local attribute features are obtained by feeding each set to a local PointNet. Follow up work by Wang *et al.* [95] further generalize the PointNet architecture by considering points pair-wise relationships.

The seminal methods assumed segmented point clouds that contain a single object, but the gap between object classification and detection is still an open question. In PointRCNN [77] the authors bridge this gap by using a PointNet [36] backbone that creates point-wise feature vectors. These vectors are then used to create background/foreground masks and generate 3D bounding box proposals. The proposals are further refined using local point feature from the previous step in a second stage network. STD [98] adopts a similar approach, however uses a PointNet++ [97] backbone and introduces a voxelisation stage for bounding box refinement. 3DSSD [78] introduces point cloud sampling based on Euclidean and Feature distance as an efficient mechanism to generate object centre proposals. Each proposed region is processed independently from the global context in a single stage using a PointNet++ based backbone. This formulation allows for an efficient parallel model that can explore prior knowledge to generate proposals without global context. Their detection results are on-par with two-stage methods such as PointRCNN [77] while having lower computational cost due to the efficient point sampling strategy.

Multi-representation Methods

Previous point cloud-based methods exclusively used one form of input representation: either projection, volumetric representations or raw point clouds. This section reviews a recent category of methods that exploit the strengths of using multiple input representations of a single point cloud to improve the overall 3D object detection performance.

Shi *et al.* propose PointVoxel-RCNN [79] the seminal work introducing the use of multiple input representations of a point cloud for the purpose of 3D object detection. The authors use a voxelised volumetric representation to efficiently learn 3D object proposals. Next, a PointNet-based [36] network

process proposal-specific features with multiple receptive fields. The authors show that the proposed PointNet-based aggregation allows encoding richer context information when compared to conventional pooling operations (fully convolutional networks). Furthermore, these context-rich features allow for more accurate estimates of objects' position and confidences.

More recently, M3DETR [80] combines different point cloud input representations (volumetric voxels, bird-eye projection and raw point clouds using PointNet-based [36] encoder) with different feature scales per representation using multi-scale feature pyramids [90]. The different representation feature branches are fused using a Transformer [99] architecture. The authors show in extensive results that fusing different representations at multiple scale yields significant detection performance gains, with the M3DETR outperforming existing state-of-the-art methods such as PointVoxel-RCNN [79], RangeDet [86] and PointPillars [93] on the Waymo Open [55] and KITTI [47] datasets. Although the authors do not present quantitative inference time information, M3DETR requires more GPU memory and computational power compared to single input representation methods due to its multiple input branches and associated Transformer architecture, which may limit its usage in real-time applications.

2.3.3 Multi-modal Fusion-based Methods

As mentioned previously, point clouds do not provide texture information, which is valuable for class discrimination in object detection and classification. In contrast, monocular images cannot capture depth values, which are necessary for accurate 3D localisation and size estimation. Additionally, the density of point clouds tends to reduce quickly as the distance from the sensor increases, while images can still provide a means of detecting far vehicles and objects. In order to increase the overall performance, some methods try to use both modalities with different fusion schemes. A summary of fusion methods is presented in Table 2.5.

Generally, there are three types of fusion schemes [100]:

- **Early fusion:** Modalities are combined at the beginning of the detection process, creating a new representation that is dependent on all modalities.
- **Late fusion:** Modalities are processed separately and independently up to the last stage, where fusion occurs at the bounding box level. This scheme does not require all modalities be available as it can rely on the predictions of a single modality.
- **Deep fusion:** Proposed in [100], it mixes the modalities hierarchically in neural network layers, allowing the features from different modalities

Table 2.5: Summary of Fusion-based 3D Object Detection Methods

Method	**Methodology/Contributions**	**Limitations**
MV3D [100]	Uses bird-eye and front view lidar projections as well as monocular camera frames to detect vehicles. 3D proposal network based on the bird-eye-view. Introduces a deep fusion architecture to allow interactions between modalities.	Although far objects might be visible through the camera, the low lidar point density prevents detection of these objects. Specifically, the RPN based on the bird-eye view only limits these detections. Detects vehicles only.
AVOD [101]	Uses bird-eye lidar projection and monocular camera only. New RPN uses both modalities to generate proposals. A Feature Pyramid Network extension improves detection of small objects by up sampling feature maps. New vector representation removes ambiguities in the orientation regression. Can detect vehicles, pedestrians and cyclists.	Detection method only sensitive to objects in front of the vehicle due to the forward-facing camera used.
F-PointNet [102]	Extracts 2D detection from image plane, extrapolates detection to a 3D frustum, selecting lidar points. Uses a PointNet instance to segment background points and generate 3D detections. Can detect vehicles, pedestrians and cyclists.	Since proposals are obtained from the front view image, failing to detect objects in this view limits the detection performance. This limits the use of this method at night time, for example.

to interact over layers, resulting in a more general fusion scheme.

In [103], the authors evaluate the fusion at different stages of a 3D pedestrian detection pipeline. Their model considered two inputs, a monocular image and a up-sampled depth frame from a depth camera, which are fused in the channel dimension. The authors conclude that late fusion yields the best performance, although early fusion can be used with minor performance drop.

One fusion strategy consists of using the point cloud projection method, presented in Section 2.3.2, with extra RGB channels of front facing cameras along the projected point cloud maps to obtain higher detection performance. Two of these methods MV3D [100] and AVOD [101] use 3D region proposal networks (RPNs) to generate 3D Regions of Interest (ROI) which are then projected to the specific views and used to predict classes and 3D bounding boxes. A third method [104] fuses lidar and RGB learned representation at multiple stages with a focus on complimentary multiple auxiliary tasks.

The first method, MV3D [100], uses bird-eye and front view projections of lidar points along the RGB channels of a forward facing camera. The network consists of three input branches, one for each view, with VGG [52] based feature extractors. The 3D proposals, generated based solely on the bird-eye view features, are projected to each view's feature maps. A ROI pooling layer extracts the features corresponding to each view's branch. These proposal-specific features are aggregated in a deep fusion scheme, where feature maps can hierarchically interact with one another. The final layers output the classification result and the refined vertices of the regressed 3D bounding box. The authors investigate the performance of different fusion methods and conclude that the deep fusion approach obtains the best performance since it provides more flexible means of aggregating features from different modalities.

The second method, AVOD [101], is the first to introduce an early fusion approach where the bird-eye view and RGB channels are merged for region proposal. The input representations are similar to MV3D [100] except that only the bird-eye view and image input branches are used. Both modalities' feature maps are used by the RPN, achieving high proposal recall. The highest scoring region proposals are sampled and projected into the corresponding views' feature maps. Each modality proposal specific features are merged and a FC layer outputs class distribution and refined 3D boxes for each proposal. Commonly, loss of details after convolutional stages prevents detection of small objects. The authors circumvent this by upsampling the feature maps using Feature Pyramid Networks [90]. Qualitative results show robustness to snowy scenes and poor illumination conditions on private data.

More recently, Liang *et al.* [104] proposed to fuse RGB and lidar bird-eye-view learned representations for 3D object detection using multiple auxiliary

tasks such as depth completion and ground estimation. They show that learning these auxiliary tasks in a 3D object detection pipeline improves the learned features by combining the multi-modal features at various stages of the pipeline. Their method outperforms all previous lidar-only methods and all multi-modal fusion methods in terms of 3D Average Precision (AP).

A second strategy consists of using the monocular image to obtain 2D candidates and extrapolate these detections to the 3D space where point cloud data is employed. In this category Frustum Point-Net [102] generates region proposals on the image plane with monocular images and use the point cloud to perform classification and bounding box regression. The 2D boxes obtained over the image plane are extrapolated to 3D using the camera intrinsic parameters, resulting in frustums region proposals. The points enclosed by each frustum are selected and segmented with a PointNet [36] model to remove the background clutter and the resulting set feeds a second PointNet instance that performs classification and 3D BB regression. Similarly, Du *et al.* [105] first select the points that lie in the detection box when projected to the image plane, then use these points to perform model fitting, resulting in a preliminary 3D proposal. The proposal is processed by a two-stage refinement CNN that outputs the final 3D box and confidence score. The detections in both these approaches are constrained by the proposal on monocular images, which can be a limiting factor due to the limitations of this modality, *e.g.* lighting conditions, *etc.* Also in this category, RoarNet [106] uses the 2D detections to estimate feasible 3D object poses based on geometrical constraints and reduce the volume of the 3D frustum, improving the efficiency of the algorithm and alleviates synchronicity requirements between the lidar and the camera.

Fusion methods obtain state-of-the-art detection results by exploring complimentary information from multiple sensor modalities. While lidar point clouds provide accurate depth information with sparse and low point density at far locations, cameras can provide texture information which is valuable for class discrimination. Fusion of information at feature levels allow to use complimentary information to enhance performance.

2.3.4 Summary

Monocular 3D object detection methods have limited detection performance when compared to lidar-based detectors due to the lack of depth information in images [104]. In contrast, point cloud-based methods have accurate depth information that can be used to detect objects accurately. Among point cloud-based methods, the projection subcategory received initial attention due to the proximity to standard image object detection. Volumetric methods have been widely explored and research into improving the efficiency of 3D convolutions on

sparse data has made such methods more competitive in terms of computational performance. More recently, PointNet methods have emerged as efficient means to learn 3D shape information directly from irregular data (3D points) and have shown remarkable detection performance with reduced computational load by leveraging efficient point cloud sampling strategies [78]. Still, lidar-based methods have limited performance for far objects due to the sparsity of the points as the distance from the sensor increases. Multiple multi-modal fusion methods were proposed to leverage the strengths of each sensor modality, generally resulting in improved detection performance when compared to single modality detection methods. Despite such efforts, occlusions and limited field-of-view still affect both sensor modalities, which causes challenges in detecting objects that are occluded and/or distant from the sensors. To mitigate such problems, cooperative 3D object detection methods leverage observations from multiple sensors spatially distributed in the driving environment.

2.4 Cooperative 3D Object Detection

The methods reviewed in the last section used either a single sensor or multiple sensors on board a vehicle, *i.e.* from a single point of view, which is the mainstream approach for perception in autonomous driving vehicles. However, these single-view methods are inherently vulnerable to a major category of sensor impairments that can indiscriminately affect various modes of sensing. These limitations include occlusion, restricted perception horizon due to limited field-of-view and low-point density at distant regions. To this end, cooperation among various agents emerges as a promising remedy for such problems. For this purpose, information from single modality, spatially diverse sensors distributed around the environment is fused as a remedy to spatial sensor impairments as alluded above. The benefits are many-fold: for example, observations of the environment from diverse poses increase the perception horizon, increase the density of point clouds, and hence reduce the adverse impacts of sensing noise. Table 2.6 summarises existing cooperative 3D object detection methods.

Preliminary cooperative perception studies for driving applications focus on fusing off-board track data with the ego-vehicle detections to improve tracking using differential GNSS measurements and Kalman filters [111]. Following studies leverage off-board sensor data to improve path planning [112], intention-awareness [113] and decision making [114]. Still, the majority of these preliminary works do not focus on the object detection problem, rather, fuse the tracking information of detected objects [111], or consider the fusion of raw sensor data to create occupancy maps that are exploited for planning and control tasks [112, 113].

Focusing specifically on cooperative 3D object detection, Chen *et al.* [107]

Table 2.6: Summary of Cooperative 3D Object Detection Methods

Method	**Methodology/Contributions**	**Limitations**
Cooper [107]	Fuses raw point clouds of two vehicles. Propose a network architecture for sparse point clouds. Considers communication costs involved in cooperative perception.	Evaluation limited to two vehicles in a small environment or using data from the same vehicle at different time instants. Lacks quantitative analysis of precision-recall metrics.
F-Cooper [108]	Proposes feature-level fusion, as opposed to raw point cloud fusion. Analyse trade-offs in processing time, bandwidth and detection performance between fusion schemes.	Evaluation restricted to two vehicles in a specific driving scenario (parking lot). Relative-pose sensitivity analysis ignores angular errors.
V2VNet [109]	Novel GNN aggregates feature-maps from different vehicles, allows warping features to account for time delays between vehicles. Comprehensive evaluation on a synthetic dataset using multiple sensors.	Noise in relative pose transformation degrades detection performance.
V2VNet + Pose Refinement [110]	Improves V2VNet [109] using a pose correction module that ensures global pose consistency prior to feature aggregation. Shows robustness to pose noises of up to 0.8m and 8deg.	Detection performance degrades significantly when considering realistic consumer-grade GNSS/INS pose noise levels.

proposes to fuse raw point clouds and introduces Cooper, a neural network architecture for object detection in sparse point clouds. After the point clouds are fused into a common coordinate system, the detection pipeline is the same as a standard single-sensor detection model. Their study considers communication costs and show that cooperative perception can enhance the performance of object detection in terms of the number of detected objects and detection confidence. However, their study lacks a quantitative analysis in terms of precision-recall metrics, which is customary in the object detection literature. In follow-up work, the same authors propose F-Cooper [108], a feature-level fusion scheme and analyse the trade-off between processing time, bandwidth usage and detection performance. The feature-level fusion is similar to the deep-fusion approach reviewed in Section 2.3.3, where the raw data is pre-processed by a network in each vehicle, resulting in a feature map, which can then shared with other near-by vehicles. Once a feature map is received from another vehicle, it must be transformed to the local coordinate system of the ego-vehicle and merged with its local feature map. The merged feature map is then processed with a detection head which outputs the detected objects' bounding boxes. Both works [107, 108] used the KITTI dataset [47], merging two sequential frames to simulate a cooperative dataset, and their own dataset obtained with two vehicles on a parking lot. Using sequential frames from a single vehicle to simulate cooperative perception limits the diversity of evaluation data as it requires all the environment to be static across frames, otherwise the merged frames would be inconsistent. Furthermore, only two sensors are used for evaluation, which prevents understanding how the detection performance scales with the number of sensors.

Wang *et al.* proposes V2VNet [109] which uses the concept of feature-based fusion from [108] to share preprocessed data among nearby vehicles. Once the pre-processed features are received by a vehicle, they are transformed to its local coordinate system and then aggregated using a Graph Neural Network (GNN) to leverage information from all vehicles in the vicinity. A key novelty in V2VNet is its capability to account for time delays between sensor measurements. The GNN learns to interpolate the received feature vectors considering the time delay between the local time and the time when the received data was sent. The evaluation on their synthetic dataset shows that this fusion mechanism outperforms raw data fusion and high-level output fusion, while reducing the communication bandwidth between vehicles.

Before fusing the data from multiple sensors, either high level detections or low-level raw point clouds, the relative pose between sensors must be established in order to transform the data from all sensors into a common coordinate system. The presence of noise in the relative pose transformation between sensors significantly degrades the performance of object detection, since

observations become spatially misaligned. The previous methods [107, 108] considered GNSS/INS sensors to obtain the relative pose between sensors, and studied the effect of GNSS drift in the detection performance. However, their analysis considered GNSS drifts in the range of 10cm, while in practice this can range to the order of meters [115]. Furthermore, they fail to consider the rotation error that occurs due to noise in the INS measurements, which also has a detrimental effect on the results of data fusion. Wang *et al.* [109] show that their GNN aggregation module is less sensitive to sensor pose noise than raw data fusion and high-level output fusion. Further work by Vadivelu *et al.* [110] improves upon V2VNet by considering a pose correction model prior to the aggregation mechanism. A learned model uses the shared features and approximate relative pose for a pair of vehicles to output a corrected relative pose between the vehicles. These pair-wise estimates are fed to a Markovian Random Field (MRF) to ensure global consistency between the relative pose among all nearby vehicles. The results show that using such pose correction module prior to feature aggregation increases the robustness of the method to higher levels of pose noise, up to 0.8m and 8deg translation and rotation noise, respectively. Still, studies show that pose noise figures in practice are in the order of 2m in terms of translation error and tens of degrees in terms of rotation error [115, 116]. Such pose noises could render previous cooperative perception methods unfeasible in scenarios with high pose noise, particularly when using consumer-grade GNSS/INS systems. For this reason, more robust methods to estimate relative pose must be investigated, such as point cloud registration methods, which are reviewed in Section 2.6

Previous cooperative 3D object detection methods assume sensors on-board of vehicles, but do not investigate the potential of using infrastructure-based sensors. While some works propose the usage of infrastructure sensors to promote intersection automation [117], increase pedestrian safety [118], and perform object-tracking [119], these studies do not consider the problem of 3D object detection. Understanding how the pose and number of infrastructure sensors affect the performance of object detection remains a research gap. When considering infrastructure-based sensors, one of the challenges is determining the number and pose (position and orientation) of sensors. The next section presents works related to optimising the pose of sensors.

2.5 Sensor Pose Optimisation

Previous works on cooperative 3D object detection considered vehicles as the only agents sharing information, however road-side infrastructure sensors could also be used for such purpose. The advantage of the latter approach is three-fold. Firstly, the sensors are always available, independent of the traffic conditions;

Secondly, the relative pose between the sensors can be accurately determined offline through calibration processes, preventing any detection errors due to pose noise; Thirdly, using shared infrastructure resources amortises the cost of autonomous driving systems. The usage of such infrastructure-based sensing would be most useful in complex road segments, such as T-junctions and roundabouts, where occlusions and limited sensor range play key challenges in detecting other vehicles across the driving environment. A fundamental question that arises when considering such infrastructure sensors is regarding their placement, *i.e.* what is the position and viewing angles that the set of sensors should have in order to maximise the detection performance of the system.

A number of works [120, 121] suggest using elevated lidars as part of road infrastructure to improve scene understanding. Wang *et al.* [120] provide analytical results for different deploying options of Road Side Units (RSU) at driving intersections, showing that deploying evenly-spaced RSUs along the road and at intersections is the most efficient deployment strategy for road coverage. However, their assessment is based on a probabilistic road coverage model and does not provide fine pose estimates, *i.e.* where within the junction should the sensors be deployed. Furthermore, they do not explicitly consider the visibility of individual objects, which prevents identifying occlusions in cluttered road environments. The remaining of this section reviews the literature on sensor pose optimisation for generic applications, as there are no works investigating sensor pose optimisation specifically for 3D object detection.

The problem of optimal sensor placement has its historical origin in the field of computational geometry with the art-gallery problem [122], where the aim is to place a minimal number of sensors within a polygon environment in such a way that all points within the polygon are visible. Although further work extended the art-gallery problem to a 3D environment considering finite field-of-view and image quality metrics [123], it still fell short of providing realistic sensor and environmental models. Further efforts treated the pose optimisation problem as an extension of the maximum coverage problem, however use very simplistic sensor assumptions, such as radial sensor coverage in a 2D environment [124].

One category of methods perform continuous sensor pose optimisation using black-box methods including simulated annealing, Broyden-Fletcher-Goldfarb-Shanno (BFGS) [125], particle swarm optimisation [126] and evolutionary algorithms [127]; as well as white-box, gradient-based optimisation [128]. The focus of these works is on maximising the coverage (visible area) of large outdoor terrain described by digital elevation maps. Nevertheless, these works do not explicitly model the visibility of target objects and thus cannot guarantee that objects placed on covered areas will indeed be visible since different occlusion

patterns may occur, particularly when considering environments with high density of objects such as traffic junctions.

Recent work by Saad *et al.* [127] uses a visibility model with a Line-of-Sight (LoS) formulation and introduces constraints over locations where sensors can be placed and detection requirements which are user-specified, then optimise the sensor pose to achieve the detection requirements using a genetic algorithm. Temel *et al.* [129] uses a LoS binary visibility model and a stochastic Cat Swarm Optimisation to maximise the coverage of a set of sensors. The aforementioned methods consider the coverage of terrain area described by digital elevation maps and corresponding LoS visibility algorithms [125, 126] that only work for an elevation map formulation.

The visibility model in previous works commonly adopt simplifying assumptions that hinders the usage of such methods in practical settings. Examples include using a simplified 2D visibility model that does not take into account pitch and yaw [130–132] or assuming horizontal cameras focusing on a single target object without occlusions [133]. In cases where occlusion is considered [134], the visibility model is based on a simplified geometrical model that only captures information about the centroid of objects, and thus cannot capture partial occlusion, which is very common in practice. Furthermore, works using LoS visibility models based on the Bresenham's algorithm [125, 127] and derived methods [126] only work assuming an environment represented by an elevation map which cannot be used to represent objects off the ground.

Given the difficulty of optimising the sensors' pose as continuous variables, a large category of methods consider a discrete approach, where a subset of candidate sensors must be chosen to maximise the binary visibility of target points [132, 135–137]. This formulation allows to solve the problem using Integer Programming (IP) solvers [132], which can be computational infeasible for large environments, or using a variety of approximated methods including Simulated Annealing [135, 136] and Markov-Chain Monte Carlo sampling strategies [135, 137]. Although some methods in this category allow to add constraints such as the number of sensors that must observe a given target [135], this category of methods can only model target visibility as a binary variable. However, many tasks of interest such as object detection and tracking require a minimum visibility over the target objects which cannot be guaranteed by single binary variables, *i.e.* objects may have different degrees of visibility depending on their position *w.r.t.* the sensors and other objects.

2.6 Point Cloud Registration

Aligning point clouds in the same coordinate system is a preliminary requirement before their fusion can happen. Alternatively to using GNSS/INS sensors,

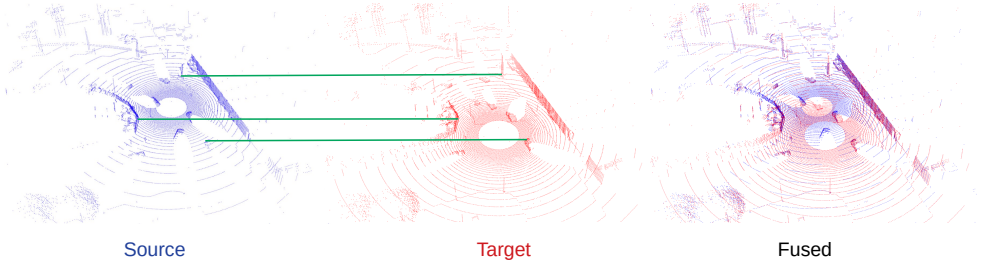


Figure 2.1: Example of Source and Target Point Clouds. Each correspondence is represented by a green lines. Once both point clouds are in the same coordinate system they can be fused into a single point cloud, as illustrated in the right-most image.

the point clouds generated by a pair of lidars can be used to obtain the relative pose transformation between their coordinate systems. The problem of point cloud registration consists of finding the rigid relative pose transformation that aligns the coordinate systems of two point clouds, commonly named as source and target point clouds. This is a key problem in many downstream applications including 3D scene reconstruction [138], localisation [139] and Simultaneous Localisation And Mapping (SLAM) [140]. The rigid relative pose transforms the points from the source point cloud to the coordinate system of the target point cloud and is parametrised a rotation matrix $R \in SO(3)$ and a translation vector $t \in \mathbb{R}^3$. In this notation, $SO(3) = \{R \in \mathbb{R}^{3\times3} : RR^\mathsf{T} = R^\mathsf{T}R = I, \det R = 1\}$ is a Lie group also known as the 3D rotation group [141]. This transformation may also be represented as $T = \left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right] \in SE(3)$, where SE is the Special Euclidean group [141]. This rigid transformation can be obtained by identifying correspondences between the pair of point clouds. A correspondence is a pair of points, one in each point cloud, that can be identified as the same point, as illustrated by the red lines in Figure 2.1. Existing registration methods in the literature can be divided into three categories: local methods, global methods using hand-crafted features and global methods using learned features. Global methods estimate the relative pose between point clouds without prior pose information. This is generally achieved by identifying correspondences between the point clouds through feature matching, where features may be hand-crafted or learned, or relative pose regression (learned-based). On the other hand, local methods require an initial estimate of the relative pose between the point clouds and are used to refine this initial estimate. Each registration method category is reviewed below.

2.6.1 Local Methods

Local registration methods are used to refine an initial relative pose estimate between a pair of point clouds. The Iterative Closest Point (ICP) [142] iteratively refines the transformation parameters in two steps. First, it computes correspondences between the point clouds by finding the closest point in the target point cloud to each point in the source point cloud. Second, it computes the relative pose transformation parameters that minimises the re-projection error between the correspondences obtained in the first step. The source point cloud is transformed using the parameters obtained in the previous step. This process is repeated iteratively until a stop criteria is met. This method is highly sensitive to the initial pose estimate, and converges to non-optimal local-minima results when the initial pose estimate is poor [143]. To mitigate this, other works estimate global optimum solutions for ICP considering branch-and-bound search over the transformation space [143, 144]. However, such global methods have significantly higher execution times, which prevents their usage in real-time applications.

Another class of local methods considers probabilistic and correlation-based methods to perform registration. Probabilistic methods such as 3D-NDT [145] and Generalized-ICP [146] use probabilistic models to describe the point clouds and find the optimal transformation using iterative numerical optimisation. Correlation-based methods [147, 148] perform registration based on the Fourier analysis of the point clouds in the spherical domain.

2.6.2 Global Methods Using Hand-crafted Features

Handcrafted features can be used to find correspondences between the point clouds. Fast Point Feature Histogram (FPFH) [149] encodes the local geometry of 3D points using multi-dimensional feature vectors. Once FPFH features are computed for each point, the correspondences are obtained by selecting the point in the target point cloud with closest FPFH feature distance for each point in the source point cloud. The correspondences obtained by comparing FPFH features are often contaminated by a large number of outliers, which prevents accurate registration. For this reason, Random Sample Consensus (RANSAC) [150] methods are used to filter out the outlier correspondences. More recently, TEASER [151] reformulates the registration problem using a truncated least-squares cost, which results in improved registration accuracy compared to RANSAC when considering a high number of outlier correspondences.

Kloeker *et al.* [152] study the problem of determining the relative pose for lidar sensors based on point cloud registration using FPFH features for initial registration and incremental updates using Generalised ICP [146] with pose-graph optimisation. Their study considers static lidar sensors whose pose

change slightly with time due to environmental factors such as wind bursts. As discussed earlier, the Generalised ICP algorithm requires a good initial estimate of the relative pose transformation. The authors use the transformation from the previous time frame as the initial estimate for a new frame, which is a valid choice under the assumption that the sensors are mostly static. However, this approach would not generalise if the sensors' pose changed significantly, *e.g.* if they were on-board of moving vehicles, since the previous pose would be a poor estimate of the current sensor pose due to the vehicles' movement. In that case, FPFH feature-matching based registration would have to applied, incurring in run-time costs between 6 and 11 seconds, as highlighted by the authors, and rendering real-time registration unfeasible.

2.6.3 Global Methods Using Learned Features

One category of learning-based registration methods focus on learning accurate correspondences between the point clouds. Generally, these methods learn a mapping from the original Euclidean space to a latent feature space and optimise the mapping such that corresponding points have a small distance in the latent space. Deng *et al.* [153] uses a PointNet [36] model to learn point-wise features and trains the model using an N-tuple loss. In contrast, [154] uses sparse fully convolutional networks to obtain voxel-wise features and trains the model using variations of triplet loss with hard negative mining. The resulting correspondences are often contaminated with outliers and need to be pruned using RANSAC [150] or further learning-based filtering [155] methods before estimating the pose transformation parameters.

Another category of methods solve the problem end-to-end by learning to directly regress the relative pose transformation parameters, *i.e.* rotation matrix and translation vector. PCRNet [156] uses a PointNet [36] model to encode a global feature vector for both source and target point clouds. These vectors are fed to a series of fully connected layers that predict the pose transformation parameters. Deep Closest Point (DCP) [157] uses a Graph Neural Network to compute point-wise feature vectors for each point cloud. The feature vectors are enhanced using a Transformer [99] attention module and are used to obtain soft-correspondences between point clouds. Finally, the correspondences are used to compute the rigid transformation through the closed-form solution to the Procrustes problem [158]. Previous methods [156, 157] assume point clouds of single objects, and its unclear to what extent these methods generalise to large outdoor point clouds. Similarly, VCR-Net [159] learns point-wise feature vectors which are used to compute soft correspondences, and obtain the transformation parameters using the closed-form Procrustes solution [158].

2.6.4 Summary

Point cloud registration methods can be used to estimate the relative pose transformation between a pair of lidar sensors. Traditional local registration methods such as Iterative Closest Point (ICP) [142] solve the problem iteratively assuming an initial relative pose. However, local methods such as ICP are not suitable for the problem due to the requirement of a initial pose, which is not necessarily available in driving applications. Feature-based correspondence matching are often contaminated by a large number outliers and must be filtered using RANSAC [150, 160] or learned models [155], which increases the registration execution time. State-of-the-art learning-based models [154, 155] require computationally demanding 3D convolutions and generate numerous putative correspondences, introducing a bottleneck on the RANSAC loop and rendering real-time execution unfeasible.

Existing registration methods assume a significant overlap between the input point clouds. This is a valid assumption for applications such as SLAM [140] and lidar odometry [161], where pairs of point clouds are obtained sequentially in adjacent time steps by a single vehicle navigating in a driving environment. However, cooperative perception and multi-agent SLAM [162] require registering point clouds obtained simultaneously from a pair of sensors on two different vehicles that are potentially far apart, and thus, may have low field-of-view overlap. Understanding the performance of traditional and learning-based methods under low overlapping point clouds is still an open question.

2.7 Research Gaps

This chapter reviewed existing perception methods for autonomous vehicles including the problems of object classification, single-vehicle and cooperative 3D object detection, sensor pose optimisation and point cloud registration. The key learnings and research gaps identified in this review can be summarised as follows.

- The predominant approach towards 3D object detection for autonomous driving applications considers one or more sensors on board of the same vehicle, which is prone to limited field-of-view, occlusions and sensor noise.
- Cooperative approaches towards 3D object detection leverage a set of observations from a number of vehicles to augment the perception horizon and increase the perception system's resilience towards occlusions and sensor noise. However, existing cooperative perception methods did not explore the potential of infrastructure-based sensors in the context of 3D

object detection. Specifically, there has been no research investigating how the detection performance scales with the number of sensors and how to optimise the sensors' pose for this application.

- Most application-agnostic sensor pose optimisation methods maximise the coverage of ground area but do not explicitly model the visibility of objects, which fails to consider occlusions and, thus, cannot guarantee that objects will be visible in particularly cluttered environments. Methods that do consider objects' visibility have limited occlusion reasoning, *e.g.* binary visibility variables, and have limited application in cluttered driving environments where complex occlusion patterns occur.

- The majority of cooperative perception methods assume nearly-perfect relative pose information between vehicles. [110] considers pose noise but still requires pose accuracy in the order of 0.8m/8deg. Studies suggest that this is unrealistic in urban scenarios where consumer-grade GNSS systems are subject to translation errors in the order of two meters and tens of degrees [115].

- Existing point cloud registration methods could be used to obtain accurate relative pose between vehicles. However, the temporal performance of global registration methods is not suitable for real-time applications such as cooperative perception. Furthermore, most point cloud registration methods are used in SLAM and odometry applications where point clouds have a significant field-of-view overlap. In contrast, pairs of point clouds obtained by different vehicles which may be far from one another will have low field-of-view overlap. The accuracy of the registration of low overlapping point clouds is yet to be investigated.

Chapter 3

Methodology

This chapter presents a concise overview of the research activities presented in this thesis and how they address the research gaps identified in Chapter 2. The research outline identifying the key learnings and contributions from each chapter is presented in Section 3.1 and illustrated in Figure 3.1. Section 3.2 discusses the implementation details of methods in this research.

3.1 Research Outline

3.1.1 Cooperative Object Classification

The literature review in Chapter 2 revealed limitations of single point sensing, including the limited field-of-view and detection performance degradation due to occlusions. To explore this further, a proof-of-concept study was designed to quantitatively measure the impact of sensor noise and occlusions on the performance of a perception algorithm, and whether these impairments can be mitigated by cooperative perception. This preliminary study, presented in Chapter 4, explores the core concept of cooperative perception upon which following studies are built. To simplify the analysis, this chapter assumes that an object has already been detected, and considers the problem of classifying the object. Specifically, this chapter assumes camera sensors that provide images of a single object which must be classified into relevant classes, *e.g.* vehicle, truck, bike, *etc.* A fusion scheme for images from multiple sensors is proposed and evaluated against single-view and other cooperative baselines. Four experiments are carried out, the first two evaluate the performance of cooperative and single-view object classification models amid fixed occlusion sizes and sensor noise power. The last two evaluate the generalisation capabilities of cooperative perception models to varying degrees of occlusion and sensor noise. The details of these experiments are described in Chapter 4. The results show that using spatially diverse images of a single object can mitigate the adverse effects of both occlusion and sensor noise on images. Furthermore, the proposed fusion

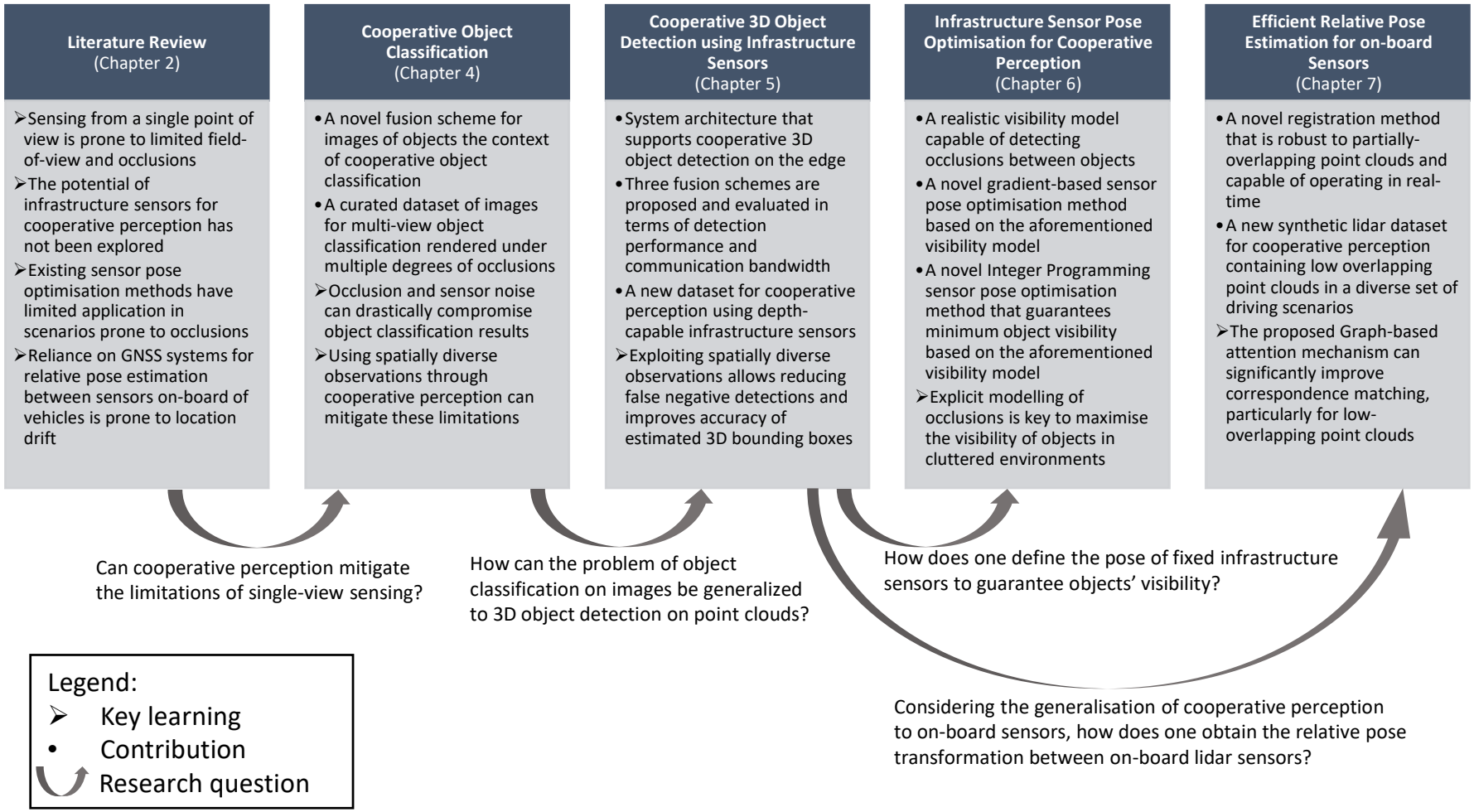


Figure 3.1: Research Outline Identifying Key Learnings and Contributions From Each Chapter.

scheme showed greater resilience to higher degrees of occlusions and sensor noise when compared to other cooperative baselines. These results validate the core concept of cooperative perception and motivate further research considering the more general problem of object detection. In summary, this chapter has the following objectives:

1. Quantify the impact of sensor noise and occlusions in the performance of object classification for single-view baselines.

2. Investigate how to fuse images from multiple sensors in the context of cooperative perception for object classification.

3. Verify the extent to which cooperative perception can mitigate the effects of occlusions and sensor noise impairments.

3.1.2 Cooperative 3D Object Detection

In Chapter 5, the preliminary concept of cooperative perception presented in the previous chapter is extended to the problem of 3D object detection, where an object's 3D position, size, orientation and class must be estimated. Colour camera sensors do not provide depth information, which poses challenges in the accurate estimation of the object's 3D position and size. On the other hand, point clouds obtained from depth sensors, e.g. depth cameras or lidars, contain the explicit 3D structure of the environment, and thus, can be fused more easily than colour images, where the depth dimension is lost. For these reasons, this chapter considers sensors that can provide point clouds. While cooperative perception methods in the literature have been limited to on-board sensors, this chapter investigates the merits of cooperative perception using infrastructure sensors. A novel system architecture that supports cooperative object detection on the edge is proposed along with three sensor fusion schemes, each exploring the fusion of data from multiple sensors at a different stages of the detection pipeline. Extensive evaluations on challenging driving scenarios are performed to assess the robustness of the cooperative object detection model and different fusion schemes. The first experiment considers the trade-off between object detection performance, computational time and communication load for the three fusion schemes. The second experiment evaluates how the number of sensors impacts the performance of the cooperative object detection model. The third experiment investigates factors that contribute to the performance of cooperative object detection, particularly, the impact of sensors with overlapping field-of-views. The fourth experiment extends the previous one by investigating how the density of lidar points over an object relates to the quality of its detection. The experiment results indicate that fusing data from multiple sensors overcomes

occlusions and restricted field-of-view, and also increase the robustness of detection by exploiting redundant observations from sensors with overlapping fields-of-view. Furthermore, these results provide practical insight into the deployment of such system by evaluating the impact of the number of sensors in the performance of 3D object detection. In total, Chapter 5 has the following objectives:

1. Investigate how to fuse the data from a set of spatially diverse sensors in the context of 3D object detection.

2. Quantify the costs of fusion algorithms in terms of communication load and computational time.

3. Measure the impact of the number and pose of sensors in the object detection performance.

4. Verify if and how cooperative perception is capable of improving the object detection performance.

3.1.3 Infrastructure Sensor Pose Optimisation

In the previous chapter, infrastructure sensors are empirically placed in the driving environment according to heuristics derived from domain-knowledge. These heuristics include the maximisation of the coverage of the traffic junction, *i.e.* ensuring that all driving areas are visible from the set of sensors; and ensuring that some sensors had a field-of-view overlap. Such heuristics are commonly found in a large category of sensor pose optimisation methods that focus on maximising the coverage of large outdoor areas. However, such approaches do not guarantee the visibility of objects in the driving environment because they fail to consider occlusions between objects. To this end, two novel sensor pose optimisation methods are proposed in Chapter 6, one using gradient-based optimisation and one using integer programming techniques, which maximise the visibility of multiple objects considering occlusions in cluttered environments. They are both based on a novel visibility model that provides pixel-level visibility information about the objects. The proposed methods are used to optimise the pose of sensors in a challenging driving environment. Their performance is evaluated by assessing the visibility of objects observed by the resulting sensor poses. The first experiment considers the comparison between the proposed methods in terms of object visibility and how it scales with the number of sensors. The second experiment compares the impact of different visibility models, particularly, the impact of occlusions between objects in the sensor pose optimisation problem. The results show that explicit occlusion considerations are key to guarantee the visibility of objects

in cluttered environments. Furthermore, the results suggest that both methods have the potential to guide the cost effective deployment of sensor networks in cooperative perception applications. In summary, Chapter 6 addresses the following research objectives:

1. Investigate how the pose of infrastructure sensors can be optimised to maximise the visibility of objects, and potentially increase the performance of cooperative perception.

2. Identify mechanisms to consider occlusions between objects in the process of sensor pose optimisation.

3. Quantify the impact of the number of sensors in the visibility of objects of interest.

3.1.4 Relative Pose Estimation

Generalising the concept of cooperative perception for infrastructure sensors, proposed in Chapter 5, to sensors on-board of vehicles requires estimating the relative pose between non-static sensors. The infrastructure sensors in Chapter 5 are considered static, and thus, can be accurately calibrated to obtain the relative pose between each sensor and a global coordinate system. Unfortunately, sensors on-board of vehicles are constantly moving and, thus, the same calibration procedure cannot be applied. Existing cooperative 3D object detection methods in the literature assume that the relative pose between sensors on-board of vehicles can be obtained accurately from GPS/GNSS systems. However, these systems are prone to drifts and multi-path losses, particularly at dense urban areas, that introduce localisation errors in the order of meters. Alternatively, the relative pose between two depth-capable sensors could be determined exclusively using the registration of their point clouds. Although this is a widely explored problem in the field of robotics and computer vision, most approaches do not comply with real-time performance, which prevents their usage for driving applications. Furthermore, existing methods exhibit limited performance for pairs of point clouds with low field-of-view overlap, which is common in the context of cooperative perception. To this end, an efficient point cloud registration method is proposed in Chapter 7. An experiment considering the performance of the proposed method and existing baselines in the literature is performed for varying degrees of overlap between point clouds. The extensive evaluation of this method on both real and simulated datasets show its resilience to low overlapping point clouds and real-time execution capabilities. Chapter 7 answers the following research objectives:

1. Investigate how to efficiently obtain the relative pose transformation between a pair of moving sensors using only their point cloud data.

2. Verify if it is possible to obtain accurate relative pose transformations when sensors have low field-of-view overlap.

3.2 Implementation Details

This section describes the data, simulation tools and software libraries used in different parts of this thesis.

3.2.1 Data and Simulation Tools

The studies in this thesis investigate the merits of cooperative perception using multiple, spatially diverse sensors that simultaneously observe a driving environment. Existing datasets for perception applications in the driving domain, including KITTI [47], nuScenes [54] and Waymo Open [55], only provide the data of sensors on-board of a single vehicle. Although such datasets could be used to simulate a cooperative perception dataset, for example, by considering the sensor data from multiple time instants as data from different vehicles, such an approach provides limited scope for research in cooperative perception, as highlighted in Section 5.2. Curating a large-scale, real-world cooperative perception dataset would incur a significant cost, due to the number of sensors, vehicles and people involved in generating and labelling such data. For this reason, most studies in this thesis create synthetic datasets to train and evaluate the performance and design choices of cooperative perception methods. This subsection provides an overview of the simulation tools used in this thesis.

In Chapter 4, Blender [163], a widely available open-source 3D computer graphics tool, is used to render 3D object models into images. Each object mesh is rendered into multiple images, considering different view-points to simulate multiple agents observing the same object. The choice for this particular renderer tool is motivated by its free and open-source availability, as well as its support for Python scripting to interface with the renderer application. Such features were useful in automating the process of generating occlusions, where the meshes of each object had to be imported, resized and an occlusion block had to be placed in a random pose relative to the object.

In Chapters 5 and 7 more advanced tools are required to simulate realistic environments containing roads and buildings, multiple types and makes of vehicles, pedestrians, cyclists, automated driving behaviour and different sensor modalities (*e.g.* lidar and depth images). Blensor [164], a fork of Blender that allows simulating lidar sensors, was considered for this purpose. However,

Blensor would still require to manually re-create the driving environment including the road network, buildings, road-side objects, vehicles, *etc.*, so this option was discarded. To create a large scale perception dataset, a simulation tool that allowed to automatically generate traffic under realistic assumptions, *e.g.* obeying traffic rules and minimum distances, is desired. Although many commercial simulation tools are available, *e.g. dSpace Sensor Simulator*, *IPG CarMaker*, these seldom allow to seamlessly simulate sensors on-board of different vehicles simultaneously, which is a primary requirement of the desired datasets in this thesis. In contrast, the free and open-source CARLA [165] simulation tool allows for the deployment of any number of sensors both in fixed locations – for the purpose of infrastructure sensors – as well as on-board sensors attached to moving vehicles. CARLA has a large community, is under active development and features several maps which range from rural areas to dense urban centres and motorways. It also supports simulating a variety of sensor modalities, including ray casting-based lidar, radar, colour camera, depth camera and semantic segmentation camera. Most importantly, CARLA can be used to automatically generate traffic following traffic rules, which in turn, allows creating a large collection of data with ground-truth annotation in an automated manner. For the aforementioned reasons, the CARLA simulation tool is adopted to synthesise cooperative driving datasets under two settings: assuming infrastructure-based lidars in Chapter 5 and assuming on-board lidars in Chapter 7. More information about these datasets is provided in Sections 5.2 and 7.3.1, respectively.

3.2.2 Software Libraries

PyTorch [166] is an open-source library that has been widely used by the research community and industry alike, has a wide community and is actively under development. For these reasons, all models proposed and evaluated in this thesis are implemented using the PyTorch library.

One of the methods proposed in Chapter 6 uses gradient-based optimisation to optimise the pose of fixed infrastructure sensors. In doing so, it requires a differentiable renderer to allow computing derivatives of a visibility cost function which depends on sensors parameters, namely, the sensors' pose. The PyTorch3D [167] is chosen as the differentiable renderer for being widely used and for its seamless integration with PyTorch [166].

3.2.3 Reproducibility

All the datasets created for the studies in this thesis are available through open-access repositories, see Appendix A. The code and trained models of most studies are also publicly available, as detailed in Appendix A, and can

be used to reproduce the results or expand upon the experiments proposed in this thesis. The code used to generate the CODD dataset, described in Section 7.3.1, is also made available. This code can be used by other researchers to generate their own custom dataset with alternative simulation parameters, including number of vehicles, driving environments, sensor configuration and sensor modalities.

Chapter 4

Cooperative Object Classification

This chapter provides a proof-of-concept study to quantitatively measure the impact of sensor noise and occlusions in the performance of a perception algorithm, and whether these can be mitigated by cooperative perception. Specifically, this chapter considers the problem of object classification in images and assumes that objects have already been detected and need only be classified. In other words, it considers the classification of images, each displaying a single object. Traditionally, a single view, *i.e.* single image, of an object is used for classification. To evaluate the benefits of cooperative perception, three multi-view object classification models leveraging the fusion of multiple images of a single object from different view points are considered.

The multi-view models are inspired by model ensembles and existing multi-modality sensor fusion methods in the literature. All models are trained and evaluated on a curated dataset of images rendered from multiple view points using 3D CAD models of objects relevant to driving applications. While existing multi-view methods consider the problem of object classification [42] and pose estimation [43] under ideal conditions, the impacts of occlusions and sensor noise that occur in practice has not been considered. To this end, a comprehensive evaluation is carried out to evaluate model generalisation and resilience to varying degrees of image occlusions and noise. The first two experiments consider how single-view and cooperative classification models trained without impairments generalise to two impairment-prone scenarios, namely, amid occlusion and amid both occlusion and sensor noise. The last two experiments investigate to what extent cooperative classification methods trained with impairments can generalise to scenarios with varying degrees of occlusion and sensor noise. The contributions in this chapter can be summarised as:

- A curated dataset of images for multi-view object classification containing

object classes relevant to driving applications rendered with multiple degrees of occlusions.

- Comprehensive evaluation of different cooperative perception methods for object classification with respect to generalisation and resilience to varying degrees of image noise and occlusions.

4.1 Problem Formulation

The problem of object classification is traditionally formulated as classifying the image of an object into a set of pre-determined classes [27]. This problem is extended to consider a set of images of the object instead of a single image. This study considers a set a set of $n = 3$ RGB images observing an object from different viewpoints, and the aim is to classify the object into one of $m = 9$ existing classes.

4.2 Object Classification Model

This section describes the proposed cooperative perception model using a novel concatenation fusion scheme, as well as two baseline models inspired by existing fusion methods in the literature. Next, the training procedure is presented.

4.2.1 Model

Deep learning models for object classification [42, 52] have outperformed classical models based on hand-engineered feature descriptors [32, 39] due to their capability of extracting prior information from large amounts of training data. For this reason, this study considers deep learning-based models for object classification. Such models map each image into a probability mass function indicating the probability of the object belonging to any particular class, and the object is classified as the class with the highest probability score. The classification model architecture and training process are described below.

The VGG-11 architecture [52] is an established benchmark model for the image classification task and shows generalisation capabilities to novel datasets [168] and tasks [169]. For these reasons, this architecture is adopted as the baseline object classification model. The architecture consists of 8 Convolutional (CNN) and 3 Fully Connected layers (FC). The convolutional layers extract features, generating an image-level feature map with dimensions 7 x 7 x 512, where the last dimension represents the number of channels of the feature map. The FC layers transform this feature map into a vector of class probabilities, which is normalized using the *soft-max* activation function ensuring the class probabilities form a probability mass function, *i.e.* are non-negative and sum

to one. The network input consists of a single 224 x 224 RGB image, while the output represents a discrete distribution over the object classes. This baseline architecture is used to create $n = 3$ single-view models, one for each view, as illustrated in Figure 4.1.

This single-view classifier is extended to a cooperative (multi-view) perception model using three different approaches, as illustrated in Figure 4.2. The first approach, known as the "voting" method, produces a class distribution for each image of the object using the single-view baseline model, then aggregate these distributions by averaging them across views. This method borrows output averaging from model ensembles, where multiple classifiers' individual output distributions are averaged to produce more accurate results [170], however considers a single classifier, averaging the model's output for different views instead. Specifically, the voting method independently classifies all the views using the same network (*i.e.* shared weights), then averages the resulting class distributions to obtain an aggregated object class distribution. Note the assumption that all the views contain the same amount of information, thus the output average has equal weight for each view. The voting fusion method employs late fusion of multiple images, since the fusion happens at the class distribution level, after classification.

In contrast, the second cooperative baseline uses view-pooling [42] to fuse information from multiple views at the feature level. This model produces image-level feature maps using the another CNN model with shared weights across views, generating n feature maps, each with 7 x 7 x 512 dimensions. These n feature maps are fused into a single global-view feature map, with 7 x 7 x 512 dimensions, using a sampling operation along the view dimension, thus called *view-pooling*. Each element in this 3D global feature map is obtained by choosing with highest value at the respective 3D spatial coordinate out of all n feature maps. In other words, the sampling is achieved through the element-wise *max* operation along the feature maps view's axis, and inherently causes loss of information. Despite the loss of information, one of the advantages this approach is its invariance to the order at which the images are fed to the model, since the *max* operation is order invariant.

Lastly, the concatenation model removes the view-pooling operation, described in the previous method, and simply concatenates the feature maps obtained from each view branch. The images from n different views are processed independently by the same convolutional stage of the baseline model, generating a set of $n = 3$ (one for each view) feature maps, each with 7 x 7 x 512 dimensions. These feature maps are then concatenated along the channel dimensions to create a single, global-view feature map, which is then fed to a fully connected network (FC). Concatenating the three feature volumes result in a model with a significantly larger number of parameters when compared to

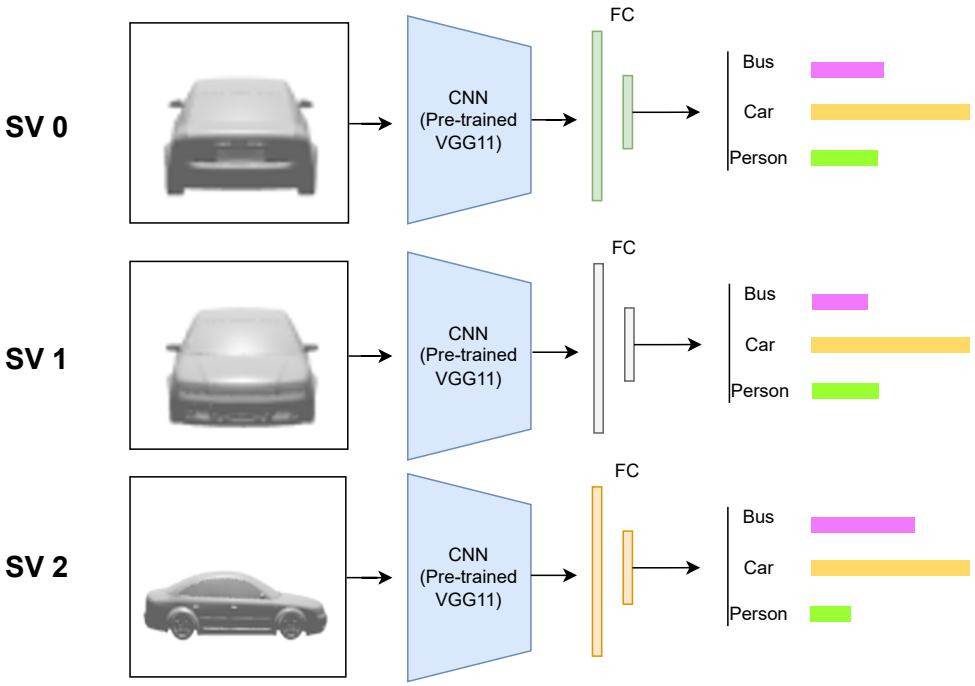


Figure 4.1: Single-view Models for Object Classification. Each model uses only one of the $n = 3$ views. Architecture components with the same colours represent shared weights. Note that all CNN shared the same VGG-11 pre-trained weights.

other baseline models. For a fairer evaluation, a "tiny" variant of the concatenation model is also considered. In this variant, each 7 x 7 x 512-dimensional feature maps are further compressed into 4096-dimensional feature vectors before being concatenated. As a result, the concatenation-tiny variant has approximately the same number of parameters of other baseline models, as highlighted in Table 4.3. Both concatenation model variants allow to exploit the information available across views, which can be beneficial in cases where objects are subject to occlusions and sensor noise. Similarly to the single-view model, the last layer output represents the class distribution.

4.2.2 Training Details

All three cooperative models and n single-view models, where n is the number of views, are trained independently on the same dataset, described in Section 4.3. Each single-view model is trained on a specific view, which allows comparing the classification performance among different views.

To avoid over-fitting to a relatively small dataset, the pre-trained model weights of a VGG-11 model is leveraged as a starting point for training. The pre-trained weights were trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset [171], which contains 1.3M

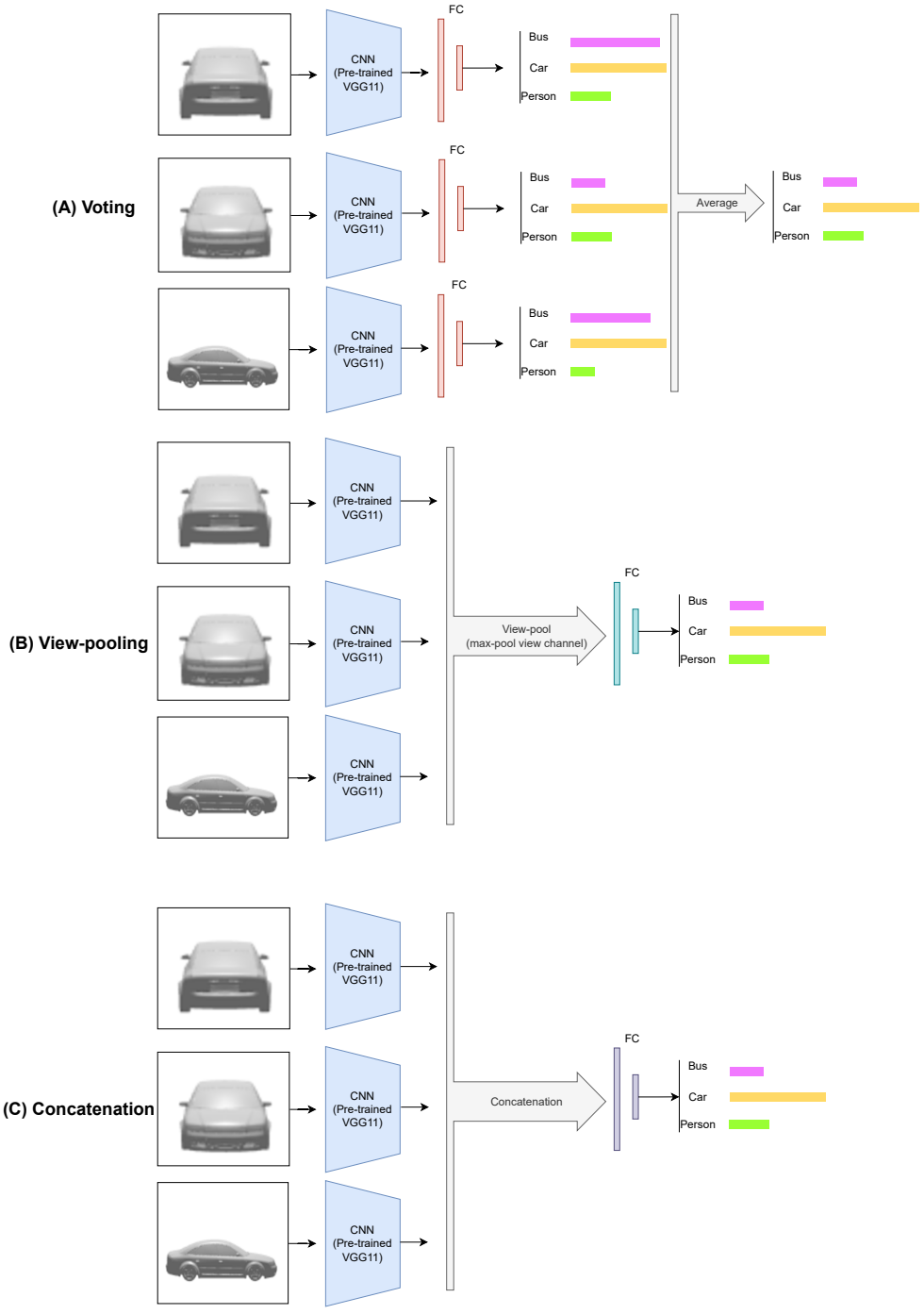


Figure 4.2: Cooperative Models for Object Classification With Number of Views $n = 3$. (a) Voting, (b) View-pooling and (c) Concatenation (proposed) Models. Architecture components with the same colours represent shared weights. Note that all CNN shared the same VGG-11 pre-trained weights.

images for training distributed over 1000 classes. The last Fully Connected (FC) layer cannot be used since the dataset proposed in this study has a different number of classes. For that reason, the last FC layer is replaced with a randomly initialized weight layer with an output unit for each class in the dataset. During training only the fully connected (FC) layers are optimised, under the assumption that the pre-trained CNN layers already provide discriminative features. This assumption is validated by attempting to optimise the parameters from the last convolutional layer without obtaining performance gains.

The loss function used to train the model consists of the weighted cross-entropy function [167], where the weight of each class is inversely proportional to its number of training samples. This addresses the problem of class imbalance and prevents the model to overfit to object classes that appear more frequently in the dataset. Dropout regularization with dropout probability $p = 0.5$ is used after each fully connected layer to further prevent over-fitting. For optimisation, the Stochastic Gradient Descent (SGD) optimiser is employed with a learning rate of 10^{-3}, momentum of 0.9. The batch sizes for all methods is 64, except for the concatenation method, which uses 32 samples per batch due to memory constraints. All models are trained independently for a total of ten epochs.

4.3 Dataset

Although a few datasets of general objects present multiple views of objects [34, 172], they use a particular camera configuration and limit the introduction of occlusions, since the images are real-photographs or pre-rendered 3D models. To be able to control the level of occlusions, *i.e.* size of the occluding object *w.r.t.* the original object, a dataset of 3D CAD object models is leveraged to render a novel multi-view object dataset with occlusions. Although there are a few options of 3D model datasets [96, 173], the ShapeNet dataset [34] offers the widest collection of 3D models. The original core dataset has 51,300 models distributed over 55 classes, while the segmentation dataset has 12,000 models over 270 classes, which are densely labelled, *i.e.* each mesh has a semantic category. A subset of relevant classes for the driving context is selected from the ShapeNet core and segmentation datasets for the collection of 3D models.

The resulting subset has 3268 object models distributed among nine classes, presented in Figure 4.3. The resulting set of 3D models are divided into training and test sets with ratio 0.7 and 0.3, respectively. The dataset is unbalanced, *i.e.* some classes have a larger number of samples than others, as observed in the histogram presented in Figure 4.3. Note that this histogram represents the 3D models and not image samples, which will be a multiple of the number of model samples, where the multiplying factors is the number of views, denoted

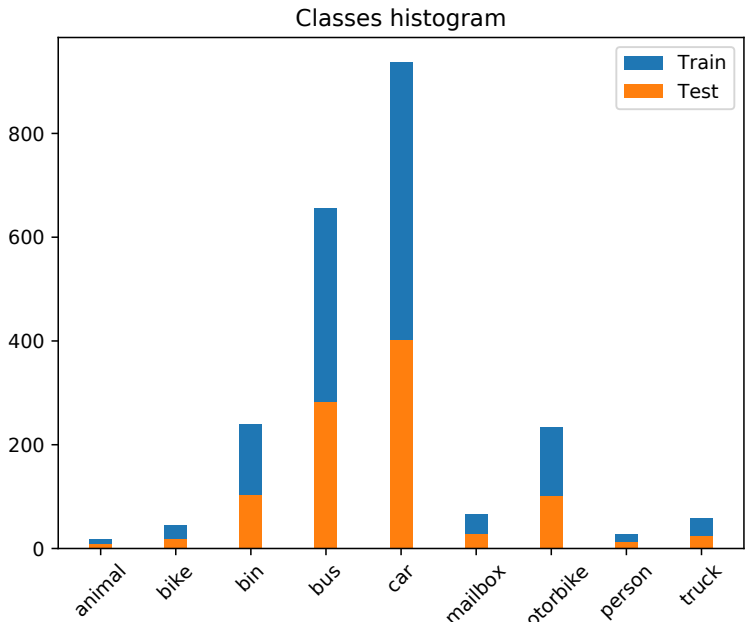


Figure 4.3: Class histogram for 3D models in the proposed dataset, divided in training and test sets. The total number of 3D objects, including both training and test sets, is 3268.

as n. The final dataset corresponds to the rendered images of the set of 3D models. Although some annotation regarding the pose of object is available, some of the 3D models were manually rotated to obtain a canonical upright pose across all samples.

The image samples are generated through a rendering process using the 3D object models. Some 3D models did not have texture information available. For consistency, all models are imported without texture information. As a result, the models cannot use colour information to discriminate between objects, which increases the classification task complexity. The objects sizes are normalized to unit size along the longest dimension, as metric sizes are not available for all models. This implies that objects such as animals may seem to have the same size as cars, for example. Considering that this object classifier would be used after a region proposal stage where the region would be cropped and scaled, this is still a valid assumption. The dataset simulates a cooperative perception setting, where a limited number of agents, *e.g.* vehicles, can share their sensor information. To this end, $n = 3$ cameras are placed in a circle centred around the target object. The cameras face the target object and are perpendicular to each other, and have a pitch of ten degrees with respect to the horizontal axis. Figure 4.4 illustrates the rendering settings with the three corresponding rendered images on the bottom row.

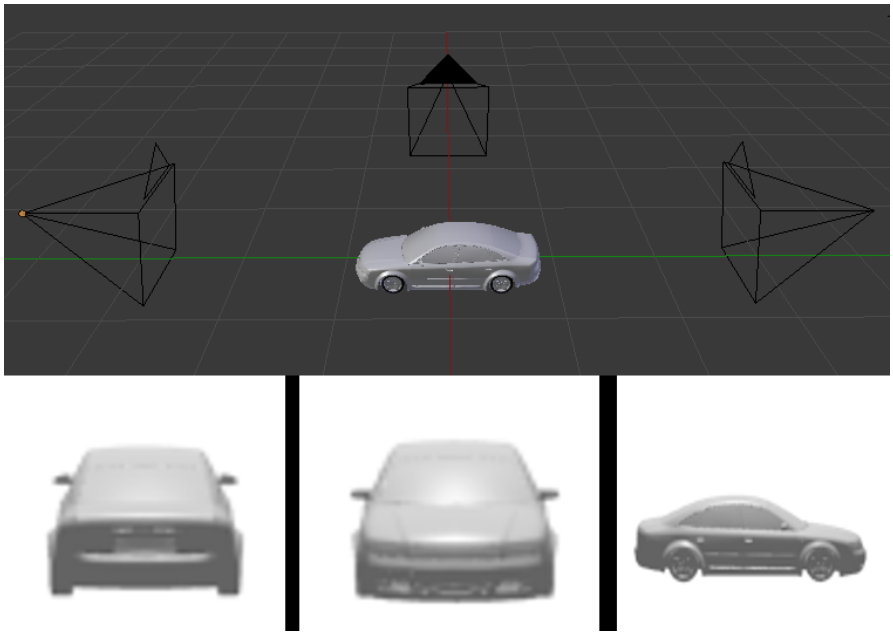


Figure 4.4: Render camera settings (Top): the three cameras are placed in a circle and tilted by 10 degrees around the X-axis. Rendering results of the cameras on a car sample (Bottom).

4.3.1 Impairment Models

Occlusions are a very common visual impairment in driving scenarios as objects can occlude other objects and themselves. To verify the impact of occlusion in the classification performance, an occlusion model is introduced. The proposed occlusion model consists of introducing an occluding object to mask part of the target object. The adopted occluding object is a cube, which is placed on a circle of radius 0.6 (relative to the object's maximum dimension of 1) around the target object, in a random angle that changes for each object sample in the dataset. This random angle is sampled from a uniform distribution between -90 and 90 degrees to ensure that the target object is occluded in at least one of the views. The cube size modulates the level of occlusion and is used as a parameter to verify different occlusion degrees.

Additionally, a sensor noise model for image sensors is introduced. Noise models for image sensors can be categorized in fixed-pattern, banding and random noise [174]. The source of random sensor noise in digital cameras can be photon emissions, photoelectric effects and thermal noise. This noise can be modelled as Additive White Gaussian Noise (AWGN) to pixel intensities [174]. Although most models use an AWGN with a signal dependent variance, a simplified version of this model is considered using a fixed Gaussian variance, which controls the intensity of the noise. Differently from the occlusion model, the AWGN can be introduced after rendering the images, not requiring to re-synthesize the dataset for each noise realisation.

Figure 4.5 illustrates sample images of the proposed dataset for three different occlusion levels and sensor noise powers.

4.4 Experiments and Results

This section presents the evaluation of the proposed object classification method and baselines. First, the evaluation metrics relevant to object classification are presented. Next, the results comparing cooperative vs single-view methods are discussed. Finally, a comparative analysis of cooperative image classification methods regarding resilience to impairments is presented.

4.4.1 Evaluation Metrics

The performance of an object detection model can be evaluated using confusion matrices. The confusion matrix is a matrix of $m \times m$ elements, with m being the number of classes, where each element C_{ij} represents the number of samples of a ground-truth class label i that were classified as class j. Considering the unbalanced nature of the dataset, the confusion matrix can be normalised along the rows, *i.e.* along the ground-truth label, to ease visualisation, creating the normalised matrix $\hat{C}$. An ideal classifier would not present any false positives or false negatives, thus, in the ideal case, $\hat{C}_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$. In practice, however, there are false positives and false negatives which reduce the classification performance. Namely, for a given class with index i, the number of True Positives (TP), False Positives (FP) and False Negatives(FN) can be computed, respectively, as:

$$\mathrm{TP}(i) = C_{ii}, \tag{4.1}$$

$$\mathrm{FP}(i) = \sum_{j=1, j\neq i}^{m} C_{ji}, \tag{4.2}$$

$$\mathrm{FN}(i) = \sum_{j=1, j\neq i}^{m} C_{ij}, \tag{4.3}$$

where m is the number of classes, and C is the un-normalised version of the confusion matrix.

The precision and recall metrics of each class can then be computed as:

$$\mathrm{P}(i) = \frac{\mathrm{TP}(i)}{\mathrm{TP}(i) + \mathrm{FP}(i)}, \tag{4.4}$$

$$\mathrm{R}(i) = \frac{\mathrm{TP}(i)}{\mathrm{TP}(i) + \mathrm{FN}(i)}. \tag{4.5}$$

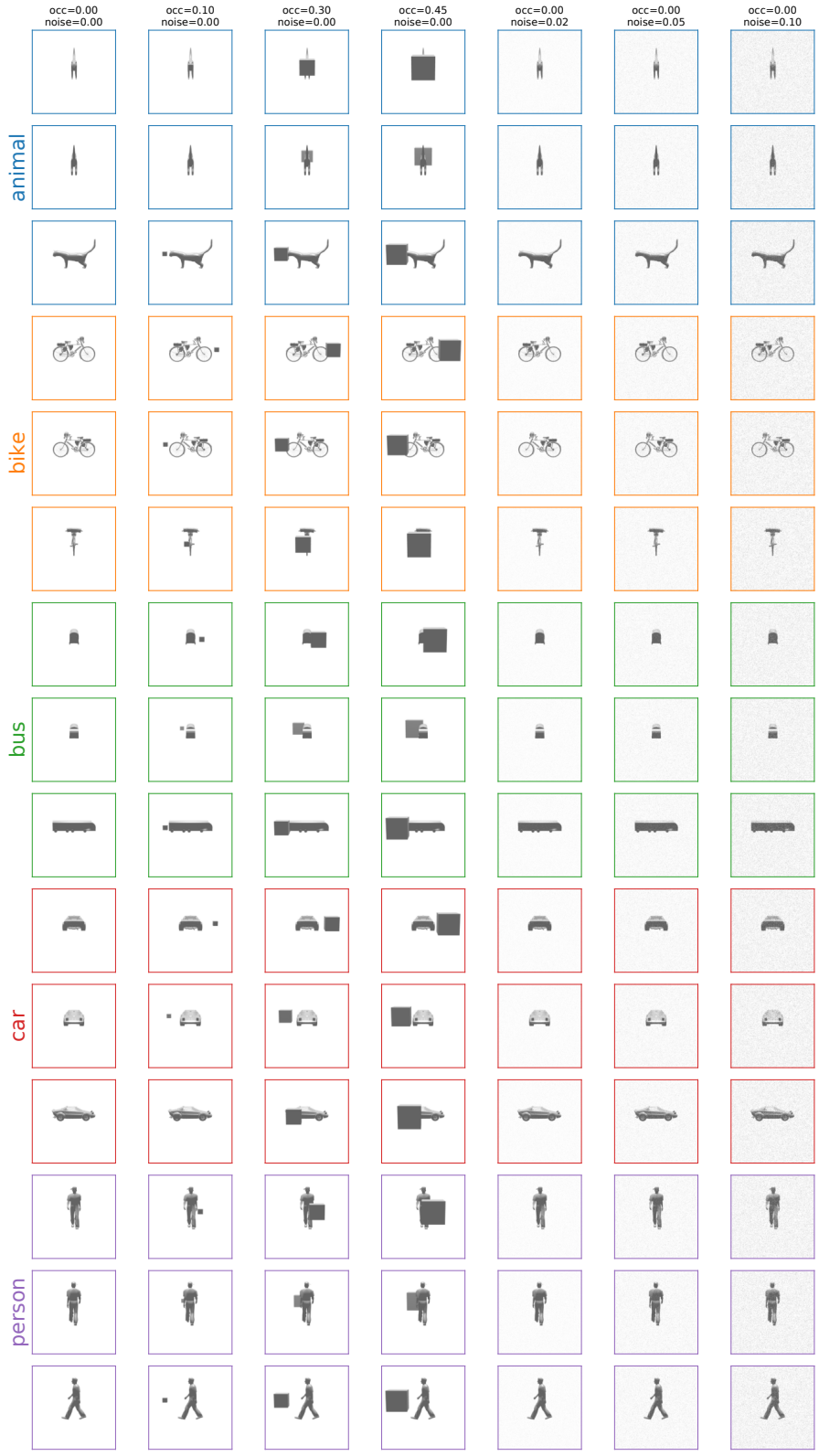


Figure 4.5: Cooperative Object Classification Dataset Sample: five classes are sampled out of the nine classes in the dataset and one object is sampled from each class. The rows show three rendered images per object (distinct colours indicate different classes) and the columns are indexed by different occlusion sizes and sensor noise powers. *Occ* indicates the occlusion level *w.r.t.* the object size, and *noise* indicates the AWGN standard deviation.

The precision metric provides the ratio of true positive samples over all samples that were classified as positives. In contrast, the recall metric provides the ratio of correctly classified samples over the total number of samples of that class. Note that for a given class, the positive samples are the samples belonging to that class and the negative samples are the samples from all other classes. Both precision and recall metrics are important, as they relate to the number of false positives and false negatives, respectively. However, using a single metric can ease the comparison between models. The F-score metric is used for this purpose, and is computed as the harmonic mean between precision and recall:

$$F_1(i) = \frac{2}{\mathrm{P}(i)^{-1} + \mathrm{R}(i)^{-1}}. \tag{4.6}$$

Note that all the above metrics are class-specific and are obtained for each individual class. The overall F-score is obtained through the weighted mean of the individual classes' F-score. To avoid the dominance of more frequent classes, the weight of each class is inversely proportional to its number of samples.

4.4.2 Cooperative vs Single-view Comparison

Firstly, the aim is to compare cooperative vs non-cooperative (*i.e.* single-view) methods by evaluating their generalisation performance under occlusions and noise impairments. Specifically, the target is to understand to what extent models trained without impairments can cope with different classes of visual impairments and whether cooperative models can better mitigate such impairments compared to single-view models. For that reason, all models are trained on the impairment-free dataset, then evaluated in three experiments. Experiment 0 provides the control condition, considering the ideal scenario where no occlusions or sensor noise is present, identically to the training condition. Experiment 1 introduces occlusion cubes with relative sizes of 0.3, which corresponds to 30% of the objects' largest dimension. Experiment 2 extends the this occlusion model by including AWGN to the pixel intensities with standard deviation $\sigma = 0.05$ (relative to the maximum pixel intensity of 1). Table 4.1 summarises the settings used for training and evaluation in the different experiments.

The results of the three experiments in the test set, in terms of the class-weighted F1-score, are presented in Table 4.2. Single-view and cooperative methods have comparable performance on the control experiment. The ranking of single-view models, trained for each view and denoted as SV0, SV1 and SV2, changes depending on the experiment conditions. The SV2 model has the best performance when the data presents no impairments and when both occlusions and AWGN are present. The SV1 model has the best performance when the

Table 4.1: Experimental Settings for Cooperative Classification Models

Experiment	Training Set	Test Set	Aim
0	Impairment-free	Impairment-free	Control experiment
1	Impairment-free	Occlusions of relative size 0.3	Evaluate generalisation to occlusions
2	Impairment-free	Occlusions of relative size 0.3 and AWGN with $\sigma = 0.05$	Evaluate generalisation to occlusions and sensor noise
3	Occlusions of 0.3 relative size and AWGN with $\sigma = 0.05$	Occlusions of relative size varying between 0 and 0.45	Evaluate models' resilience to various degrees of occlusions
4	Occlusions of 0.3 relative size and AWGN with $\sigma = 0.05$	AWGN considering σ varying between 0 and 0.15	Evaluate models' resilience to various levels of sensor noise

data presents occlusion-only impairments. Cooperative models showed better generalisation capabilities to occlusions when compared to single-view schemes: with the exception of the voting scheme, cooperative schemes outperform all single-view models in terms of the F1-score metric. This suggests that late fusion is less resilient to impairments. Considering occlusions and sensor noise, the voting and view-pooling schemes underperform single-view methods. Both concatenation model variants consistently outperforms single-view and other cooperative methods alike under all experiment conditions. This is due to the concatenation model being able to weigh the contributions from specific views without information loss caused by sampling - as in the view-pooling model, and output fusion - as in the voting model. Furthermore, the "tiny" variant underperforms the default variant due to reduced model complexity. Still, the concatenation-tiny model has comparable number of parameters than all previous baselines, which is important for a fair-comparison between models.

The confusion matrices of all methods in experiments 1 and 2 are presented in Figures 4.6 and 4.7, respectively. Noise significantly degraded classification performance, more intensely for classes that have ambiguous appearance such as "car" and "bus", as evidenced by the confusion matrices of experiment 2 in Figure 4.7. Particularly to the voting scheme, the "bus" class receives many false positives. This is also observed for non-cooperative classifiers on views 1 and 2, where trucks and cars show many miss-classifications. This phenomena happens due to the "car", "bus" and "truck" classes having similar image features and the fine distinctions between these classes being corrupted by noise. Overall, the models presented good generalisation to small occlusions (0.3 relative size), but classification performance degraded drastically with sensor noise.

Table 4.2: F1-score for Cooperative and Single-view Models in Experiments 0, 1 and 2.

Model	Experiment 0 No Impairments	Experiment 1 Occlusion	Experiment 2 Occlusion + Sensor Noise
SV0	0.971	0.887	0.733
SV1	0.960	0.898	0.640
SV2	0.979	0.845	0.759
MV voting	0.971	0.891	0.641
MV view-pooling	0.978	0.899	0.591
MV concatenation-tiny	0.984	0.936	0.924
MV concatenation	**0.987**	**0.979**	**0.959**

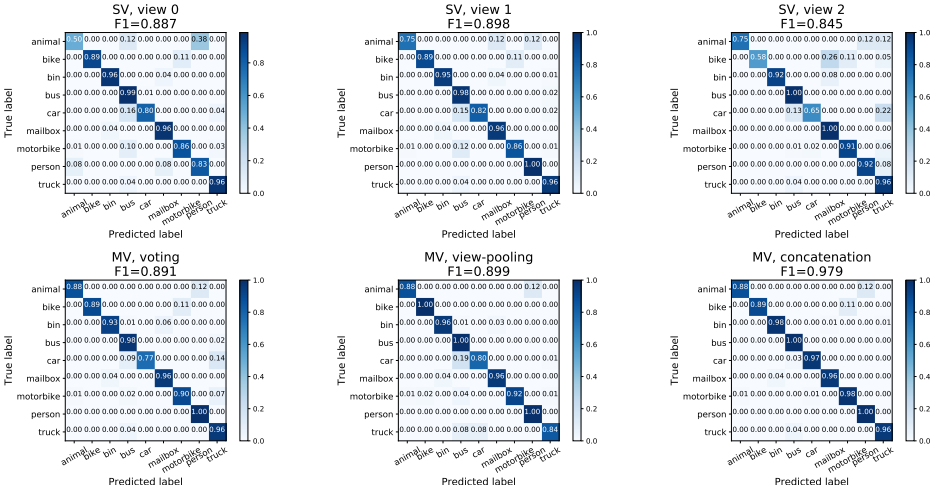


Figure 4.6: Confusion Matrices From Experiment 1: models trained on impairment free dataset evaluated with occlusion cubes sized 0.3.

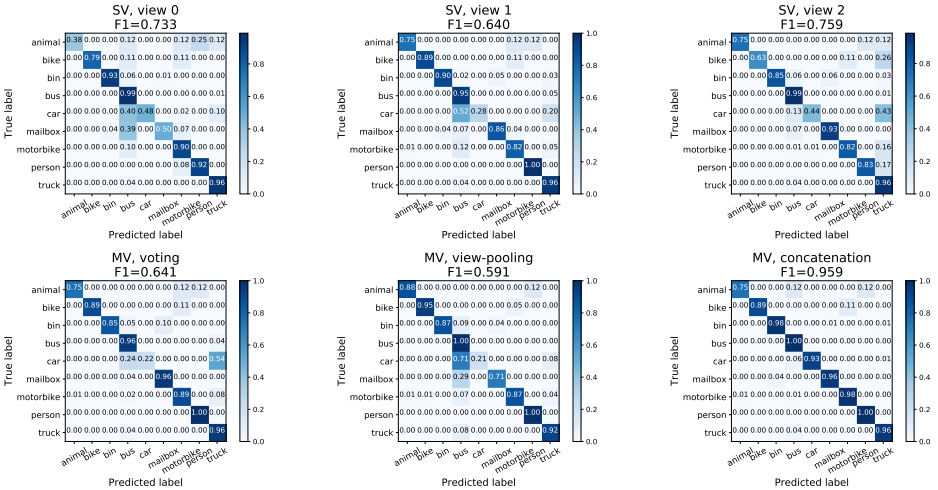


Figure 4.7: Confusion Matrices From Experiment 2: models trained on impairment free dataset evaluated with occlusion cubes sized 0.3 and AWGN $\sigma = 0.05$.

4.4.3 Resilience to Impairments on Cooperative Methods

The results in the previous section indicate an advantage in using cooperative methods for object classification regarding generalisation to unseen impairments. This section investigates the resilience of cooperative methods considering varying degrees of sensor noise and occlusion. The models are trained on a dataset containing fixed impairments levels, namely occlusion cubes with relative size 0.3 and AWGN with standard deviation $\sigma = 0.05$. The models are then evaluated on two experiments. Experiment 3 verifies the classification performance considering varying levels of occlusion, parametrised by the occlusion cube size varying between 0 (no occlusion) to 0.5 (half of the object's largest dimension) in steps of 0.05. Analogously, experiment 4 measures the performance of classifiers on an occlusion free scenario with varying sensor noise power, parametrized by the Gaussian standard deviation σ varying between 0 and 0.15 in steps of 0.01. The results are presented graphically on Figures 4.8 and 4.9, respectively.

Firstly, cooperative methods performed better when trained with occlusion compared to training with perfect perfect data in the previous experiments, as noted comparing results from Experiments 3 to 1 in Table 4.2. This is expected as the network is forced to adapt to the missing information during training time. View-pooling shows similar but superior performance to the voting scheme. The concatenation scheme outperformed both of them for all degrees of occlusion and sensor noise, most significantly when considering larger occlusion sizes and AWGN power. This performance gain can be explained due to the concatenation scheme preventing information loss, in contrast to pooling which samples and discards information and voting which only fuses high level class distributions. Although some performance drop is expected as the occlusion level increases, the concatenation model showed invariance to all the considered degrees of occlusion and sensor noise. This suggests that the model has learnt to use cues from different views to overcome occluded parts, demonstrating the capability of overcoming impairments up to occlusion boxes of size 0.4.

Despite surpassing previous methods classification performance, the concatenation model poses an important limitation for practical usage. Due to the fully connected layer after concatenating all the feature maps, the model requires a fixed number of input views, the same used during training. Not only the number of views has to be the same, but the views should have the same order (pose of the object relative to each camera), since the weights of the fully connected network will fit to each particular view. For example, if the order of the input views of the concatenation model are shuffled, the performance is expected to drop.

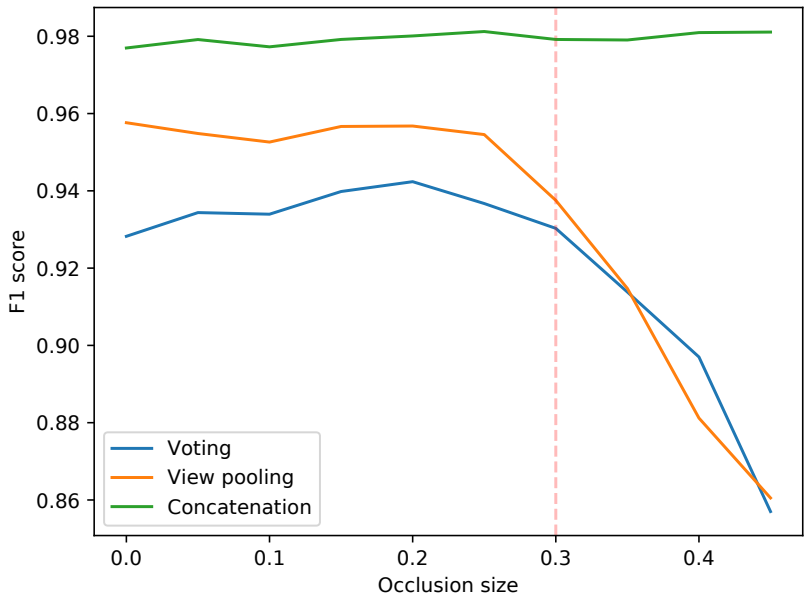


Figure 4.8: Results From Experiment 3: models trained on impaired dataset evaluated with varying occlusion levels. The highlighted vertical line indicates the occlusion level used for training.

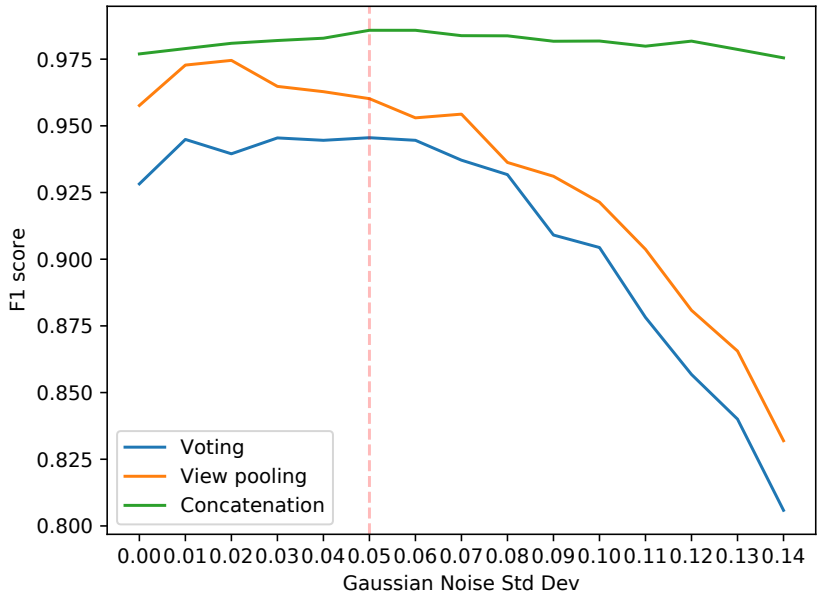


Figure 4.9: Results From Experiment 4: models trained on impaired dataset evaluated with varying sensor noise levels. The highlighted vertical line indicates the noise standard deviation used for training.

4.4.4 Time Performance

The number of parameters, indicating model complexity, and the inference time per sample (n images) for all three cooperative methods are presented in Table 4.3. The concatenation method has approximately three times more parameters since the feature maps from all three views are concatenated in the fully connected layers, which triplicates the number of parameters on the fully connected layers. In contrast, the concatenation-tiny variant has approximately the same number of parameters when compared to previous models due to the reduction in dimensionality of features before fusion. The inference time for all methods is approximately constant due to GPU processing. All results were obtained on a Quadro M4000 GPU using PyTorch [166].

Method	**Number of parameters**	**Inference time**
Voting	128.8 million	18.73 ms
View-pooling	128.8 million	18.73 ms
Concatenation-tiny	128.8 million	18.75 ms
Concatenation	334.3 million	19.57 ms

Table 4.3: Number of Parameters and Temporal Performance Evaluation of MV Models.

4.5 Summary

This chapter investigated the performance of object classification methods among occlusions and sensor noise impairments, which are common in driving applications. Three cooperative object classification methods, where multiple images of the same object seen from different view-points are exploited to generate more accurate class predictions, were evaluated. The experimental assessment took into account the model generalisation and resilience to varying degrees of occlusion and sensor noise impairments. A novel occluded object dataset is introduced with higher resemblance to real-world occlusions and with varying degrees of occlusion. The results showed that cooperative perception can improve classification performance, especially in the presence of occlusions and sensor noise, when compared to single-view methods. These results suggest that the fundamental principle of cooperative perception is promising in mitigating occlusion impairments and motivate further research considering more challenging problems, such as 3D object detection.

One limitation of the studied models is the underlying assumption that the object has been previously detected (the study in this chapter assumed images of a single object) and is aligned to a canonical pose. In a practical setting, the pose of the objects are unknown and can differ significantly from the pre-aligned

canonical poses assumed in this study. Moreover, the studied concatenation method relies on a fixed number of views, which limits the application to scenarios with a fixed number of agents (or cameras). Furthermore, this model assumes a particular order of images (each representing a particular object view), which means that the model performance is expected to drop if the images are shuffled.

In the next chapter, the problem of classifying an object from a set of images is generalised to the problem of 3D object detection, which means estimating the objects' 3D position, orientation, size and class given a set of sensor observations which may contain multiple objects. In that problem setting, part of the aforementioned limitations are naturally addressed as part of of the problem of estimating the object's position and orientation. Furthermore, other fusion schemes that are scalable with respect to the number of sensors and do not require ordered inputs are investigated.

Chapter 5

Cooperative 3D Object Detection using Infrastructure Sensors

Chapter 4 showed the potential of cooperative perception in mitigating some limitations of single-view sensing, namely occlusions and sensor noise, for the problem of object classification in images. Building on those results, this chapter investigates how cooperative perception can be used for the more general problem of 3D object detection, which consists of estimating the 3D position, size, orientation and class of objects of interest in the driving environment. Specifically, this chapter investigates different fusion methods for infrastructure sensor data in the context of cooperative 3D object detection, and studies how the number and pose of sensors impact the performance of 3D object detection, which remains a research gap in the literature.

As observed in the literature review in Section 2.3.1, monocular images do not provide depth cues to allow accurate localisation of objects in the 3D environment. As discussed in Chapter 4, fusing image data from multiple views is challenging as it requires to align the images' features in a canonical space, *e.g.* bird-eye-view or canonical object poses. For these reasons, this chapter assumes the usage of depth-capable sensors, *e.g.* lidars, which provide explicit 3D points that can be aligned in a common coordinate system. In doing so, a novel dataset is created assuming lidar-based infrastructure sensors in two challenging driving scenarios, a T-junction and a roundabout, which is used for the training and evaluation of the methods proposed in this chapter.

This chapter applies the concept of cooperative 3D object detection with two distinct fusion schemes, namely *late* and *early fusion*, and a hybrid of the two, the so called *hybrid fusion* scheme. A perception system that produces a highly accurate and reliable perception of complex road segments is proposed using a network of road-side infrastructure sensors with fixed positions. The

perception information then can be disseminated in the form of periodic cooperative perception broadcast messages to the areas of interest in real time to assist safe autonomous driving in such areas. Comprehensive evaluations are carried out to determine the impact of the number and pose of the sensors into the object detection performance. The main contributions of this chapter can be summarised as follows:

- System architecture that supports cooperative perception on the edge featuring three fusion schemes, each of them employing a distinct fusion mechanism, a bespoke deep neural network based detection and customized training procedure.
- A new dataset is synthesised for cooperative perception using up to eight infrastructure sensors that can be used for multi-view simultaneous 3D object detection.
- Comprehensive evaluations of both early and late fusion schemes, as well as their hybrid combination, are carried out in terms of detection performance and communication costs required for the operation of the system.
- The impacts of sensors and system configurations are analysed in order to provide insights into practical deployment of such systems.

5.1 System Model and Fusion Schemes

Firstly, the proposed system model for all fusion schemes is described in Section 5.1.1. Section 5.1.2 presents the data preprocessing stage, while the proposed early, late and hybrid fusion schemes for cooperative 3D object detection are described in Sections 5.1.3, 5.1.3 and 5.1.3, respectively. Finally, Section 5.1.4 introduces the 3D object detection model used in the proposed system.

5.1.1 System Model

As shown in Figure 5.1, the proposed system model considers n infrastructure sensors, each capable of depth sensing, *e.g.* lidar or depth camera, and equipped with a local processor. These infrastructure sensors are linked to a central fusion subsystem through wired or wireless data links. The sensors are assumed to be accurately calibrated, *i.e.* their absolute pose, including position and orientation, is known to the central system. The central fusion subsystem is equipped with a fairly powerful processor to fuse data from sensors and a wireless broadcast system to periodically disseminate cooperative perception messages to the vehicles in the proximity. To benefit from the cooperative

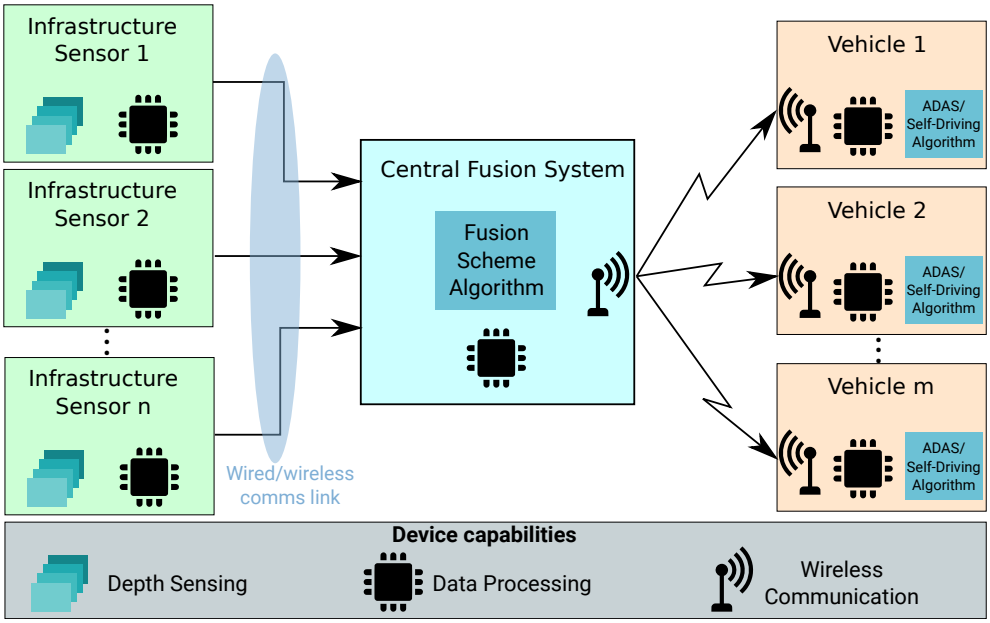


Figure 5.1: Cooperative 3D Object Detection System Model. The data provided by sensors is fused at the central fusion system resulting in a list of objects which is then shared with all nearby vehicles.

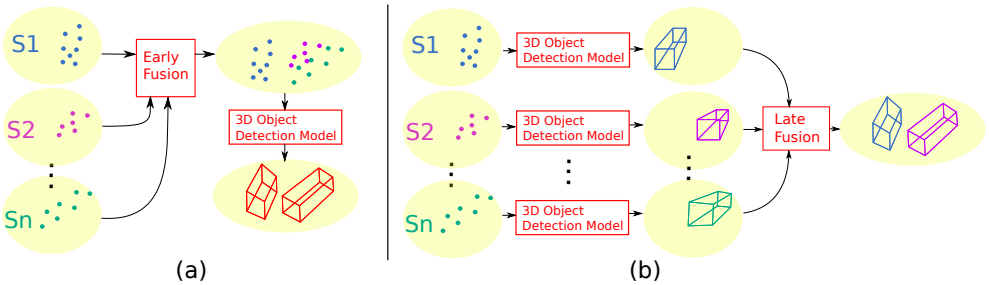


Figure 5.2: Logical Illustration of Early (a) and Late (b) Schemes for Cooperative 3D Object Detection.

perception messages, each vehicle will need to be equipped with a wireless reception system. Autonomous vehicles also are assumed to have their own onboard processing system to handle the local processing of either Advanced Driver Assistance Systems (ADAS) or autonomous driving functions, including perception and control (*e.g.* path/trajectory planning). It shall be noted that the central fusion subsystem is not responsible for the control of the vehicles. Each autonomous vehicle will therefore use on and off-board (broadcast by the central fusion subsystem) information to make their own control decisions. Hence, in the system model, the role of the central fusion system is only to assist the vehicles in making safer control decisions. Note that the network delay and communication losses are not considered in this study.

5.1.2 Data Preprocessing

The detection model considered in this chapter requires point cloud data, such as that provided by lidars or depth cameras. While lidars can produce point clouds as a standard output, the depth images produced by depth cameras can be processed (details in Section 5.2) to generate point clouds. Each sensor in a physical configuration provides points relative to its own coordinate system, thus they need to be transformed to a global coordinate system before being processed. This transformation consists of a rotation and a translation operation that maps points from the sensor coordinate system to a global coordinate system and is specified by the inverse of the extrinsic matrix of each sensor. Given the coordinates (x, y, z) of a point in the coordinate system of sensor i, the global reference point (x_g, y_g, z_g) can be obtained using:

$$\begin{bmatrix} x_g \\ y_g \\ z_g \\ 1 \end{bmatrix} = M_i^{-1} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} R_i & t_i \\ 0 & 1 \end{array}\right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{5.1}$$

where M_i is the extrinsic matrix of sensor i, which can be decomposed into a rotation matrix $R_i \in SO(3)$ and translation vector $t_i \in \mathbb{R}^3$.

The extrinsic matrix M of a sensor, and hence R and t, must be obtained through a calibration process. This can be challenging in practice for the reason that M depends on the position and orientation of sensors; hence, the result can only be as accurate as the measurements of these variables. Realistically, if these sensors are mounted on mobile nodes (*e.g.* onboard of a vehicle) any error in the localisation of the mobile node will result in alignment errors in the fused point cloud which could result in false positives and missed detections. In the system model the sensors are fixed at the road side; therefore, the calibration process can be carried out very accurately in practice [175, 176].

Once the point cloud from each sensor is transformed into the global coordinate system, all the points outside the specified detection area (defined in Section 5.2) and above 4m height are removed since such points do not carry relevant information.

5.1.3 Fusion Schemes

Cooperative perception for 3D object detection can be realised using different fusion schemes. The multi-modality sensor fusion schemes discussed in Section 2.3.3 can be applied with minor adjustments to the single-modality sensor fusion setting studied in this chapter. This study considers three fusion schemes: early fusion, late fusion and a hybrid of the two. The distinction between early and late fusion is based on whether the fusion happens before or after the object detection stage. Early fusion is defined as the aggregation of raw sensor data into a single point cloud before the detection stage. In contrast, in late fusion, each sensor observation is processed independently, generating per-sensor 3D detections, which are then fused as an end product. The hybrid fusion leverages the benefits of both early and late fusion, aiming at a lower communication cost. This hybrid solution is defined as performing early and late fusion simultaneously, however limiting early fusion to points in the medium-far range, which reduces the amount of shared sensor data.

All fusion schemes can extend the perception horizon and field-of-view of the sensing system, however the early fusion scheme can most effectively exploit complimentary information obtained from raw sensor observations. A simple illustrative example is when a vehicle is partially occluded when observed from two different sensing poses. In such case, each sensor observes a different occlusion pattern, resulting in a potentially unsuccessful detection from both observations. In contrast, when fused, these occluded observations can provide sufficient information to successfully detect the subject vehicle. For this reason the late fusion scheme is an example of high level fusion (object level), while early fusion is an example of low level fusion (signal level) [177].

Early Fusion Scheme

This scheme, as illustrated in Figure 5.2a, is based on the fusion of point clouds generated by n sensors, as depicted in Figure 5.1. This allows aggregation of complementary information from distinct parts of the objects in the detection area through spatially diverse observations, which increases the likelihood of a successful detection, particularly for objects that are occluded or have low visibility. The processing pipeline for this scheme incorporates the preprocessing stage carried out onboard of each sensor, which results in n point clouds in the global coordinate system. Each respective point cloud is transmitted to the

central fusion system where they are concatenated into a single point cloud and then fed to the 3D object detection model. The results of the object detection model in the central fusion system consists of a list of objects, *i.e.* 3D bounding boxes, which is then disseminated to the vehicles in the vicinity, as depicted in Figure 5.1.

Late Fusion Scheme

This scheme, as illustrated in Figure 5.2b, fuses the output of the 3D object detection model (a list of 3D bounding boxes) obtained locally at each sensor node. Thus, if an object is not detected in at least one of the observed point clouds, *e.g.* due to occlusion or low point density, it cannot be detected by the overall system. First, each point cloud is preprocessed and fed into the detection model onboard each sensor, which generates a list of objects represented by their 3D bounding boxes. The list of detected objects from the n sensors are then transmitted to the central fusion system, where they are fused into a single list. Considering that some objects may be in the field-of-view of multiple sensors, the aggregated list may have multiple detections for a single object. In order to mitigate this effect, a post-processing algorithm known as Non-Maximum Suppression (NMS) [178] is employed. This algorithm identifies the overlap of the detected boxes, measured by the Intersection Over Union (IOU) metric (described in Section 5.4). If the overlap between any two detected boxes exceeds a specified threshold, the box with lowest confidence score is removed. The confidence score of a detected box indicates the confidence of the presence of an object within the box, and is obtained by the 3D object detection model (detailed in Section 5.1.4). Figure 5.2b illustrates an example case where S2 and Sn observations resulted in two detections of a single object, thus, during the fusion stage the box detected by Sn is omitted. A number of detected boxes that overlap could be potentially combined to create a new detection box with higher confidence, however this would require a new model specifying the box fusion process. The conducted experiments showed that NMS was successful in eliminating the overlapping detected boxes and, thus, this algorithm is used for simplicity. Once the fusion and post-processing are completed, the resulting object list is broadcast to all vehicles in the vicinity, similar to the previous scheme.

Hybrid Fusion Scheme

The early fusion scheme can increase the likelihood of detecting objects compared to late fusion due to the aggregated information prior to the detection stage but requires raw sensor data sharing, which increases the communication cost of the system. As an intermediate solution, the hybrid fusion scheme uses

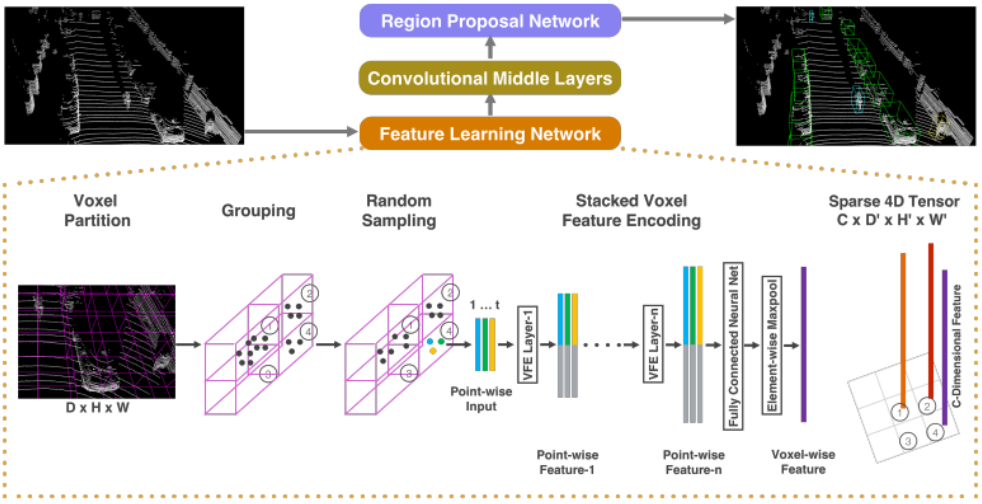


Figure 5.3: Voxelnet 3D Object Detection Model Architecture, obtained from [75].

both of the previous schemes to increase the likelihood of a detection without a drastic increase in the communication cost. The key concept is to share high level information (late fusion) where the sensor has high visibility and share low level information (early fusion) where the visibility is poor. Objects close to a sensor will have a high density of points and thus are more likely to be detected using a single sensor's observation. Thus points in the close vicinity of a sensor need not be transmitted to the central fusion system, which allows to reduce the communication bandwidth. First, the late fusion scheme is employed in each sensor node and the detected boxes are shared to the central fusion system. Next, each sensor node selects all points from its point cloud whose projection in the horizontal plane are outside a circle of radius R and share them with the central fusion system. The radius R modulates the trade-off between early and late fusion – as R decreases more raw data is shared with the central fusion system. The central fusion system then uses early fusion on the received point clouds and fuses the detected bounding boxes with the late fusion results from each sensor node. The bounding box fusion follows the same NMS procedure defined in Section 5.1.3.

5.1.4 3D Object Detection Model

The object detection model adopted in this chapter for both fusion schemes is based on Voxelnet [75], shown in Figure 5.3.. This model consists of three main functional blocks: a feature learning network, multiple convolutional middle layers and a Region Proposal Network (RPN). Each block is described below.

The feature learning network converts the 3D point cloud data into a fixed sized representation that can be processed by the convolutional layers.

Originally, the Voxelnet architecture uses the laser reflection intensity channel as well as the 3D spatial coordinates (x, y, z). Their original architecture is adapted to support using the spatial coordinates alone, which enables the model to generalise to point clouds obtained from depth cameras. The input point cloud is grouped into voxels of equal size (v_x,v_y,v_z), representing the width, length and height, respectively. For each voxel, a set of t points within its boundaries is selected to create a voxel-wise feature vector. If t is greater than T (threshold on the maximum number of points per voxel), a random sample of T points is selected, which reduces the computational load and the imbalance of the number of points between different voxels. These points' coordinates are fed into a chain of Voxel Feature Encoding (VFE) layers. Each VFE layer in the chain consists of fully connected layers, local aggregations and max-pooling operations that allow concentration of information from all voxel points into a single voxel-wi se feature vector. The output of this network is a 4D tensor, indexed by the voxel feature dimension, height, length and width.

The convolutional middle layers in the processing pipeline apply three stages of 3D convolutions to the 4D voxel tensor obtained previously. These stages incorporate information from neighbouring voxels, adding spatial context to the feature map.

The resulting tensor from the convolutional middle layers is then fed into the Region Proposal Network. This network is composed of three stages of convolutional layers, followed by three stages of transposed convolutional layers which create a high resolution feature map. This feature map is then used to generate two output branches: a confidence score indicating the probability of presence of an object and a regression map indicating the relative size, position, and orientation of a bounding box with respect to an anchor. Each element of the output branch is mapped to an anchor in a uniformly arranged grid, whose density is controlled by the anchor stride hyper-parameter. Anchors are used since the regression of detection boxes relative to an anchor gives more accurate results than regression without any prior information [30].

5.2 Dataset

At the time of writing this study, no public dataset was available to be readily used for cooperative 3D object detection in the literature. Note that authors in [107] and [108] simulate an environment where two vehicles share their sensors' information by using point clouds from the KITTI dataset [47] generated by the same vehicle at two different time instants. However, their approach is limited to a small number of scenes where all objects remain static, except for the ego vehicle, which also restricts the number and pose of sensors that can be used. Hence, it falls short of providing a comprehensive dataset to

investigate dynamic and complex driving scenarios where multiple objects are moving and/or are occluded. To this end, a novel cooperative dataset for driving scenarios using multiple infrastructure sensors is generated as described in the following.

The proposed dataset was created using the CARLA simulation tool [165], which allowed simulation of complex driving scenarios as well as obtaining accurate ground-truth data for training and evaluation. This dataset is used in this chapter to establish the underlying concepts of the proposed cooperative 3D object detection schemes and gauge their potential benefits. The choice for the simulation scenarios is motivated by the urban driving use cases of the Cloud-Assisted Real-time Methods for Autonomy (CARMA) project [179], whose funds supported this research. Within that category of use cases, the choice was guided by the scenarios offering challenging perception conditions, including occluding objects as well as having large spatial dimensions, which challenges total coverage using a single sensor. As a result, two scenarios were chosen: a T-junction and a roundabout scenario. The dataset is generated in each scenario using fixed road-side cameras, which provide RGB and depth images with resolution of 400 x 300 pixels and horizontal field-of-view of 90 degrees. The resolution and field-of-view are conservative estimates of new generation solid state lidars [180]. The T-junction scenario uses six infrastructure cameras mounted on 5.2m high posts. Three of these cameras point towards the incoming roads and the remaining three to the opposite direction of the junction. In contrast, the roundabout scenario uses eight cameras at 8m mounting posts placed at the intersections, four of them facing the oncoming lanes to the roundabout and the other four facing outwards the roundabout. The sensors' height was chosen according to the typical light poles height already available in the simulation scenarios and to conform to local UK standards [181]. Both sensor configurations were empirically positioned to fully cover the roundabout and junction, while providing some overlapping between sensors' field-of-view, as illustrated in Figure 5.4. Note that both scenarios have occluding objects, such as buildings (both scenarios) and a central statue in the middle of the roundabout.

The proposed dataset consists of four independent collections: two for the T-junction, containing 4000 training and 1000 test frames respectively, and two for the roundabout, with an equal number of training and test frames. A frame is defined as the set of depth and RGB images from all cameras corresponding to a single instant in time. Each frame also contains an object list describing the ground-truth position, orientation, size and class of all objects in the scene.

The dataset contains three classes of objects: vehicles, cyclists/motorcyclists or pedestrians. During the generation of the dataset, the maximum number of objects at any given time was set to 30, which was observed as a threshold

above which severe traffic congestion happens. The probabilities of spawning cars, cyclists and pedestrians is equal to 0.6, 0.2 and 0.2, respectively, which guarantees a higher number of cars but still allows a representative sample of cyclists and pedestrians. During the simulation, each object has a life span of four frames, which forces new objects to be spawned periodically and increase the diversity of objects and poses. The motion of the objects in the simulation is governed by traffic rules and internal collision avoidance mechanisms of the simulator. All object models available in CARLA are used during the simulation – twenty for cars (sport, vans and SUVs), six for cyclists and fourteen for pedestrians.

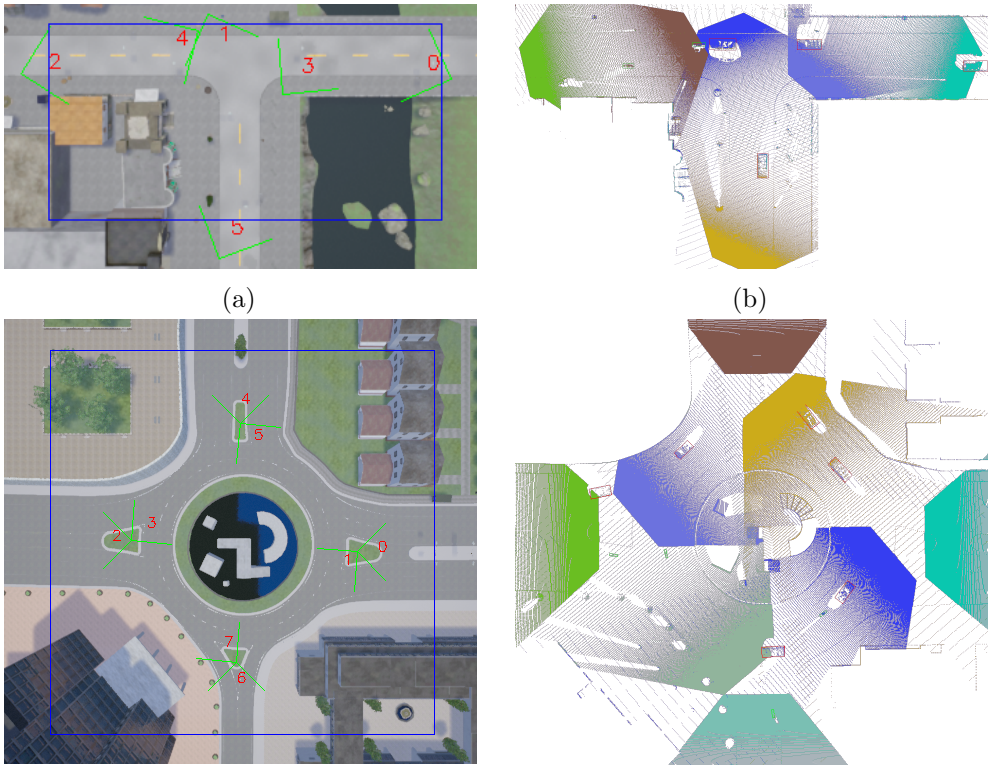
The detection areas are defined as a rectangle of 80 x 40m for the T-junction scenario and a square of 96m centred at the roundabout, illustrated by the blue rectangles in Figure 5.4a, 5.4c. These areas of interest are chosen to cover all the junction/roundabout area and some extent of the roads leading to it in order to increase the perception horizon of the system, while taking in account the constraints in the processing system memory. The T-junction and roundabout scenarios detection areas cover 3200 m^2 and 9216 m^2, respectively.

The proposed approach uses depth images, also known as depth maps, exclusively. These images represent the distance from the camera to the surface of objects in the camera field-of-view. More accurately, each pixel in a depth image specifies the distance of the projection (into the camera's Z axis) of the vector from the camera to a surface point. Each depth image is used to reconstruct a point cloud, where each pixel is transformed into a 3D point in the camera coordinate system using the pinhole camera model [182], described by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (u - C_u)\frac{d}{f} \\ (v - C_v)\frac{d}{f} \\ d \end{bmatrix}, \tag{5.2}$$

where (x, y, z) are the coordinates of the 3D point corresponding to pixel coordinates (u, v) in the depth image, C_u, C_v, f are the camera focal centre and length (given by the intrinsic camera matrix), and d is the respective depth value of pixel with coordinates (u, v). The point cloud produced by combining the 3D points from all cameras should have a similar size as one produced by a standard lidar, around 200 thousand points, for processing time constraints. To this end, the depth image resolution is downsampled in half to 200 x 150 pixels, which yields 30000 3D points per camera, and approximately 200 thousand points when combining points from six or eight cameras.

A surface agnostic Additive White Gaussian Noise (AWGN) model is introduced with mean $\mu = 0$m and standard deviation $\sigma = 0.015$m to the depth image, following the specification and mathematical model of a lidar



(a) (b)

(c) (d)

Figure 5.4: Bird Eye View of the T-Junction and Roundabout Scenarios. a and c show the sensors configuration, where each sensor is indicated by its ID number and green lines representing the field-of-view; the blue rectangles indicate the detection areas of each scenario. b and d show the fused point clouds, where each colour represents a sensor and the 3D bounding boxes represent the labelled data, with colour indication of the class (red for cars, green for cyclists and blue for pedestrians). Note that this sensor configuration fully covers the detection areas and provide overlapping field-of-views (indicated by areas with multiple colours).

sensor in [183]. It must be noted that, in contrast to lidar sensors, the depth estimation error of stereo-matching-based cameras increases exponentially with the distance between the camera and the object [184]. However, since the data is assumed to be obtained by lidar arrays, the noise is considered independent of the distance between the sensor and the object.

5.3 Training Process

The model described in Section 5.1.4 is trained using the procedure presented below. One instance of the 3D object detection model is trained for each scenario using the fused point clouds from multiple sensors. The models are trained with Stochastic Gradient Descent (SGD) optimisation for 30 epochs with learning rate of 10^{-3} and momentum of 0.9, as proposed in [75], using PyTorch [166]. The loss function is adopted from [75], and penalises the regression of position, size and yaw angle relative to a fixed anchor. The same hyper-parameters suggested in [75] are employed: a single anchor of size $(3.9, 1.6, 1.56)$m with two orientations (0 and 90 degrees) and maximum number of points per voxel $T = 35$.

The voxel size (v_x, v_y, v_z) and the anchor stride along the X and Y dimensions for the T-junction model is set to $(0.2, 0.2, 0.4)$m and 0.4m, respectively, identical to those in [75]. Using these same hyper-parameters in the roundabout model is unfeasible since the roundabout scenario has approximately thrice the area of the T-junction scenario, which would result in feature maps that do not fit in the GPU memory. Hence, the spatial resolution of the X and Y axis is reduced in half by adopting a voxel size of $(0.4, 0.4, 0.4)$m and anchor stride 0.8m for the roundabout model.

The object detection models are trained to detect vehicles only. The samples of pedestrians and cyclists are present in the dataset to avoid over-fitting as they force the model to learn distinct features for vehicles.

During the training stage, each ground-truth bounding box is rotated by a random angle with a uniform distribution in the range of $[-18, 18]$ degrees, similar to previous studies in [75, 76], to increase the generalisation of angle estimation. Rotating the whole point cloud to avoid model over-fitting to the buildings and fixed objects surrounding the junction was considered, however this operation did not result in a significant performance gain.

5.4 Performance Evaluation

The performance of the proposed cooperative perception system for 3D object detection is evaluated through a series of experiments in two scenarios, a T-junction and a roundabout. The performance evaluation is carried out on

an independent test dataset for each scenario using the metrics described in Section 5.4.1. First, the fusion schemes are compared in terms of their detection performance, computation time and communication costs for data sharing in Section 5.4.2. Secondly, the impact of the number of infrastructure sensors and their pose on the detection performance is evaluated in Section 5.4.3. Then, the benefits of fusing information from multiple sensors with overlapping field-of-view in early fusion scheme are evaluated in Section 5.4.4. Additionally, it is evaluated how the number of infrastructure sensors relates to the quality of the information acquired from the objects (in terms of the density of points in the point cloud data), and, in turn, how this number relates with the accuracy of the detected boxes in Section 5.4.5. Finally, the proposed system performance is compared to existing benchmarks in Section 5.4.6.

5.4.1 Evaluation Metrics

Four evaluation metrics related to object detection are used in this chapter, namely, Intersection Over Union (IOU), precision, recall and average precision, which is derived from the previous two. Additionally, the communication cost metric is defined as the average data volume exchanged between a sensor and the central fusion system in *kilobits (kbit) per frame*, where a frame is defined as a single operation of the whole processing chain in this chapter.

The IOU measures the spatial similarity of a pair of bounding boxes, one normally chosen from the set of estimated bounding boxes and the other from the ground-truth set, given by

$$\text{IOU}(B_{gt}, B_e) = \frac{\text{volume}(B_{gt} \cap B_e)}{\text{volume}(B_{gt} \cup B_e)}, \tag{5.3}$$

where B_{gt} and B_e represent the ground-truth and the estimated bounding boxes, respectively. The set of estimated bounding boxes includes all positive boxes, *i.e.*, those identified by the 3D object detection model in Section 5.1.4 with confidence scores greater than a threshold, denoted by τ. The IOU simultaneously takes into account the location, size, and orientation (yaw angle) of both bounding boxes. Its value ranges from 0 (when the bounding boxes do not intersect) to 1 (when the location, size, and orientation of both bounding boxes are equal). Normally, when the IOU metric for a pair (B_{gt}, B_e) is above a certain threshold, denoted by κ, B_e can be regarded as the matching estimation of B_{gt}. The IOU threshold κ is typically set to 0.5 or 0.7 [47], in this chapter it is assumed to be 0.7 unless stated otherwise.

The precision metric is defined as the ratio of the number of matched estimated boxes, according to the above definition, to the total number of bounding boxes in the estimated set. Similarly, the recall metric is defined as

the ratio of the number of matched estimated boxes to the total number of bounding boxes in the ground-truth set. It shall be noted that the precision and recall metrics are functions of κ and τ. And, there is an inherent trade off between the precision and recall metrics, described in the literature by the Precision-Recall (PR) curve [185].

The Average Precision (AP), denoted as $\mathrm{AP}_{3\mathrm{D}}$, is a single scalar value, computer by taking the average of the precision for M recall levels [185, 186]:

$$\mathrm{AP} = \sum_{n=0}^{M-1} (r_{n+1} - r_n) p_{\mathrm{interp}}(r_{n+1}), \tag{5.4}$$

where

$$p_{\mathrm{interp}}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}). \tag{5.5}$$

Here M is the number of estimated bounding boxes, $p(r)$ is the precision as the function of recall r, and $p_{\mathrm{interp}}(r)$ is a smoothed version of the precision curve $p(r)$ [185]. The recall value $r_i \in \{r_1, \ldots, r_M\}$ in Equation 5.4 is obtained considering the confidence threshold τ equal to the confidence score of the i-th bounding box within the set of estimated bounding boxes when sorted by the confidence score in descending order. Throughout this chapter the $\mathrm{AP}_{3\mathrm{D}}$ metric is used for different values of the IOU threshold κ, denoting it as $\mathrm{AP}_{3\mathrm{D}}$ @ IOU κ.

5.4.2 Comparative Evaluation of Fusion Schemes

The purpose of this experiment is to compare the performance of early, late and hybrid fusion schemes in terms of their detection performance, communication cost and computation time. The detection performance of each scheme is quantified by the $\mathrm{AP}_{3\mathrm{D}}$ metric for $\kappa \in \{0.7, 0.8, 0.9\}$. The communication cost is computed as the number of kilobits shared between sensors and the central fusion system, averaged across sensors and frames. The computational time is measured as the average time required for the system to output detection results for a frame. This time does not consider any communication delays and assumes that all sensors process data in parallel. The performance evaluation was carried out in both T-junction and roundabout scenarios in this experiment. For the late fusion scheme, the NMS algorithm uses an IOU threshold of 0.1, which was experimentally determined to remove multiple detections of a single object. For the hybrid fusion scheme, the radius R of the circle limiting the area for low level data sharing was experimentally determined for best trade-off between communication cost and detection performance to be 20m and 12m for the T-junction and roundabout scenarios, respectively.

Table 5.1 summarises the results of this experiment in terms of the $\mathrm{AP}_{3\mathrm{D}}$,

Table 5.1: Comparative Evaluation of Early Fusion (EF), Hybrid Fusion (HF) and Late Fusion (LF) Schemes

		$\mathbf{AP_{3D}}$			**Comm. Cost (kbit)**	**Comp. Time (ms)**
		$\kappa = 0.7$	$\kappa = 0.8$	$\kappa = 0.9$	**per sensor**	**per frame**
Tjunction	**LF**	0.8181	0.6259	0.07072	0.51	298
	HF	0.8903	0.7056	0.07277	64	380
	EF	0.9870	0.9447	0.3861	516	380
Roundabout	**LF**	0.8143	0.5986	0.01762	0.26	214
	HF	0.8398	0.6289	0.02013	372	299
	EF	0.9670	0.8816	0.04638	674	299

communication cost metrics and computation time for both scenarios. These results show that the early fusion scheme outperforms hybrid fusion, which in turn outperforms late fusion. More specifically, the early fusion scheme outperforms late fusion scheme up to 20% in terms of detection performance in the T-junction scenario and 18% in the roundabout scenario measured by the AP_{3D} metric with IOU threshold of 0.7. The early fusion scheme demonstrates a significantly better detection performance compared to the other schemes when considering higher values of κ, such as 0.8 and 0.9. It can also be seen that for a given value of κ, the detection performance in the T-junction scenario is consistently superior to the detection performance in the roundabout scenario. This arises from the larger voxel sizes that had to be adopted in the latter scenario, for the reasons described in Section 5.3, which reduces the spatial resolution of the system and results in less accurate bounding box regression.

The results in Table 5.1 show that the superiority of the detection performance of the early fusion scheme comes at a higher communication cost. This is due to the larger data volume required to transmit raw point clouds in early fusion compared to the transmission of the estimated objects in late fusion from the sensors to the central system. For example, in early fusion, a frame is transmitted using a depth map of 200 x 150 32-bit values, equivalent to 960 kbits, which become roughly 500-600 kbits after filtering points outside the detection area (defined in Sec 5.2); while late fusion encodes and transmits a detected object with as little as 8 32-bit values determining the position, size, orientation and confidence of the object. Furthermore, the hybrid fusion outperforms late fusion with a significantly lower communication cost than that of early fusion, due to filtering out points in the short range which are likely to be detected by the late fusion component. However, the hybrid fusion underperforms early fusion due to the loss in the omitted points. It should be noted that the actual required link capacity from a sensor node to the central fusion system will depend on the processing frame rates. For example, using early fusion in the proposed T-junction scenario with a processing frame rate of 10 frames per second, common for lidars, will require a communication link

with the capacity of 5.16 Mbps (516 kb per frame times 10 frames per second) from each infrastructure sensor to the central fusion system. Such rates can be easily supported by the commercial wired as well as wireless Local Area Network (LAN) technologies that may be needed to implement the proposed system model in Figure 5.1. Adopting point cloud compression mechanisms could also provide effective means of reducing the communication bandwidth, although such methods are out of the scope of this study. Although network delay is not considered in this study, the insight is that it could constitute a significant problem to the fusion system by preventing successful detections due to missing frames or by generating false positives due temporal misalignment of incoming frames. The likelihood of such miss detections depends on the communication channel properties and should be rigorously investigated in future studies.

The computation time required to compute each frame increases in early fusion when compared to late fusion because the fused point cloud has more points than the individual point clouds, which increases the detection model inference times. However, these measurements were conducted assuming that sensor nodes (processing late fusion) have the same computational capabilities of the central fusion system. In practice this is not a valid assumption, since sensor nodes are low-end devices, while the central fusion system is expected to have high-performing GPUs. As a result, in practice, it is expected that late fusion running on sensor nodes is significantly slower than early fusion running on the central fusion system. Note that the computation times are hardware dependent and in this case were obtained using a Nvidia Quadro M4000 GPU for all experiments and fusion schemes.

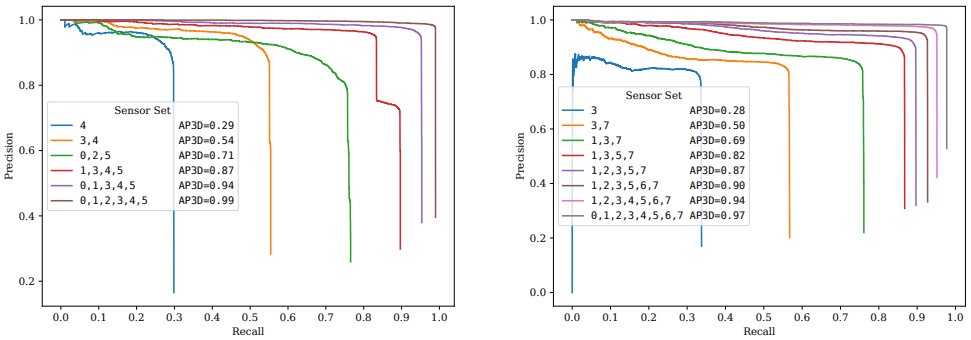
5.4.3 Impact of Sensors Pose and Number on Detection Performance

This experiment focuses on the evaluation of the impact of the pose (position and orientation) and number of sensors on the object detection performance. The performance is evaluated for early and late fusion schemes on both scenarios considering all objects within the detection area, defined in Section 5.2. The evaluation is carried out for all possible sensor sets, where the number of sensors in a set ranges from one to six in the T-junction and one to eight in the roundabout scenario. The NMS algorithm and threshold are the same as the previous experiment for consistency of the results.

Table 5.2 reports the top-3 performing sensor sets for each number of sensors in both scenarios in terms of AP_{3D} metric ($\kappa = 0.7$). The results show that the detection performance increases as the number of engaged sensors is increased. In particular, there is a steep performance increase of more than

Table 5.2: Detection Performance of Early Fusion (EF) and Late Fusion (LF) for Various Sensor Combinations

	T-junction			Roundabout		
No. Sensors	**Sensor Set**	**EF AP_{3D}**	**LF AP_{3D}**	**Sensor Set**	**EF AP_{3D}**	**LF AP_{3D}**
8	-	-	-	0,1,2,3,4,5,6,7	**0.9670**	**0.8143**
7	-	-	-	0,1,2,3,5,6,7	**0.9385**	**0.7904**
	-	-	-	1,2,3,4,5,6,7	0.9340	0.7868
	-	-	-	0,1,3,4,5,6,7	0.9307	0.7834
6	0,1,2,3,4,5	**0.9870**	**0.8181**	1,2,3,5,6,7	**0.9050**	0.7627
	-	-	-	1,2,3,4,5,7	0.9017	0.7627
	-	-	-	0,1,2,3,5,7	0.8973	**0.7664**
5	0,1,3,4,5	**0.9441**	0.7611	1,2,3,5,7	**0.8678**	**0.7385**
	0,1,2,3,5	0.9348	0.6672	1,3,5,6,7	0.8610	0.7314
	1,2,3,4,5	0.9327	**0.7914**	1,3,4,5,7	0.8576	0.7313
4	1,3,4,5	**0.8653**	**0.7336**	1,3,5,7	**0.8231**	**0.7069**
	0,1,2,5	0.8596	0.4765	1,2,3,7	0.7356	0.6557
	0,1,3,4	0.8394	0.6827	1,3,4,7	0.7263	0.6482
3	0,2,5	**0.7123**	0.4039	1,3,7	**0.6892**	**0.6230**
	2,3,5	0.6938	0.5834	3,5,7	0.6713	0.5831
	3,4,5	0.6837	**0.6656**	1,3,5	0.6382	0.5861
2	3,4	**0.5365**	**0.5361**	3,7	**0.5016**	0.4594
	2,3	0.4591	0.4530	1,3	0.4995	**0.4998**
	2,5	0.4453	0.3309	5,7	0.4664	0.4638
1	4	**0.2862**	**0.2862**	3	**0.2811**	**0.2811**
	3	0.2512	0.2512	5	0.2596	0.2596
	2	0.2013	0.2013	1	0.2151	0.2151



(a) T-junction (b) Roundabout

Figure 5.5: Precision-Recall curves of early fusion with different number of sensors for a T-junction and b roundabout scenarios. The curves are produced for the sensor sets highlighted in bold in Table 5.2. AP_{3D} values are calculated for $\kappa = 0.7$.

50% when using two sensors instead of one in both scenarios. However, the performance gain saturates as the number of sensors increases. Also, it can be seen that as the detection area increases, more sensors are needed to maintain the detection performance. For example, in the roundabout scenario, eight sensors need to be engaged to achieve a performance level equal to that of six engaged sensors in the T-junction scenario, which has smaller detection area. Furthermore, the early fusion scheme consistently outperforms late fusion with respect to the detection performance. Their disparity becomes more significant as the number of sensors grow since the early fusion scheme can exploit more information at detection time compared to late fusion.

Figure 5.5 presents the PR curves of the best sensor sets (rows with a bold font in Table 5.2) for both scenarios using the early fusion scheme. The curves show that the maximum recall of detected objects increases significantly for both scenarios when the number of engaged sensors increase. Specifically, a single sensor can detect only 30% of all the vehicles in the T-junction scenario and slightly more than 30% of all the vehicles in the roundabout scenario. However, when all sensors (six for the T-junction and eight for the roundabout) are engaged, both scenarios show similar performance and detect more than 95% of the ground-truth objects with precision above 95%.

As one could anticipate, the results show that the number and pose of sensors have direct impact on the performance of the system. The results also demonstrate the impact of spatial diversity in terms of the improved quality of the input information to the object detection model. For example, the sensor set (0,2,5) achieve the best performance for three sensors in the T-junction scenario. Adding sensor 1 to the aforementioned set does not increase the field-of-view of the system, as observed in Figure 5.4a, however the results in Table 5.2 show that this addition has a notable impact on detection performance (20% increase in AP_{3D}). The impact of spatial diversity on detection performance is further explored in the next experiment.

5.4.4 Spatial Diversity Gain of Cooperative Perception

The enhanced detection performance in cooperative perception seen in the previous experiments can be associated with two factors: 1) the increased field-of-view; 2) the spatial diversity gain, which manifests itself in point clouds with higher point density in areas that are covered by multiple sensors. This experiment intends to shed light into the latter factor.

This experiment focus on objects within a defined Region of Interest (ROI), where all objects are within the field-of-view of two specific sensors. For example, the ROI for the sensor set (2,4) in the T-junction scenario is limited to road to the left of the junction (filled with green and brown points in Figure

Table 5.3: Impact of Spatial Diversity on Detection Performance

T-junction			Roundabout		
ROI	**Sensors Set**	**AP_{3D}**	**ROI**	**Sensors Set**	**AP_{3D}**
	1	0.4717		1	0.1474
1,5	5	0.3222	1,7	7	0.8874
	1,5	0.8722		1,7	0.8925
	2	0.5621		3	0.3944
2,4	4	0.7942	3,5	5	0.8819
	2,4	0.9560		3,5	0.8996

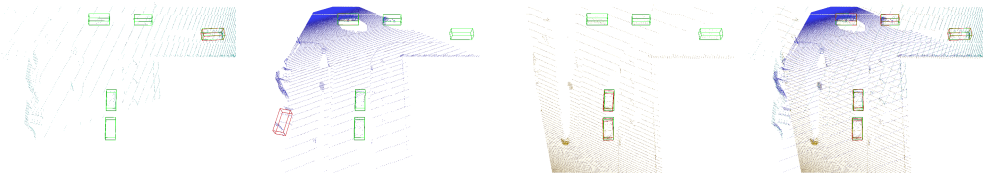
5.4b), and the ROI for the sensor set (3,5) for the roundabout is limited to the upper right quadrant of the roundabout (filled with yellow and light purple points in Figure 5.4d). For each of the these ROIs, the detection performance of the early fusion scheme using the specified sensor sets is compared to that of a single sensor covering the same ROI. The detection performance is quantified by the AP_{3D} metric restricted to objects within the specified ROI. For each scenario, the two sensors sets with highest field-of-view overlap are considered: (1,5) and (2,4) for the T-junction scenario and (1,7),(3,5) for the roundabout scenario.

The impact of spatial diversity on the detection performance of early fusion scheme is visualised in Figure 5.6. As it can be seen in the snapshots in Fig 5.6c, when a single sensor is engaged most objects fail to be detected or are detected with poor accuracy (*i.e.* incorrect size or yaw angle). However, upon increasing the point density by combining multiple point clouds with early fusion, it is possible reduce the number of false negatives and increase the quality of estimated bounding boxes, as illustrated in Figure 5.6d.

The results of this experiment, summarised in Table 5.3, show that the early fusion scheme using only two sensors outperforms the best single sensor by 20% and 85% in the T-junction scenario. However, the detection performance gain when using two sensors in the roundabout is marginal. This is due to the fact that the fields-of-view of the specific sensor set used has minimal overlap in this particular roundabout scenario. The results presented indicate that early fusion can: a) reduce the number of false negatives caused by occlusion and low point density; b) improve the quality of estimated boxes when the sensors have significant overlapping coverage.

5.4.5 Impact of Point Density on Estimated Bounding Boxes Accuracy

One can intuitively stipulate that a denser point cloud will provide additional information about the objects in the scene, as has been the case for Airborne



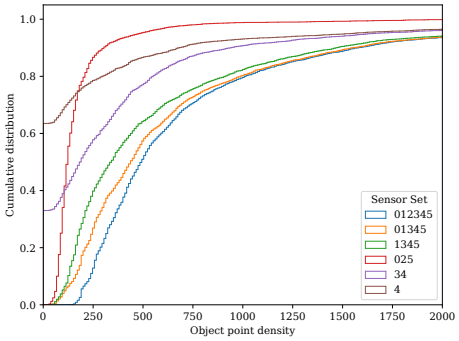
(a) From Sensor 0 (b) From Sensor 1 (c) From Sensor 5 (d) Early Fusion from sensor set (0,1,5)

Figure 5.6: Illustration of the impact of spatial diversity on the performance of the early fusion scheme for various sensor configurations (green boxes represent the ground-truth objects and red boxes represent estimated objects). Points' colour indicate the sensor of origin.

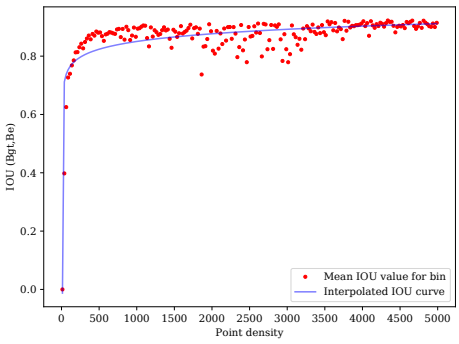
lidar scanners and object classification in the context of remote sensing [187]. This experiment analyses how the number of sensors affects the density of points in the point cloud and, in turn, the accuracy of the boxes estimated by the object detection model in a driving context. The point density of an object is defined as the number of points within the boundaries of its ground-truth bounding box. This point density density is a discrete random variable that is a random function of the number and pose of the sensors that observe the object.

Figure 5.7a shows the Cumulative Distribution Function (CDF) of objects' point density for the best sensor sets in the T-junction scenario from Table 5.2 (highlighted in bold font). Given a point $(d, F(d))$ where F represents one of the CDF curves, the vertical coordinate $F(d)$ represents the ratio of objects whose point density is smaller or equal to the horizontal coordinate d. The intersections with the vertical axis shows that using Sensors 4 and the sensor set (3,4) alone results in more than 60% and 30% of the objects having zero point density, respectively. Similarly, one can compute the ratio of objects whose point density is within an interval $[d_1, d_2]$ by computing $F(d_2) - F(d_1)$. Thus, using the sensor set (0,2,5) guarantees that all ground-truth objects have non-zero point density but results in 90% of the objects having point density below 300 points. When the number of engaged sensors is increased, the number of objects with point density in the range of $[250, 1000]$ points increases significantly, but saturates for point densities above 1750 points.

Next, the relationship between an object's point density and the accuracy of the estimated bounding box, measured by the IOU metric, is investigated. For this purpose,the objects are split into 200 uniform-sized bins according to their point density. The IOU value for a bin is computed by averaging the individual IOU values among all objects in the bin. Figure 5.7b shows the scatter plot of the IOU value per point density bin and a log curve interpolation. The accuracy of objects with point density below 70 points is poor, but increases



(a) CDF of objects point density for a varying number of sensors. An object's point density is the number of points within its ground-truth box. The curve slope represent the number of objects with a specific point density.



(b) IOU between ground-truth and detected boxes as the object point density varies. Each point represents the average IOU for a bin of objects. The number of bins used in the interval was 200.

Figure 5.7: Analysis of the point density and the IOU metric over estimated boxes in the T-junction test set.

significantly when the point density surpasses 100 points. The outliers observed in the range of $[1800, 3400]$ are caused by objects that are close to a sensor, thus have a high number of points concentrated on a small surface but few points elsewhere, resulting in a poorly estimated bounding box. In conclusion, the point density of an object can provide a useful prediction of the accuracy of the estimated bounding box. Thus, given the accuracy requirement for the estimated bounding boxes, it is possible to find the minimum required point density and the number of sensors required for a specific scenario.

5.4.6 Comparison with Existing Benchmarks

The direct comparison of the proposed fusion schemes with other approaches [75, 108] may not be meaningful due to its unique sensing strategy. However, for a fairer-comparison, two approaches are considered: (a) the AP_{3D} results obtained using a single sensor in this chapter are compared to the reported results produced by Voxelnet [75]; (b) the AP_{3D} results using two sensors in the T-junction scenario from Section 5.4.4 is compared to the reported results produced by Cooper [107] and F-Cooper [108] in a road intersection scenario. Both comparisons are reported in Table 5.4.

The first comparison considers the results produced by Voxelnet [75] using the KITTI dataset [47]. Their results are reported for AP_{3D} in three complexity categories, easy, moderate and hard, are 81.97%, 65.46% and 62.85%, respectively. The results from Section 5.4.3 shows that the proposed method using a single sensor achieve much lower AP_{3D}, around 28% for both scenarios. This significant performance gap emerges due to the evaluation in this

Table 5.4: Comparison with Existing Benchmarks

	$\mathbf{AP_{3D}}$	
	Single sensor	**Two sensors**
Cooper [107]	0.1960	0.7237
F-Cooper [108]	0.1960	0.7237
Proposed (early fusion)	0.4717	0.8722

study considering all the ground-truth objects within the detection area, while Voxelnet and other studies using the KITTI benchmark consider only the objects within the sensor's field-of-view. The performance gap highlights the complexity of detecting objects in both scenarios using a single sensor, since its field-of-view cannot cover all the detection area and is susceptible to occlusion caused by buildings and other objects. As discussed in Section 5.4.3, increasing the number of sensors used is highly beneficial to the detection performance in the proposed system.

Secondly, the proposed early fusion scheme results are compared with the results produced by F-Cooper [108]. For a fair comparison, the proposed system in this study uses only two engaged sensors in the T-junction scenario, which is similar to the "road intersections" scenario reported in [108]. In [108], the authors report results in two categories, "near and far", according to the distance from the object to the sensor. The "near" category shows marginal improvement, hence the comparison focus on the "far" category. The AP_{3D} results in [108] for a single sensor and fusion of two sensors are 19.60% and 72.37%, respectively. The results of the proposed method in this chapter under a similar scenario for a single sensor and early fusion of two sensors are 47.17% and 87.22%, respectively, as noted in Section 5.4.4. Although the direct comparison of these values is not meaningful given the dataset differences and sensing strategy, it is possible to see that both approaches show a comparable performance gain when considering more than a single sensor.

5.5 Summary

This chapter proposed a cooperative perception system for 3D object detection using three fusion schemes: early, late, and hybrid fusion. The proposed system model contains n infrastructure sensors that share data with a central fusion system, where information is fused and the resulting detections (3D bounding boxes) are disseminated to all the vehicles in the vicinity. A novel cooperative dataset containing depth maps from multiple infrastructure sensors in a T-junction and a roundabout scenario was used for the evaluation of the proposed system. The evaluation indicated that increasing the number of sensors in the

proposed system is highly beneficial in complex scenarios, which allowed to overcome occlusions and restricted field-of-view. The results showed that the early fusion scheme outperformed other fusion methods at the cost of higher communication bandwidth, while the hybrid fusion compromised detection performance and communication costs. Furthermore, the proposed system was able to increase the perception horizon and the density of the fused point cloud by exploiting spatially diverse observations with overlapping fields-of-view, which reduced false negative detections and allowed more accurate estimates of the 3D bounding boxes. Finally, the results suggested that the system can be realised with current communications technologies and can reduce the costs of individual vehicles through shared infrastructure resources.

Chapter 6

Infrastructure Sensor Pose Optimisation for Robust Cooperative Perception

Chapter 5 motivated the usage of infrastructure sensors for cooperative 3D object detection. The experiments in that chapter showed that fusing the data from multiple sensors allowed to improve the detection recall and precision metrics. Furthermore, the experiments showed a correlation between the number of points observed on the surface of an object and the accuracy of its detected bounding-box. The pose of the sensors in both the T-junction and roundabout scenarios was empirically determined through domain-knowledge heuristics. Specifically, these heuristics included ensuring that the sensors cover the ground-area of all the areas of interest (blue rectangles in Figures 5.4a and 5.4c) and that some of the sensors had overlapping fields-of-view, which has been shown to improve detection performance in Section 5.4.4. Manually determining the pose of the sensors can be challenging in practice, particularly in cluttered scenarios where objects occlude one another. In such scenarios, maximising the visibility of ground area may not guarantee the visibility of objects placed over those areas due to the aforementioned occlusions. This chapter investigates how to automatically optimise the pose of infrastructure sensors such that the visibility of objects is maximised considering the effects of occlusions.

A major category of the existing studies formulate this problem as a discrete optimisation problem where a finite set of possible sensor poses is considered and the target objects' visibility is described by a set of binary variables [130–132, 134]. The problem is then solved by using various forms of Integer Programming (IP) solvers to either maximise the number of visible target objects (coverage) with a fixed number of sensors; or to minimise the number of sensors required to achieve a given coverage constraint. However, the majority

of the applications that may use such sensor networks, *e.g.* object detection and tracking, require a minimum level of visibility over the target objects which cannot be encoded by single binary variables. For example, an object may have different degrees of visibility due to occlusions and due to its position *w.r.t.* the sensors, which causes ambiguity in the assignment of a binary visibility variable. Another category of the existing studies consider the optimisation of continuous sensor pose variables using various forms of continuous optimisation algorithms [125–128]. Most of the studies in this category focus on maximising the coverage (visible ground area) of extensive 3D environments described by digital elevation maps. However, such formulation does not consider the distribution of objects in the environment, and instead, assume an object would be visible if it is within a region covered by the sensors. As a result, these studies fail to detect and prevent occlusions between objects since they do not explicitly model the visibility of the target objects. This becomes a limiting factor when considering cluttered environments such as traffic junctions with a significant number of vehicles and pedestrians.

In this chapter, an occlusion-aware visibility model is proposed based on a differentiable rendering framework and two novel approaches for object-centric sensor pose optimisation based on gradient-ascent and Integer Programming are proposed. Different from the existing approaches in the literature that aim to cover ground areas on elevation maps, this chapter considers explicitly modelling the visibility of the target objects by considering a set of object configurations, defined as frames. In this definition, each frame contains a number of target objects with specific sizes, positions and orientations within the environment. The objective of the proposed method is to maximise the visibility of all target objects across the frames. The distribution of frames is considered to be application specific and can be obtained by empirical evaluations or simulation. The proposed methods are comprehensively evaluated in a challenging traffic junction environment and compared with previous methods in the literature. The results of this evaluation indicate that explicitly modelling the visibility of objects is critical to mitigate occlusions in cluttered scenarios. Furthermore, the results show that both of the proposed methods outperform existing methods in the literature by a significant margin in terms of object visibility. In summary, the contributions of this chapter are:

- A realistic visibility model, created using a rendering process, that produces pixel-level visibility information and is capable of detecting occlusions between objects;
- A novel gradient-based sensor pose optimisation method based on the aforementioned visibility model;
- A novel IP sensor pose optimisation method that guarantees minimum

object visibility based on aforementioned rendering process;

- The performance comparison between both methods and existing works in the literature in a simulated traffic junction environment.

6.1 Problem Formulation

This section firstly presents the formulation of the sensor pose optimisation problem upon which the gradient-based and Integer Programming (IP) methods in Sections 6.2 and 6.3, respectively, are based. Next, a novel sensor pose parametrisation is introduced to constrain the sensor poses to feasible regions which are pre-defined according to the environment where the sensors are deployed.

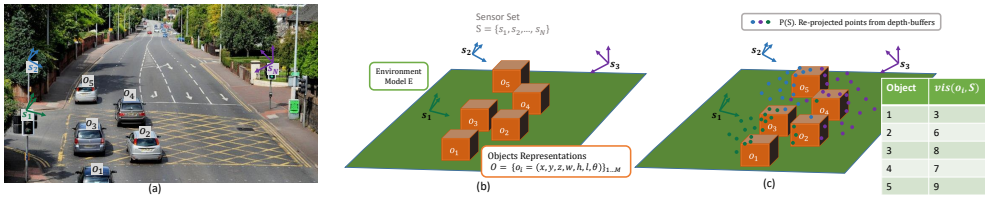


Figure 6.1: Illustration of the problem formulation for an exemplar driving environment with $N = 3$ sensors and $M = 5$ target objects. (a) Physical representation of target objects and sensor poses. (b) Objects and environment representation under the sensor pose problem formulation. (c) re-projected point cloud $P(S)$ and objects' visibility metric. Note that the visibility metric of a target object is obtained by counting the number of points of $P(S)$ on the surface the respective object, as defined in Equation 6.1.

The sensor network, depicted in Figure 6.1a, consists of a set of fixed infrastructure sensors S that collectively observe a set of target objects, denoted by O, in a driving environment. Each target object is represented using a three-dimensional cuboid encoded by $o = (x, y, z, w, h, l, \theta) \in O$, where x, y, z correspond to the 3D centroid of the box, while w, h, l represent the box size and θ corresponds to the pitch angle (rotation around the vertical axis), as depicted in Figure 6.1b. The visibility of object o by the sensor set S, denoted by $\text{vis}(o, S)$, is defined as the number of pixels, or points, that the set of sensors S project onto the object's surfaces. Visibility in this sense intuitively quantifies the information that sensors capture about each object and has shown to be correlated with the performance of perception tasks such as 3D object detection, as shown in Section 5.4.5, and tracking [188]. This visibility metric is computed in two steps. First, the frame containing objects O is rendered. Then, the depth-buffer from each sensor in S is re-projected into 3D space, creating an aggregated point cloud $P(S)$, as described in Section 6.2.2 and illustrated in Figure 6.1c. Finally, the visibility of each object $o \in O$ is

obtained by counting the number of points of $P(S)$ that lie on the surface of each respective object:

$$\text{vis}(o, S) = \sum_{p \in P(S)} \begin{cases} 1, & \text{if } p \text{ on } o\text{'s surface} \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

This visibility metric provides pixel-level resolution which successfully captures the effects of total or partial occlusions caused by other target objects and by the environment. The environment model, denoted by E, can also be modified according to the application requirements. For example, it is possible to include static scene objects, such as buildings, lamp posts and trees, that may affect the visibility of target objects.

The formulation proposed so far considered a single, static configuration of target objects, denoted by O. However, driving environments are dynamic and typically contain moving vehicles and pedestrians. Dynamic environments are accounted for by considering a set of L static frames. Each frame contains a number of target objects with specific sizes, positions and orientations within the environment. The number of frames, denoted by L, must be chosen such that the distribution of objects over the collection of frames approximates the distribution of target objects' in the application environment. For example, one can obtain a set of frames for driving environments using microscopic scale traffic simulation tools, such as SUMO [189] or through the empirical observation of the driving environment.

The underlying optimisation problem is to find the optimum poses for N sensors, denoted by $S = \{s_1, \ldots, s_N\}$ that maximise the visibility of target objects across the L frames. Formally, the optimal set of sensor poses is defined as

$$\hat{S} \triangleq \arg\max_{S} \min_{o \in \mathbb{O}} \text{vis}(o, S), \tag{6.2}$$

where $\mathbb{O}$ is the set of objects across L frames. In practice, each frame is rendered independently so that objects from different frames do not occlude one another, but the optimisation is still performed across all frames.

It should be noted that one can alternatively maximise the mean visibility of the target objects, which can be formulated as $\arg\max_S \frac{1}{M} \sum_{o \in \mathbb{O}} \text{vis}(o, S)$. However, this may result in some of the objects having very low or zero visibility in the favour of others having un-necessarily large visibility. But maximising the minimum visibility biases the optimisation algorithm towards sensor poses that guarantee the visibility of all target objects.

6.1.1 Sensor Pose Parametrisation

Generally speaking, the pose of a sensor in a 3D environment can be described by the canonical six degrees-of-freedom parametrisation $s = (x, y, z, \varphi, \theta, \phi)$, where the (x, y, z) represent the sensor position and (φ, θ, ϕ) its viewing angles. However, unconstrained optimisation under such parametrisation is seldom useful in practice as most environments have restrictions regarding sensors' location, *e.g.* sensors must be mounted close to a wall, on lamp posts, and clear from a road, etc. To this end, a continuous sensor pose parametrisation called virtual rail is proposed. This parametrisation imposes constraints over the sensors' location without adding any penalty term to the optimisation objective function or requiring any changes to the optimisation process, such as gradient projection.

A virtual rail is defined by a line segment between two points in 3D space. The sensors can be placed at any point within this line segment, as illustrated in Figure 6.5. The viewing angles are described by the rotations along the X and Y axis, as no rotation is assumed along the camera principal axis (Z). The pose of a sensor on a virtual rail between points $p_1, p_2 \in \mathbb{R}^3$ has its pose fully determined by the parameters $s = (t, \alpha, \beta)$ through the parametrisation

$$\begin{aligned} (x, y, z) &= p_1 + \sigma(t)(p_2 - p_1), \\ \varphi &= 2\pi\sigma(\alpha), \\ \theta &= \pi\sigma(\beta), \\ \phi &= 0, \end{aligned} \tag{6.3}$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{6.4}$$

is the sigmoid function. This function enforces the bounds of position within the rail, *i.e.* (x, y, z) on the line segment between p_1, p_2, and viewing angles $\varphi \in [0, 2\pi]$, $\theta \in [0, \pi]$ for unbounded variables $t, \alpha, \beta \in \mathbb{R}$.

This parametrisation allows the use of unbounded gradient optimisation with guaranteed constraints over the sensors' poses. Note that the choice of the number and position of virtual rails are hyper-parameters defined to fit the needs of the application according to the complexity of the environment/task.

6.2 Gradient-based Sensor Pose Optimisation

This section describes the proposed gradient-based sensor pose optimisation for multi-object visibility maximisation.

The objective function proposed in Equation 6.2 is not differentiable *w.r.t.* the sensor pose parameters due to the non-continuity introduced by the

threshold operation in vis($\cdot$). Thus, gradient-based solutions cannot be applied to solve this optimisation problem. Therefore, a processing pipeline featuring a differentiable objective function that approximates the objective function in Equation 6.2 is proposed. A crucial element of this approximation is the visibility score, a continuous variable in the interval $[0, 1]$ that measures the visibility of a given 3D point *w.r.t.* a sensor. The processing pipeline considers the continuous visibility score of multiple points over each target object, which ensures the objects' visibility and implicitly approximates the original problem in Section 6.1. It shall be noted that the visibility score is different from the visibility metric (Equation 6.1) in two ways: 1) the former is differentiable while the latter is not; 2) the former indicates the degree of visibility of a single point on a target object while the latter is the number of points on the surface of a target object. The proposed processing pipeline for the computation and optimisation of the objective function is depicted in Figure 6.2.

The processing pipeline consists of five stages.

1. a set of target points, denoted by $T \in \mathbb{R}^{MF\times 3}$, is created by sampling F points from each of the M target objects. The points are randomly distributed along the objects' surfaces proportionally to each surface area.

2. the points T are projected onto the image plane of each sensor and a visibility score is computed for each target point according to their position *w.r.t.* the visible frustum of the respective sensor, as described in Section 6.2.1.

3. an occlusion-aware visibility model, described in Section 6.2.2, is used to update the visibility score created in the previous stage. This is required since some projected points will be in the visible frustum of a given sensor but occluding objects prevent direct line-of-sight between the point and the sensor.

4. the objective function is computed as the mean visibility score of all points T on target objects, as described in Section 6.2.3.

5. gradient-ascent is used to maximise the objective computed in the previous step, as described in Section 6.2.4.

The proposed processing pipeline can work for any continuous sensor pose parametrisation. However, this chapter considers the parametrisation proposed in Section 6.1.1, which constrains the sensor position to a line segment and allows for unconstrained gradient-based optimisation.

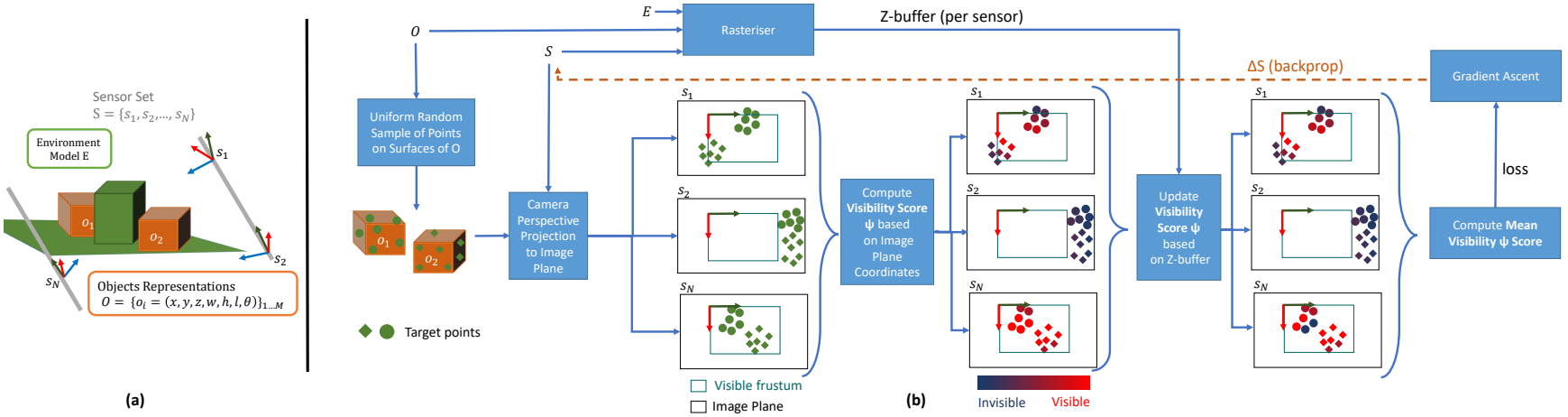


Figure 6.2: Processing pipeline of the proposed Gradient-based sensor pose optimisation method. (a) an exemplar frame with two objects and a set S of N sensors, including an environmental model with an occluding block (in green). (b) the optimisation pipeline.

6.2.1 Visibility Model

In this section, a realistic visibility model is proposed based on the perspective camera model provided by PyTorch3D [167]. Built on top of PyTorch [166], this camera model provides differentiable transformations from the global coordinate system to the camera image plane which is fundamental for a fully differentiable pipeline. The cameras' extrinsic matrix is determined by the pose of the sensors, specified by the set of parameters S, being optimised. It shall be noted that all cameras intrinsic properties are identical: 90-degree horizontal field-of-view, $D_{\text{near}} = 1\text{m}$, $D_{\text{far}} = 100\text{m}$ near and far clipping planes, respectively, and resolution of $W = 200$ x $H = 200$ pixels. The resolution is kept relatively small in order to reduce the computational complexity of the rendering process, described in Section 6.2.2. Increasing the image resolution directly increases the visibility of the target objects as there will be a higher number of pixels/points per object. In practice, the sensor poses resulting from the optimisation process can be used for cameras with higher resolution, as long as they have the same aspect ratio and field-of-view.

The projection of a point $p = [x\ y\ z]^T \in \mathbb{R}^3$ in the global coordinate system into the image plane of sensor s is given by

$$\begin{bmatrix} u_s d_s \\ v_s d_s \\ d_s \end{bmatrix} = M_i M_e(s) \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{6.5}$$

where $[ud\ vd\ d]^T$ are homogeneous coordinates that can be divided by d to obtain the canonical form $[u\ v\ 1]^T$. Here, u, v are the image plane coordinates in pixels, d is the depth of the point in the view frustum and M_i, M_e are the intrinsic and extrinsic camera matrices of sensor s, respectively. The point p is within the visible frustum if and only if $W \geq u \geq 0$, $H \geq v \geq 0$ and $D_{\text{far}} \geq d \geq D_{\text{near}}$ where W and H are the image width and height in pixels. $D_{\text{near}}, D_{\text{far}}$ are the camera near and far clipping planes in meters, respectively.

The visibility of a given point from the perspective of a sensor s is determined by verifying that the image plane projection of this point, given by Equation 6.5, satisfies the bounds described in the previous paragraph. If the bounds are satisfied, the point is considered visible, otherwise it is not. Since the threshold operations used to identify the visibility of a point are not differentiable, the sigmoid function (Equation 6.4) is chosen as a differentiable approximation of the binary visibility. This continuous visibility score can be interpreted as a probabilistic visibility measure [125] of a point, ranging from 0 (completely invisible) to 1 (completely visible). This is formulated by a *window* function as follows:

$$w(z, \gamma, z_0, z_1) = \sigma(\gamma(z - z_0)) - \sigma(\gamma(z - z_1)), \tag{6.6}$$

where $\gamma \in \mathbb{R}$ controls the rate of transition on the limits of the interval $[z_0, z_1]$, as illustrated in Figure 6.3. As γ increases the window function tends to a binary threshold operation. However, this reduces the intervals with non-zero gradients, and consequently inhibits parameters updates through gradient optimisation. Empirical tests revealed that $\gamma = 1$ was the best out of the three tested values (0.1, 1, 10) for this hyper-parameter.

The visibility score of a point p with image plane projection $[u_s d_s \; v_s d_s \; d_s]^T$ is given by

$$\Psi(p, s) = w(u_s, \gamma, 0, W) \cdot w(v_s, \gamma, 0, H) \cdot w(d_s, \gamma, D_{\text{near}}, D_{\text{far}}). \tag{6.7}$$

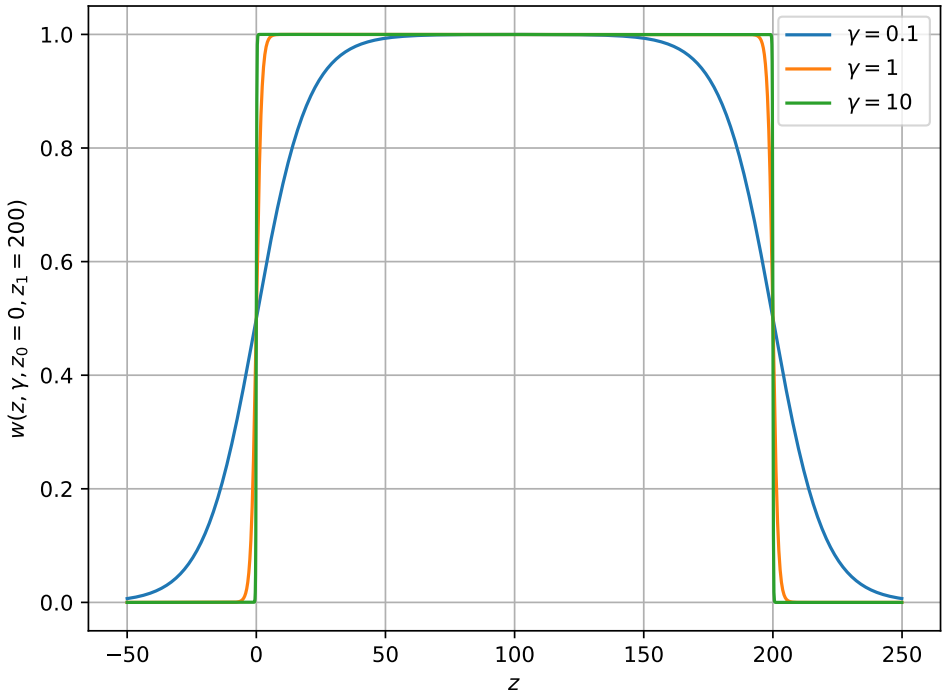


Figure 6.3: Window function $w(z, \gamma, z_0, z_1)$ plotted for $z_0 = 0, z_1 = 200$ and varying values of γ.

This visibility model does not take into account occlusions caused by other objects or the environment since a point being within the visible frustum of a sensor is a required but not sufficient condition to guarantee direct line-of-sight visibility from the sensor to the point. To account for occlusions, an occlusion aware visibility model, based on the visibility model presented in this section, is presented in Section 6.2.2.

6.2.2 Occlusion Awareness

Determining the visibility/occlusion of objects is a common problem in computer graphics, and appears in different applications, *e.g.* when casting shadows [190] in a scene. The process of casting shadows in a scene is equivalent to determine the occlusions from the light source perspective [190], and rendering those points as unlit in the camera perspective view. The same approach can used to identify if a given sensor has line-of-sight visibility of a point in 3D space in the proposed occlusion-aware visibility model. However, instead of considering the light source perspective as in [190], the proposed model considers the sensors' perspective. The line-of-sight visibility is verified using the depth buffer generated by the PyTorch3D's [167] rasteriser. This rasteriser transforms the meshes representing the environment and the target objects into a raster image with a corresponding depth buffer. When an object is projected to the image plane, the orthogonal distance between the object and the sensor is stored in the corresponding pixel position of the depth buffer. If another object is projected to the same pixel, the depth buffer keeps the smallest depth distance among the two. By comparing the value of the depth buffer at the projection of a given 3D point with its distance to the camera, it is possible to determine if the point is visible (*i.e.* the depth values match), or if the point is occluded (the depth buffer has a smaller depth value, indicating there is another object closer to the camera in that direction).

Given a target point $p \in T$ and a sensor s, the point is considered to be occluded from the point of view of sensor s if

$$|d_s - Z_s(u_s, v_s)| > \kappa, \tag{6.8}$$

where $[u_s d_s \; v_s d_s \; d_s]^T$ is the projection of p on the image plane of s according to Equation 6.5. Here, $Z_s(u_s, v_s)$ is the depth buffer of sensor s at the pixel position (u_s, v_s) and κ is a threshold for the maximum disparity between the projection depth value and the depth buffer. The experiments of this chapter consider $\kappa = 0.5$m, which allows to accurately detect occlusions. Figure 6.4 illustrates this occlusion-aware visibility model for a visible and an occluded point. In this figure, the depth of a point projected on the image plane matches the depth buffer measurement at the corresponding pixel if the point is visible; if the point is occluded the depth buffer value will be smaller since there is another object closer to the sensor.

This occlusion-aware visibility model leads to an enhanced version of the

visibility score of a point p observed by sensor s, given by:

$$\Psi(p,s) = \begin{cases} w(u_s,\gamma,0,W)\cdot \\ \quad w(v_s,\gamma,0,H)\cdot \\ \quad\quad w(d_s,\gamma,D_{\text{near}},D_{\text{far}}) & \text{, if } |d_s - Z_s(u_s,v_s)| \leq \kappa \\ 0 & \text{, otherwise.} \end{cases} \tag{6.9}$$

In the case where p is out of the visible frustum of sensor s, the visibility score is given by Equation 6.7. Note that if the point is occluded, there is no gradient signal to change the pose of the sensor in which the point is occluded. Yet, the occluded point can be targeted by other sensors in the network.

The depth buffer from each sensor is re-projected into 3D space, using the inverse of Equation 6.5, to create a 3D point cloud representing all points observed by the respective sensor. Effectively, the depth buffers from each sensor $s \in S$ are re-projected and aggregated into a fused point cloud $P(S)$, shown in Figure 6.7b. This fused point cloud is used to compute the visibility metric $\text{vis}(o,S)$, used by the IP method and during the system performance evaluation.

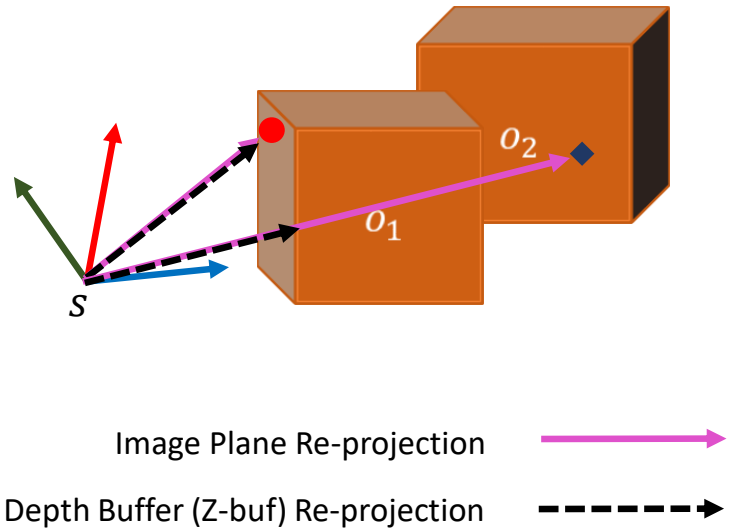


Figure 6.4: Illustration of the occlusion-aware visibility model: a point is considered to be visible by sensor s if it lies within the visible frustum of s and the Z component of the projection in the image plane closely matches the Z component obtained from the depth buffer (red point on o_1). If the disparity between these distances is above a threshold ($\kappa = 0.5$m) the point is considered occluded (blue point on o_2).

6.2.3 Objective Function

A target point p may be observed by multiple sensors, thus, the overall visibility of a point by a set of sensors S is computed as

$$\Psi(p, S) = 1 - \prod_{s \in S} (1 - \Psi(p, s)). \tag{6.10}$$

According to Equation 6.10, a point's overall visibility score is forced to be 1 if at least one sensor has a visibility score of one. Conversely, sensors that cannot observe a point (zero visibility score) do not affect the overall visibility score. Furthermore, when multiple sensors observe the same point, the combined visibility score improves.

The proposed sensor pose optimisation model in this chapter aims to maximise the mean visibility score across all objects O for a given set of sensors S. Hence, the following objective function is maximised in the proposed gradient-based formulation:

$$\mathcal{L} = \frac{1}{|T|} \sum_{p \in T} \Psi(p, S), \tag{6.11}$$

where T is a set of randomly sampled target points from target objects' surfaces, and $\Psi(p, S)$ is the overall visibility score of point p across all sensors S according to Equation 6.10 considering the enhanced occlusion-aware visibility model, described by Equation 6.9.

6.2.4 Optimisation

The Adam optimiser [191] is used to allow per-parameter learning rate and adaptive gradient-scaling, which has been shown to stabilise and shorten the optimisation process. The optimiser uses a global learning rate of 0.1, and is executed for 20 iterations over the whole collection of frames. These optimisation hyper-parameters were determined empirically through experiments. Algorithm 1 describes the optimisation process for a set of frames and Table 6.1 specifies the input variables used in the algorithm.

The objective function in Equation 6.11 is maximised *w.r.t.* the continuous sensor pose parameters (t, α, β) described in Section 6.1.1. These parameters specify the pose of a sensor within a virtual rail. In an environment containing multiple virtual-rails, there must be an assignment between each sensor and the virtual-rail it belongs to. This assignment is represented by a discrete variable that maps each sensor to one of the virtual-rails and is also subject to optimisation. However, since it is a discrete variable, it cannot be part of the gradient-based optimisation process. To overcome this problem, multiple runs of the optimisation process are performed, each with a random virtual-rail

Table 6.1: Description of Variables in Algorithm 1

Variable	Description	Value
N	Number of sensors	1-6
$O_1, \ldots, O_L$	Sets of objects for each of the L frames	
L	Number of frames in the dataset	1000
F	Number of target points sampled per object	400
E	Environmental model	
virtualRails	The set of virtual rails described by two end-points in $\mathbb{R}^3$	
epochs	Number of optimisation iterations	20

assignment. The results are reported using the best sensor poses across all runs in terms of the objective function.

The sensor poses are initialised using a uniform distribution on the interval $[-2, 2]$ over the parameter t, which controls the sensor position (x, y, z) along the virtual-rail according to Equation 6.3. The limits of the uniform distribution are chosen such that the sensors initial position within the rail can be anywhere from 10% to 90% of the length of the rail. The viewing angles can be randomly initialised in the same fashion. However, there may be some prior information of the environment that can guide this decision. For example, in a traffic junction objects are likely to traverse the central area of the junction, thus, sensors could benefit by focusing towards the junction centre. Although this step is not strictly required, it introduces prior information into the problem which reduces the amount of time required to achieve satisfactory results in the optimisation process.

6.3 Integer Programming-based Sensor Pose Optimisation

Integer Programming (IP) is an effective approach for solving optimisation problems where some or all of the variables are integers and may be subject to other constraints [192]. Applied to sensor pose optimisation, this formulation assumes that the optimal set of sensors are chosen from a finite set of sensor poses, called candidate poses. The problem is a combinatorial search to find the optimal subset of candidate poses that maximise an objective function. This objective function typically models the visibility of an area or objects. Additional constraints, such as the maximum number of sensors in the optimal set can be added to the problem formulation. This section describes how IP can be applied to solve the sensor pose optimisation problem formulated in Section 6.1. The objective is to find the subset of candidate sensor poses that maximises the minimum visibility metric of target objects. First, a method for the discretisation of the sensor pose parameter space into a finite set of

Algorithm 1 Gradient-Ascent Sensor Pose Optimisation

Input: $N, O_1, O_2, O_3, \ldots, O_L, F, E$, virtualRails, epochs
Output: $\hat{S}$, minVisibility

Initialisation :

$S \leftarrow \emptyset$
for $i \leftarrow 1$ to N **do**
  $p_1, p_2 \leftarrow$ random virtual rail from virtualRails
  Draw sample t from Uniform$(-2, 2)$
  Set α, β such that sensor focus on the centre of the junction {Alternatively, sample them from the uniform distribution.}
  Set $s = f(p_1, p_2, t, \alpha, \beta)$ {f is the sensor pose parametrisation given by Equation 6.3}
  $S \leftarrow S \cup s$
end for

Optimisation loop

for $e \leftarrow 1$ to epochs **do**
  $\mathcal{L} \leftarrow 0$
  for $O \in \{O_1, \ldots, O_L\}$ **do**
    $T \leftarrow$ sample F points from each target objects $o \in O$ surfaces
    $T' \leftarrow$ image plane projection of $p \in T$ for each sensor $s \in S$ according to Equation 6.5
    $Z, P \leftarrow$ depth-buffer and reconstructed point-cloud from rasteriser as a function of O, S, E
    $\Psi \leftarrow$ visibility score for each $p \in T'$ according to Equation 6.9
    $\Psi_S \leftarrow$ overall visibility score over all sensors according to Equation 6.10
    $\mathcal{L} \leftarrow \mathcal{L} + \text{mean}(\Psi_S)$
  end for
  minVisibilityMetric $\leftarrow \min_o \text{vis}(o, S) \forall o \in O_1 \cup \cdots \cup O_L$ {Computes the visibility metric using the reconstructed point-cloud P}
  if minVisibilityMetric improved since last epoch **then**
    $\hat{S} \leftarrow S$
  end if
  Compute $\frac{\partial \mathcal{L}}{\partial S}$ using automatic differentiation
  Update S based on gradient-ascent update rule
end for
return $\hat{S}$, minVisibility

candidate poses is introduced. Next, the base optimisation problem in Eq. 6.2 is cast into an IP optimisation problem and three approaches are devised to solve it: a heuristic off-the-shelf solver and two approximate methods based on sampling strategies.

6.3.1 Discretising Pose Parameters

To apply Integer Programming to the sensor placement problem, one needs to discretise the sensor pose parameter space into a finite set of candidate sensor poses. The choice of discretisation steps is kept to a minimum to minimise the computational cost of computing the visibility of all objects across all frames for each considered candidate pose. To this end, the concept of virtual rails, described in Section 6.1.1, is used to create a set of candidate sensor poses by dividing each virtual rail into 10 equally spaced sensor positions. The horizontal viewing angles (yaw) at each position is also divided into 10 feasible angles, between 0 and 360 degrees, covering all horizontal directions. The vertical viewing angles (pitch) at each position is divided into three feasible angles, defining low, medium and high inclination towards the ground. The rotation around the camera axis (roll) is not relevant in this application and is fixed at zero degrees, as discussed in Section 6.1.1. As a result, the set of candidate sensor poses for a given virtual rail is $S' = \{(t, \varphi, \theta) : \sigma(t) \in \{0.1, 0.2, \ldots, 1\}, \varphi \in \{36, 72, \ldots, 360\}, \theta \in \{18, 36, 54\}\}$, containing 300 candidate poses. For simplicity, in the rest of this chapter assume that S' represents the union of candidate poses from all virtual rails and the number of candidate poses is given by $|S'| = N'$. Figure 6.5 illustrates the set of candidate poses S' for a T-junction scenario, where a single horizontal and vertical viewing angles are shown per sensor position for visualisation purposes.

6.3.2 IP Objective

The general sensor pose optimisation problem can be formulated as the following IP problem

$$\begin{aligned} &\max_{b_1,\ldots,b_{N'}} f(b_1, \ldots, b_{N'}, o_1, \ldots, o_M) \\ &\text{s.t.} \quad \sum_{i=1}^{N'} b_i \leq N, \end{aligned} \tag{6.12}$$

where b_i is a binary variable indicating if the i-th sensor in the candidate set, denoted by $s_i \in S'$, is part of the optimal set. In other words, the sensor s_i is part of the optimal set if b_i is 1 and the optimal set of sensors is given by $\hat{S} = \{s_i \in S' : b_i = 1 \quad \forall i \in \{1, \ldots, N'\}\}$. The constraint guarantees that the maximum number of chosen sensors do not exceed N. The objective function

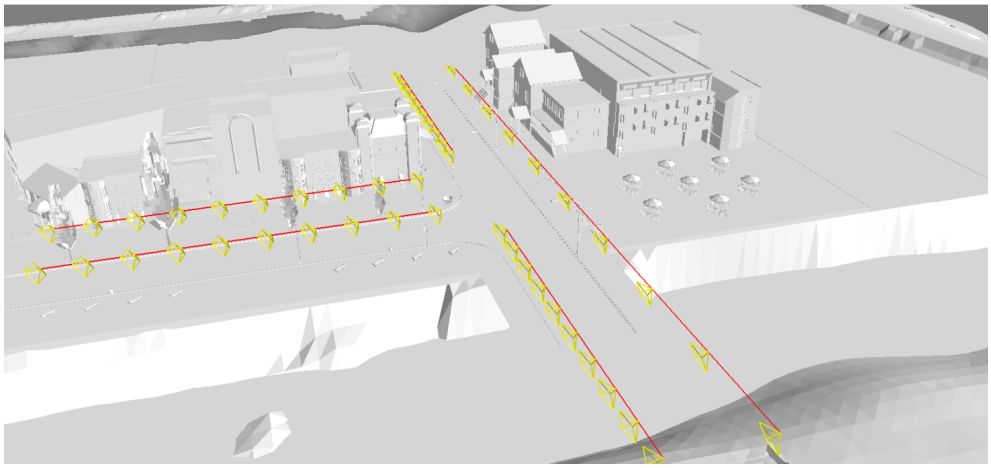


Figure 6.5: Candidate set S' over 5 virtual rails (red line segments) in a T-junction environment. Each yellow wireframe represent a sensor's viewing pose. To ease visualisation, only 10 candidates sensors are represented for each virtual rail, but the entire set consider 10 rotations along the Y axis and 3 rotations along the X axis, resulting in a total of 300 candidates per rail, or 1500 candidates overall.

$f(\cdot)$ represents the targets' visibility, which depends on the choice of sensors $b_1, \ldots, b_{N'}$ and the targets $o_1, \ldots, o_M$. Previous works [132, 135] define $f(\cdot)$ as the sum of binary visibilities of environment points. This is a poor estimate of target objects' visibility since there are varying degrees of visibility which cannot be encoded as a binary variable. To address this problem, a novel IP formulation that takes into account the visibility metric of a target object o observed by a sensor s, $\text{vis}(o, \{s\})$, defined in Equation 6.1, is proposed. The motivation is to find the sensor set that maximise the minimum visibility metric among target objects. Hence, the equivalent IP problem is described by

$$\begin{aligned} \max_{z, b_1, \ldots, b_{N'}} \quad & z \\ \text{s.t.} \quad & \sum_{i=1}^{N'} b_i \, \text{vis}(o, \{s_i\}) \geq z \quad \forall o \in O, \\ & \sum_{i=1}^{N'} b_i \leq N, \end{aligned} \tag{6.13}$$

where $z \in \mathbb{Z}_{\geq 0}$ is the minimum visibility metric among target objects. The first constraint guarantees that z is the minimum visibility metric among all objects. Note that the effect of multiple sensors observing a given object is cumulative *w.r.t.* the visibility metric, *i.e.* $\text{vis}(o, \{s_1, s_2\}) = \text{vis}(o, \{s_1\}) + \text{vis}(o, \{s_2\})$.

The formulation proposed so far considers a single frame, denoted by O, containing the target objects. This is extended to L frames by rendering each frame individually, including the objects and the environmental model, for

all candidate sensors. The visibility of an object o, as observed by sensor s_i, denoted by $\text{vis}(o, \{s_i\})$, is obtained by counting the number of points in the re-projected point cloud generated by sensor s_i that are on the surface of the object o, as described in Section 6.1 and illustrated in Figure 6.6. In practice, the visibility of all objects are computed frame by frame, for each candidate sensor, prior to the optimisation and stored in a visibility matrix V. This allows to solve the IP problem in Eq. 6.13 for any number of sensors without recomputing the objects' visibilities.

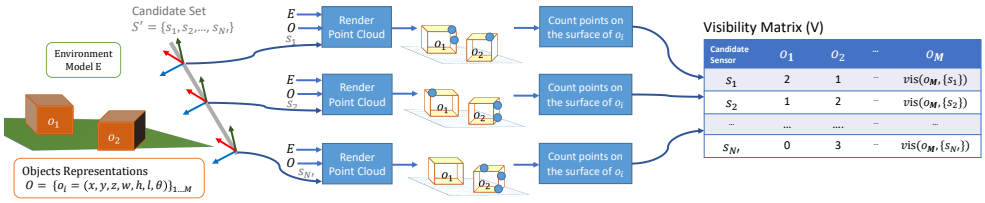


Figure 6.6: Illustration of the process of computing the visibility metric of object $o_i \in O$ by each candidate sensor $s_i \in S'$. The rendered point cloud naturally handles any occlusion caused by the environment model E and other target objects in the frame. The visibility of a given object is obtained by counting all points (represented by the blue dots) from the respective sensor that are on the surface of the respective object. The output is a visibility matrix V that depicts how many points each candidate sensor cast on each object in the frame, *i.e.* the object's visibility. This process is repeated for all frames and the matrices computed for each frame are concatenated horizontally.

6.3.3 Heuristic Solution

IP problems are NP-complete [192], thus, finding the solution using exhaustive search is computationally expensive or even unfeasible when the search space is large. Particularly, the size of the search space of the IP problem in Eq. 6.13 is $\binom{N'}{N}$. For example, for a candidate set with $N' = 1500$ poses and a given number of sensors N, *e.g.* 6, the size of the search space is $\binom{1500}{6} \approx 3^{17}$. For this reason, there are multiple algorithms that attempt to solve the problem using heuristic methods such as cutting plane and branch-and-cut methods [192].

In this chapter, the *Coin-or Branch and Cut (CBC)* open-source IP solver [193] and the *python-mip* wrapper [194] are used to solve the problem. This solver uses Linear Programming (LP) relaxation for continuous variables and applies branching and cutting plane methods where the integrality constraint does not hold. The solver cannot always guarantee the optimality of the solution, specially when exhaustive search is infeasible. Thus, the problem in Equation 6.13 is solved using the default optimisation settings until the optimal solution is found or the time since an improvement in the objective

function exceeds a limit.

6.3.4 Approximate Solutions

Approximate solutions to the IP problem are often used for the camera placement problem when exact solutions cannot be obtained in feasible time [135]. As described in the previous section, exhaustive search is unfeasible for the IP problem in Eq. 6.13 due to the size of the search space. For this reason, two approximate methods are implemented: Naïve sampling and Markov Chain Monte Carlo (MCMC) sampling.

The Naïve sampling method assumes that all sensors in the candidate set S' are equally likely to be part of the optimal set. This method explores the search space by uniformly sampling N sensors from the candidate set S' without replacement. The algorithm, described in Algorithm 2, runs until time since the last improvement in the objective function exceeds a limit.

The MCMC method uses the Metropolis-Hastings sampling algorithm [135] to select sensors that are likely to maximise the objective function. Algorithm 3 describes the full and detailed execution steps of the proposed sampling scheme. The process starts with an initial sample of N random sensors from S', denoted by S_0. At each subsequent iteration, a new sample set is computed as follows. At iteration i, a random and uniformly selected element of S_{i-1} is exchanged with a random and uniformly selected element of S', generating an intermediate set S_i^*. The ratio $r = \frac{f(S_i^*)}{f(S_{i-1})}$, is computed, where $f(S) = \min_{o \in O} \text{vis}(o, S)$. The solution set at iteration i is then set according to

$$Si = \begin{cases} S_{i-1}, & \text{if } u \leq r \\ S_i^*, & \text{otherwise,} \end{cases} \tag{6.14}$$

where u is a sample from the uniform distribution $U[0, 1]$. The algorithm is executed until the time since the last improvement in the objective function exceeds a limit.

6.4 Performance Evaluation

This section describes the evaluation of the proposed sensor pose optimisation methods. First, the evaluation metrics are defined in Section 6.4.1. Next, the experiment setup is described, including details of the simulation scenario in Section 6.4.2. Then, a comparative evaluation between the methods proposed in this chapter is presented in Section 6.4.3. Finally, a comparison of the proposed methods with existing works in the literature and a comparison of different visibility models are reported in Section 6.4.4 and 6.4.5, respectively.

Algorithm 2 Naïve Sampling Approximate IP Solution

Input: $S', N, O, \text{maxTime}$
Output: $\hat{S}$

Initialisation :
z_best = 0
$\hat{S} = \emptyset$
Sampling loop
while timeSinceLastImprovement $\leq$ maxTime **do**
  $S = N$ samples from S' without replacement;
  z = $\min_{o \in O} \text{vis}(o, S)$
  if (z $\geq$ z_best) **then**
    z_best = z
    $\hat{S} = S$
    reset timeSinceLastImprovement
  end if
end while
return $\hat{S}$

Algorithm 3 MCMC Metropolis-Hastings Sampling Approximate IP Solution

Input: $S', N, O, \text{maxTime}$
Output: $\hat{S}$

Initialisation :
z_best = 0
$\hat{S} = \emptyset$
$S = N$ samples from S'
MCMC loop
while timeSinceLastImprovement $\leq$ maxTime **do**
  $S^* = S$
  replace one random element of S^* with a random element from $S' \setminus S^*$
  $z = \min_{o \in O} \text{vis}(o, S)$
  $z^* = \min_{o \in O} \text{vis}(o, S^*)$
  $r = \frac{z^*}{z+\epsilon}$ {ϵ is a small value to avoid division by zero}
  $u \leftarrow$ sample from Uniform$(0, 1)$
  if $(u \leq r)$ **then**
    $S = S^*$
    $z = z^*$
    if (z $\geq$ z_best) **then**
      z_best = z
      $\hat{S} = S$
      reset timeSinceLastImprovement
    end if
  end if
end while
return $\hat{S}$

6.4.1 Evaluation Metrics

Existing studies in the literature assess sensor pose optimisation methods using the number of visible targets [135] or the mean ground area coverage [125, 127, 128], where coverage is defined as the probability that an area is visible to a sensor. However, such metrics are unsuitable for object-centric visibility for two reasons. First, adopting a binary visibility for an object is a coarse measure, since an object can be visible to different degrees due to its distance from the sensors, due to occlusions and limited sensor field-of-view. Secondly, the coverage of a ground area does not guarantee that an object placed within this area will be visible, as occlusions may limit the object's visibility. For the aforementioned reasons, a set of sensor poses S are evaluated based on the minimum visibility metric across all objects, denoted by $\min_{o \in \mathbb{O}} \text{vis}(o, S)$ in this study. Recalling from Equation 6.1, the visibility metric is defined as number of pixels that the set of sensors S observe on the surface of a given target object. In addition to the minimum visibility metric, the Empirical Cumulative Distribution Function (ECDF) of the visibility metric is computed for all objects across frames, which provides broader insight into the visibility patterns across objects.

6.4.2 Evaluation Setup

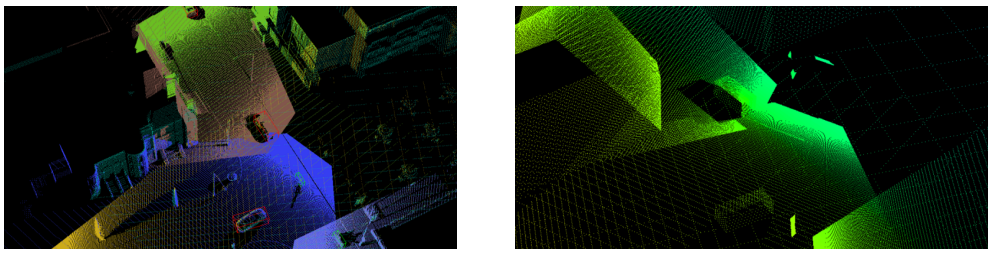
The performance evaluation of the proposed sensor pose optimisation methods is carried out by simulating traffic on a T-junction environment. This is motivated by the challenging conditions faced in such environments. For safety reasons, it is critical to guarantee that all vehicles, *i.e.* target objects, are visible to the sensors. Yet, vehicles are subject to occlusions from other vehicles and buildings.

The driving environment is simulated using the CARLA open-source simulator [165] under the same settings that produced the dataset in Section 5.2. A typical urban T-junction is chosen from one of the existing maps in the simulation tool. It has an area of 80 x 40 meters with several tall buildings and road-side objects, such as trees, bus shelters and lamp-posts. Within this environment, a dataset consisting of 1000 frames is generated. Each frame is a snapshot of the environment at a particular time, containing the number of vehicles and their representation. The objects' representation, as described in Section 6.1, defines their position, size and orientation in the environment.

The environment model, available through CARLA open-source assets, contains a high number of complex meshes that slow down the rendering process. For this reason, a simplified version of the environment is created, which allows for fast rendering and optimisation. To this end, cuboid meshes are created for the buildings near the junction, and represent vehicles as cuboids

using the same dimensions of the original objects' bounding boxes. This approximation significantly speed up the rendering process without detrimental impact to the measurement of objects' visibility metric. Figures 6.7a and 6.7b illustrate the original and simplified environment models, respectively.

Sensors placed in such driving environments must be placed by the road-side and clear from the road. This constraint is addressed by creating five virtual rails alongside the junction, each aligned with the curb over a segment of the junction, as illustrated in Figure 6.5. The parametrisation of the rails is application dependent and may need adjustment. In this application, the virtual rail configuration allows sensors to be positioned on existing road-side infrastructure, such as traffic lights. The virtual rails are positioned on a height of 5.2m above the ground, following the standards of public light infrastructure in the United Kingdom [181]. However, the height of each sensor could also be included in the optimisation process.



(a) (b)

Figure 6.7: T-junction environment models described by re-projected point clouds created using a the original environment model representation from CARLA and b the simplified environment model proposed in this chapter.

6.4.3 Comparative Evaluation of the Proposed Methods

Table 6.2 shows the results comparing the gradient-based, IP optimisation methods in terms of the minimum object visibility metric and the duration of the optimisation process for a varying number of sensors, denoted by N. Additionally, the results for the empirical sensor poses adopted in Chapter 5 are reported for comparison. Specifically, for each number of sensors N, the sensor set is chosen based on the best results from Table 5.2. The runtime performance of the IP methods does not include the time required to compute the visibility matrix, i.e. rendering 1000 frames for each of the 1500 candidate sensors poses, which took 28 hours. However, this process is only done once and the resulting visibility matrix is used by all IP methods for any number of sensors. None of the methods could find a pose for a single sensor that can observe all objects, thus, the results are reported for $N > 1$. The gradient-based method results are reported for the best out of 10 runs for each number of sensors. Each run

has a random sensor-rail assignment and random sensor position initialisation, as described in Section 6.2.4. The best minimum visibility metric observed in each run is reported in Figure 6.8.

The evaluation shows that the IP method consistently outperform the gradient-based method, which can be explained by two factors. First, the loss function being maximised in the gradient method is non-convex and presents local-maxima, which may result in sub-optimal results. Secondly, the gradient-based method does not optimise the sensor-rail assignment. The latter factor is circumvented by performing multiple optimisation runs for the gradient-based method, each with a random sample of sensor-rail assignment. However, the variance of the visibility metric obtained across runs, observed in Figure 6.8, suggests that ten samples may not be enough to explore the sensor-rail assignment space. Including more samples of sensor-rail assignments requires more optimisation runs, which becomes time costly. On the other hand, the IP method handles the sensor-rail assignment naturally as the candidate sensor pose set includes sensor poses in all virtual rails. Furthermore, both proposed optimisation methods outperform the empirical set of sensor poses from Chapter 5, except for $N = 6$ where the gradient-based method has similar but smaller minimum visibility.

Figure 6.9 shows the resulting sensor poses found by each method for different numbers of sensors and the associated ECDF of the visibility metric of the target objects for each set of sensor poses. The visibility metric distributions obtained with IP solutions show similar visibility patterns, except for $N = 6$ where the heuristic IP approach has a significant advantage over its counterparts. The distribution of visibilities for gradient-based solutions is significantly skewed towards smaller visibilities if compared to IP solutions. Particularly, for $N = 5$, approximately 80% of the objects have less than 1000 points when observed by the gradient-based solution, while only 40% of objects have less than 1000 points for the IP solutions.

6.4.4 Comparison With Existing Works

The performance of the proposed sensor pose optimisation methods is compared with two existing works. Akbarzadeh *et al.* [128] maximise the coverage of a ground area using gradient-ascent and Zhao *et al.* [135] uses Integer Programming to maximise the number of target points visible in an environment. These methods are reproduced in the simulated T-junction environment considering the coverage of uniformly distributed points over the T-junction ground area. Note that these methods do not explicitly model the visibility of the target objects, instead they maximise the coverage of the ground area. The evaluation considers the ground surface coverage, *i.e.* the ratio of ground points that

Table 6.2: Comparison of Optimisation Results for Different Number of Sensors Across Proposed Methods

Method	**N**	**Min Visibility**	**Runtime till Best (min)**	**Overall Runtime (min)**
Gradient-based*	2	17	25	27
	3	55	32	32
	4	102	39	39
	5	123	30	46
	6	178	16	53
IP CBC	2	**26**	325	565
	3	67	416	656
	4	213	286	526
	5	**447**	175	415
	6	**590**	354	594
IP Naïve	2	26	0.2	240
	3	**114**	163	406
	4	201	44	284
	5	354	179	419
	6	405	97	337
IP MCMC	2	26	0.2	240
	3	107	190	430
	4	**220**	423	663
	5	321	87	327
	6	411	20	260
Empirical**	2	0	-	-
	3	33	-	-
	4	77	-	-
	5	81	-	-
	6	186	-	-

*Best results out of 10 runs with random initialisation. Overall Runtime reported for the single best run.
*Based on the empirical sensor poses from Chapter 5.

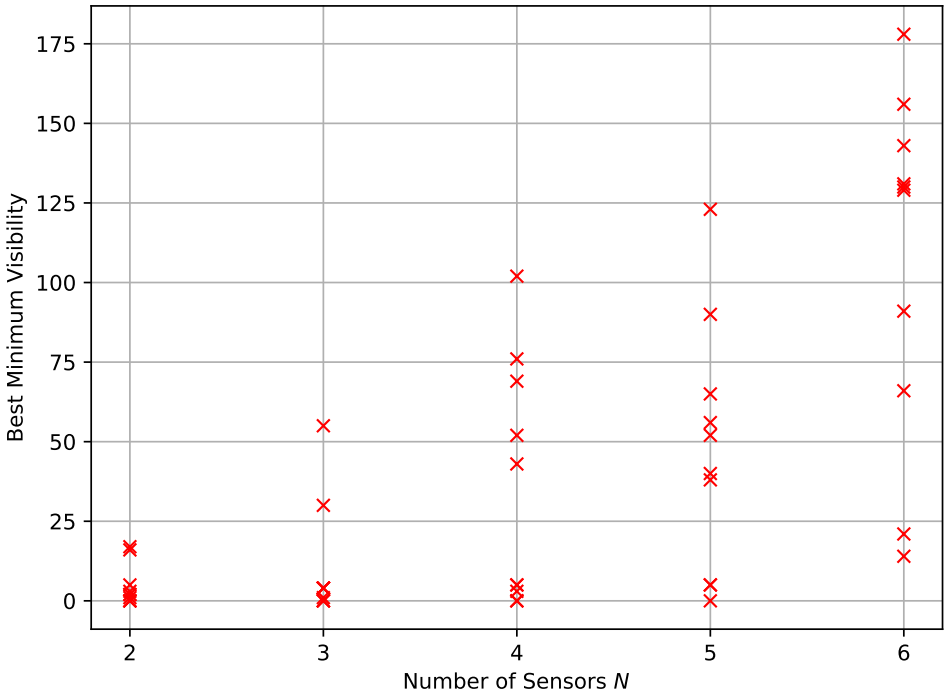
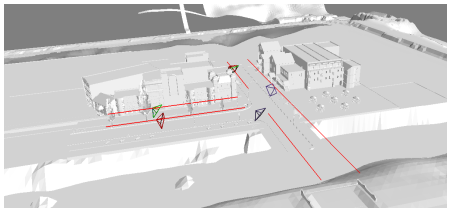


Figure 6.8: Best Minimum visibility for each out of the ten runs of the Gradient-ascent optimisation method for varying number of sensors N.

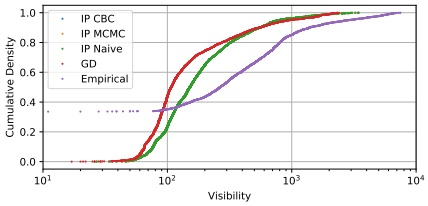
are visible to the sensors, and the minimum visibility of objects placed over this area. The results are reported in Table 6.3. These results show that the previous methods are successful in maximising the coverage of the T-junction's ground area. However, this does not guarantee the visibility of target objects since occlusions between objects are a key factor in determining the visibility of objects in cluttered environments. This underpins the importance of explicitly considering the visibility of target objects in contrast to the coverage of ground areas.

Table 6.3: Performance Comparison With Existing Works in Terms of Ground Area Coverage and Minimum Object Visibility for Different Number of Sensors

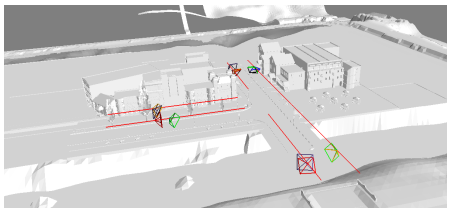
Method	**N**	**Ground Area Coverage (%)**	**Min Visibility**	**Overall Runtime (min)**
Akbarzadeh et al. [128]	2	51	0	2
	3	69	0	2
	4	73	0	2
	5	78	1	2
	6	91	41	2
Zhao et al. [135]	2	79	0	3
	3	88	0	3
	4	91	0	3
	5	92	0	3
	6	92	0	3



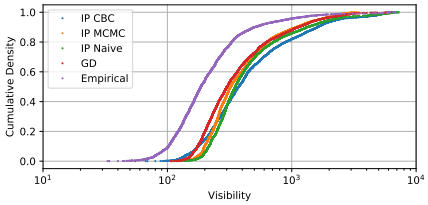
(a) $\hat{S}$ for $N = 2$



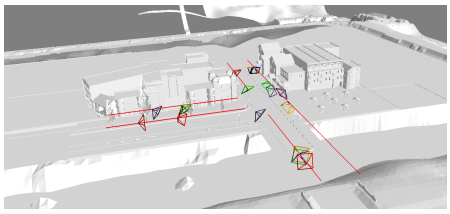
(b) ECDF of $\text{vis}(o, \hat{S})$



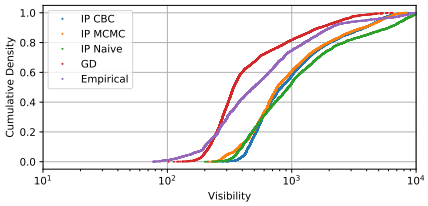
(c) $\hat{S}$ for $N = 3$



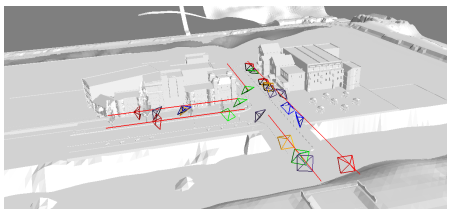
(d) ECDF of $\text{vis}(o, \hat{S})$



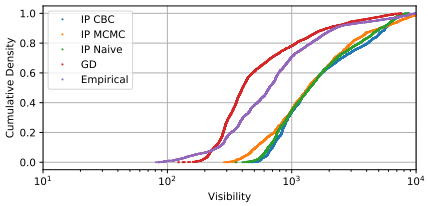
(e) $\hat{S}$ for $N = 4$



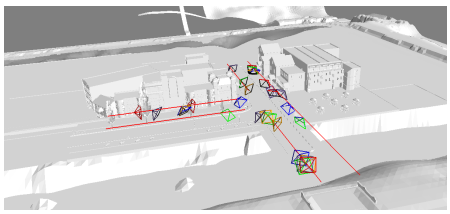
(f) ECDF of $\text{vis}(o, \hat{S})$



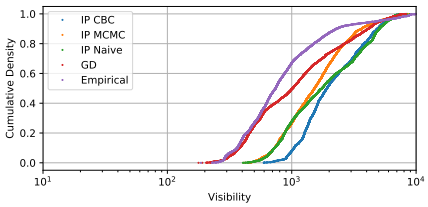
(g) $\hat{S}$ for $N = 5$



(h) ECDF of $\text{vis}(o, \hat{S})$



(i) $\hat{S}$ for $N = 6$



(j) ECDF of $\text{vis}(o, \hat{S})$

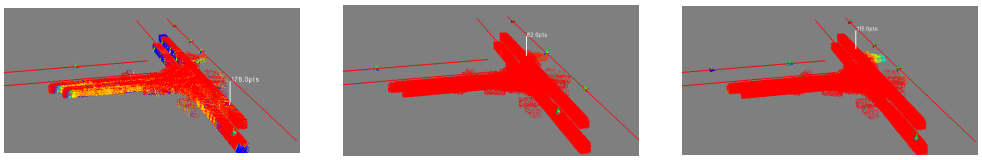
Figure 6.9: Resulting Sensor Poses and Visibility Distributions. The left column represents the perspective view of the junction showing the pose of the resulting set $\hat{S}$. The right column shows the ECDF of object's visibility for the optimal set of sensors found by different methods. The colour of the sensors in the perspective view follows the legend of the ECDF plot. Each row describes the results for a given number of sensors, denoted by N.

Table 6.4: Comparison Between Visibility Models Used in the Gradient-Based Method

Visibility Model	**Min Visibility**
Proposed (without Occlusion-Aware model, Eq. 6.7)	82
Proposed (with Occlusion-Aware model, Eq. 6.9)	**178**
Akbarzadeh et al. [128]	115

6.4.5 Comparison Between Visibility Models

A study comparing the performance of the gradient-based method is performed considering three different visibility models: the proposed visibility model with and without occlusion awareness (Eq. 6.7 and 6.9, respectively) and the visibility model from Akbarzadeh *et al.* [128]. This study considers $N = 6$ sensors and explicitly model the visibility of the target objects using the three aforementioned visibility models. Table 6.4 reports the results of this study. The proposed occlusion-aware visibility model achieves the best performance as it can realistically determine which points are visible and accordingly change the sensors' pose to account for potential occlusions. This is highlighted in Figure 6.10, depicting the point clouds of target points, where the colour of each point encodes its visibility score, ranging from blue (invisible) to red (visible). Note that the proposed occlusion-aware visibility model correctly identify non-visible parts of the objects due to occlusion (blue) or only partially visible (yellow). In contrast, the two other visibility models fail to identify areas of occlusion, mistakenly determining that all points are visible (red). As a result, the optimisation process cannot improve the visibility of such areas.



(a) (b) (c)

Figure 6.10: Point clouds showing the target points over all objects in all the frames for three visibility models. a proposed visibility model including occlusion awareness, b proposed visibility model without occlusion awareness and c Akbarzadeh et al. [128]. The point colours indicate the visibility score Ψ, ranging from blue ($\Psi = 0$, invisible) to red ($\Psi = 1$, visible). The white vertical pointer marks the position of the object with least visibility. Sensors poses are indicated by XYZ axis within coloured spheres.

6.5 Summary

This chapter investigated sensor pose optimisation strategies targetting the visibility of multiple objects in crowded environments. The systematic study, in addition to the proposition of novel approaches for sensor pose optimisation, reveals a number of key insights that can be useful for researchers and system designers. Firstly, explicit modelling of the visibility of the target objects is critical when optimising the poses of sensors, particularly in cluttered environments where sensors are prone to severe occlusions. Secondly, rendering-based visibility models can realistically determine the visibility of target objects at the pixel level and, thus, improve the pose optimisation process. Thirdly, the IP optimisation method seems to outperform the gradient-ascent method in terms of minimum object visibility, at the cost of increased computational time. The sensor pose optimisation methods proposed in this chapter can guide the deployment of sensor networks in traffic infrastructure to maximise the visibility of objects of interest. Such sensor network infrastructures can be used to increase the safety and efficiency of traffic monitoring systems and aid the automation of driving in complex road segments, particularly, in areas where accidents are more likely to happen.

Chapter 7

Efficient Relative Pose Estimation for On-board Sensors

The previous chapters assumed fixed infrastructure sensors whose poses were known by design, *i.e.* Chapter 6, or could be obtained accurately through calibration [175, 176]. Generalising the cooperative perception concept from fixed infrastructure to on-board sensors, *i.e.* sensors within vehicles, requires estimating the relative pose between pairs of sensors such that the data can be aligned in the same coordinate system. While the majority of previous cooperative perception methods have considered GPS/GNSS systems to obtain such relative transformation, this approach is prone to errors in the orders of meters in translation and tens of degrees in rotation estimation [115]. This chapter investigates how to obtain the relative pose transformation between a pair of sensors by exploiting their sensor data alone, namely using point cloud registration methods.

Existing registration methods are often designed and evaluated assuming a significant overlap between the input point clouds. This assumption is valid for applications such as SLAM [140, 195] and lidar odometry [161], where pairs of point clouds are obtained sequentially in adjacent time steps by a single vehicle navigating in a driving environment. On the other hand, applications such as cooperative perception and multi-agent SLAM [162, 196] require registering point clouds obtained simultaneously from a pair of sensors on two different vehicles that are potentially far apart, and thus, may have low field-of-view overlap, *e.g.* Figure 7.8. As the relative translation between the sensors increases, the number of identifiable correspondences decreases, which poses challenges in registering the point clouds accurately.

The majority of existing point cloud registration methods cannot guarantee real-time execution. Traditional local registration methods such as Iterative

Closest Point (ICP) [142] solve the problem iteratively assuming an initial relative pose. However, the iterative nature of such methods renders them unfeasible for real-time applications, particularly considering large scale point clouds. These methods are also prone to non-optimal solutions when the initial pose estimate is poor, which may be addressed with global-optimisation variants [143, 144] at the cost of higher computational complexity. Another category of methods identify correspondences between point clouds using a distance metric between hand-engineered features [149] or learned point-wise features [154]. These correspondences are often contaminated by a large number outliers and must be filtered using Random Sample Consensus (RANSAC) [150, 160] or learned models [155], which further increases the registration execution time. Furthermore, state-of-the-art learning-based models [154, 155] require computationally demanding 3D convolutions and generate numerous putative correspondences, introducing a bottleneck on the RANSAC loop and rendering real-time execution unfeasible.

To mitigate the aforementioned limitations, a novel point cloud registration method capable of operating in real-time and robust to low-overlapping point clouds is proposed. The proposed method identifies correspondences between the source and target point clouds by learning point-wise features. A novel encoder hierarchically subsamples the point clouds to reduce the number of key points and improve the run-time performance. The resulting features are refined using self- and cross-attention based on a graph neural network. The attention network leverages geometrical relationships between key points and their features to improve the correspondence accuracy, particularly in regions of low overlap. The relative pose parameters are obtained by fitting the learned correspondences using RANSAC to robustly reject outliers. During inference, the RANSAC fitting is done efficiently considering a small number of correspondences, which allows end-to-end inference times below 410ms. The model is trained and evaluated separately on the KITTI odometry dataset and a novel Cooperative Driving Dataset (CODD). The relative translation between sensors in CODD ranges up to 30m, introducing challenging pairs of point clouds with low overlap. This chapters' contributions are summarised as:

- A computationally efficient point-wise feature encoder that allows identifying correspondences between point clouds;
- A graph neural network that provides self- and cross-attention between point clouds and improves the quality of correspondences;
- A novel registration method for point clouds that is robust to partially-overlapping point clouds and capable of operating in real-time;
- A new synthetic lidar dataset containing low overlapping point clouds in

a wide range of driving scenarios;

7.1 Problem Formulation

Given two input point clouds $P_X \subset \mathbb{R}^3$ and $P_Y \subset \mathbb{R}^3$, the registration problem is to estimate the rigid relative pose transformation that aligns P_X into the coordinate system of P_Y. This transformation is parametrised by a rotation matrix $R \in SO(3)$ and a translation vector $t \in \mathbb{R}^3$. The problem can be solved by identifying pairs of correspondences between P_X and P_Y. Given a set of correspondences, $X = \{x_1, \ldots, x_N\} \subset P_X, Y = \{y_1, \ldots, y_N\} \subset P_Y$, where $(x_i, y_i), i = 1, \ldots, N$ are correspondence pairs, the transformation parameters are obtained by the minimisation of the least-squares error:

$$E(R,t) = \frac{1}{N} \sum_{i=1}^{N} \|Rx_i + t - y_i\|^2 . \tag{7.1}$$

This error is a form of the Orthogonal Procrustes problem [158] and admits the closed-form solution described below. First, the centroids are computed as

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i, \qquad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i, \tag{7.2}$$

and the covariance matrix, denoted by H, is obtained using

$$H = \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})^\mathsf{T}. \tag{7.3}$$

Finally, the rotation matrix and translation vector R, t that minimise Eq. 7.1 are computed in closed-form as

$$\begin{aligned} R &= V \begin{bmatrix} 1 & & \\ & 1 & \\ & & \det(V^\mathsf{T}U) \end{bmatrix} U^\mathsf{T}, \\ t &= -R\bar{x} + \bar{y}, \end{aligned} \tag{7.4}$$

considering the Singular Value Decomposition (SVD) $H = USV^T$. The next section proposes a novel method to efficiently obtain correspondences between pairs of point clouds.

7.2 Proposed Method

This section presents a novel method for robust point cloud registration using learned correspondences targetting efficient, real-time inference. Figure 7.1 describes the components and data flow of the proposed method. The proposed

method can be summarised as follows:

(A) Both point clouds are fed to an encoder to obtain a subset of key points and associated point-wise features.

(B) A graph neural network refines the point-wise features considering self- and cross-attention.

(C) The resulting features are used to identify correspondences between the source and target key points.

(D) The relative transformation parameters R, t are robustly estimated using a RANSAC formulation of the problem defined in Section 7.1.

The aforementioned components and the training process are described in the following subsections.

7.2.1 Point-wise Feature Encoder

The encoder is a core component of the pipeline, as it computes point-wise features that will be used to identify correspondences. A novel and computationally efficient encoder network is proposed based on Set Abstraction (SA) and Feature Propagation (FP) layers [97]. While previous works [197] have used PointNet++ feature encoders, the proposed encoder is distinguished by adopting a customised architecture that hierarchically subsamples points at each layer, resulting in improved computational performance. The proposed encoder architecture, including the hyper-parameters of each layer, is depicted in Figure 7.2. The encoder outputs subset of sampled coordinates (key points) from the source and target point clouds, denoted by X and Y, and their respective feature vectors, f_X and f_Y. The input to the encoder consists of 3D point coordinates and corresponding features, *e.g.* lidar return intensity. Note that the input features are optional, but in this work they consist of a single scalar per point representing the lidar intensity. The source and target point clouds are fed to the encoder independently.

The first four encoder layers are SA layers. A SA layer consists of four operations:

1. n coordinates are sampled from the previous layer using Farthest Point Sampling (FPS) [198].

2. A local neighbourhood of each sampled coordinate is established by selecting all points within radius r of the respective coordinate.

3. The features of the points in each neighbourhood are fed to a shared Multi Layer Perceptron (MLP), denoted as a list L containing the number of intermediate nodes per layer.

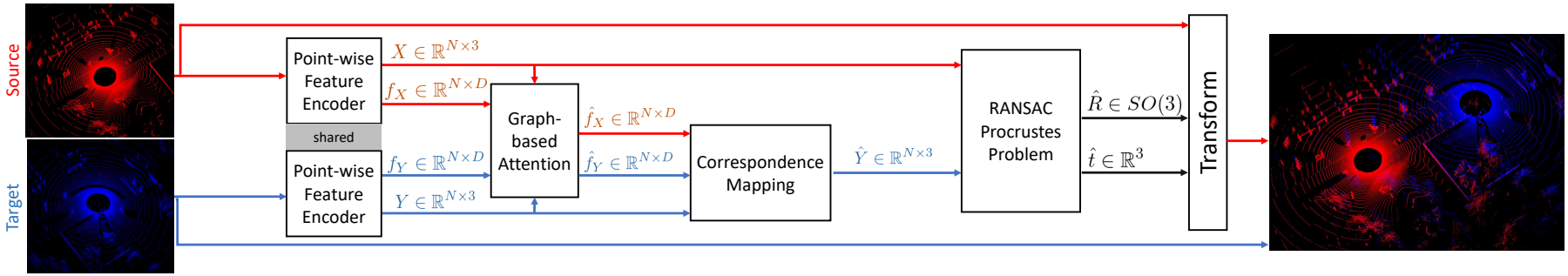


Figure 7.1: Pipeline and data flow of the proposed point cloud registration method.

4. The resulting feature vectors of the n sampled coordinates is computed using an aggregation function (*max-pooling*) over the MLP output of the points in the respective neighbourhoods.

Each SA layer hierarchically subsamples and aggregates information from the previous layer with progressively larger receptive volumes, which is a fundamental step in reducing the computational cost of the proposed pipeline. At the same time, it is also important not to discard valuable information, *i.e.* prioritising that points from one layer are within the neighbourhood of sampled points in the next layer. This trade-off is achieved by tuning the layers' hyper-parameters, namely n, r, L, such that the sampled points' neighbourhood include most points from the previous layer.

The last encoder layer is an FP layer. It propagates high level information from SA4 to the points in the previous layer (SA3) as illustrated in Figure 7.2. This is achieved by interpolating the feature vectors in SA3 layer using the features from the three nearest-neighbour coordinates in SA4. The final features are obtained fusing the original SA3 features with the interpolated SA4 features using a shared MLP, represented by a list L of intermediate nodes. More details about the interpolation can be found in [97].

The encoder hyper-parameters are optimised for large outdoor driving environments considering trade-offs between computational performance and registration accuracy, as discussed below. Having a small number of sampled points (reducing n) increases the computational efficiency since less points are processed in the subsequent layer, however, makes the point cloud sparser and, thus, more challenging to find correspondences. The hyper-parameter r controls the receptive volume by adjusting the maximum radius of the neighbourhood aggregation operation. While decreasing r reduces the computational cost (less points to aggregate), it also limits the amount of information a given point has about its neighbourhood, which results in points with less distinctive features. Finally, L controls the number of nodes at each MLP layer, which defines the model complexity when transforming features from one layer to the next.

Several variants of the proposed encoder architecture were evaluated, namely, changing the number of encoder layers and varying the layers hyper-parameters n, r, L, with the best iteration being the one reported in this chapter. The registration results are most sensitive to the number of points sampled per layer, n, since this has the most impact in the sparsity of the key points. During the hyper-parameter experiments, it was observed that adjusting the radius r in relation to the number of sampled points n and the density of the point cloud can improve the registration performance. Specifically, ensuring the radius r at each layer is large enough to aggregate neighbouring points, otherwise, a point only has information about itself and fails to propagate

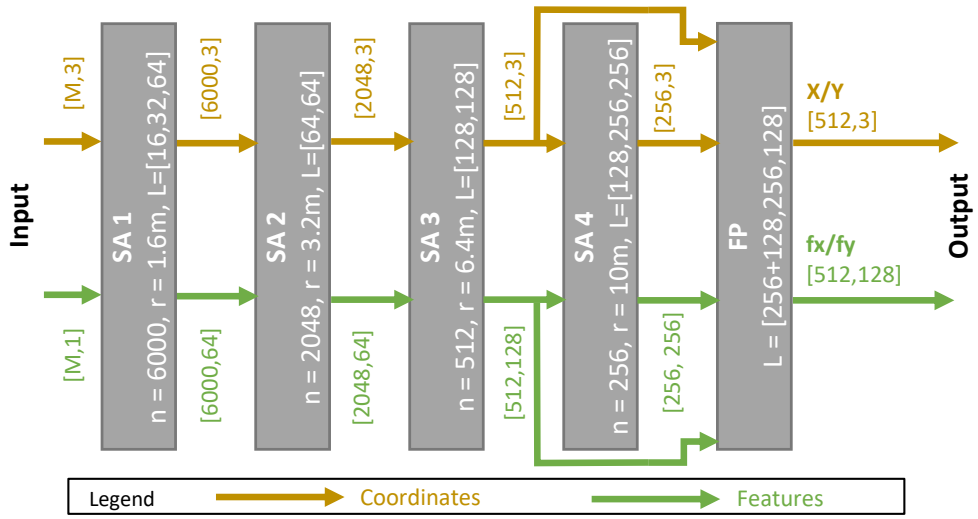


Figure 7.2: Point-wise feature encoder model architecture. The model consists of four Set-Abstraction (SA) layers and one Feature Propagation (FP) layer. Point coordinates and features are represented in yellow and green, respectively. The brackets indicate the dimensionality of each matrix. The number of input points, denoted by M, is arbitrary, and the model consistently outputs $N = 512$ point coordinates and their respective feature vectors with $D = 128$ dimensions.

neighbourhood information.

7.2.2 Graph-based Attention

The feature vectors obtained with the encoder network represent local point cloud information. However, these features are agnostic to the global context of the point cloud. For example, if a point cloud contains multiple objects, *e.g.* trees, it would be difficult to distinguish between individual trees. Another problem, most critical for low overlapping point clouds, is that regions of overlap generally have different point densities in each point cloud, challenging the correct identification of correspondences since a point's features change with the density of points in its neighbourhood. To mitigate both problems, a graph-based attention module is proposed to transforms points' feature vectors considering the wider point cloud context (self-attention) and the context of both source and target point-clouds (cross-attention). Self-attention increases the distinctiveness of key points by attending to their surrounding context. The cross-attention layer learns to refine point-wise features by attending to the most similar features across point clouds. Differently from the feature-based attention mechanism in [159], the proposed graph attention layers leverage both spatial and feature dimensions of local neighbourhoods to refine point-wise features. Both layers, illustrated in Figure 7.3, increase the likelihood of finding accurate correspondences, even in cases of low overlap.

The self-attention layer introduces attention between points within the same

point cloud. A graph connecting the points (nodes) is created for each point cloud using the k-Nearest-Neighbours of the points' spatial coordinates. Let f_i be a feature vector from either f_X or f_Y. The self-attention layer computes a residual term for f_i using the Crystal Graph Convolution Operation [199]:

$$\hat{f}_i = f_i + \max_{j \in \mathcal{N}(i)} \sigma(z_{i,j} W_f) \odot \text{Softplus}(z_{i,j} W_s), \tag{7.5}$$

where $\mathcal{N}(i)$ indicates the k neighbour nodes of i, $z_{i,j} = [f_i, f_j]$ is the aggregated features of nodes i, j. The function $\sigma(\cdot)$ indicates the sigmoid function, and the *Softplus* function is defined as $\text{Softplus}(z) = \log(1 + e^z)$. The attention matrices W_f, W_s are parameters to be learned and the operation $\odot$ represents element-wise multiplication. The number of nearest neighbours is set as $k = 32$, which provides a good trade-off between accuracy and computational efficiency. This process is performed independently with shared parameters for both source and target point clouds.

Following the self-attention layer, the cross-attention layer allows interaction between the source and target point cloud features. This layer creates a bi-partite graph between source and target points. Each source point is connected to the k-Nearest-Neighbours nodes in the target point cloud, where the distance metric is the dot product between the feature vectors of the respective points. This layer uses the same residual update rule from Eq. 7.5, considering the different underlying graph and independent attention matrices W'_f, W'_s. The graph-attention network output is given by the updated feature vectors from source and target point clouds, denoted by $\hat{f}_X$ and $\hat{f}_Y$, respectively.

7.2.3 Identifying Correspondences

The correspondences between the point clouds can be obtained by comparing the point-wise features of the source and target key points, denoted respectively as $\hat{f}_X, \hat{f}_Y \in \mathbb{R}^{N \times D}$. These feature vectors are normalised to unity D-dimensional vectors using the Euclidean norm. A matching probability map indicating the probability of correspondences between X and Y is computed as

$$\phi = \text{Softmax}\left(\frac{\hat{f}_X \cdot \hat{f}_Y^{\mathsf{T}}}{T}\right) \in \mathbb{R}^{N \times N}, \tag{7.6}$$

where T is a temperature hyper-parameter and the *Softmax* function is applied row-wise. Each element ϕ_{ij} represents the probability that the i-th key point in X matches the j-th key point in Y. The *Softmax* function scales the coefficients of each row ϕ_i, ensuring a probability distribution over the points in Y. The temperature parameter, denoted by T, controls the entropy of distribution across points in Y. In the limit, when $T \to 0^+$, the coefficients

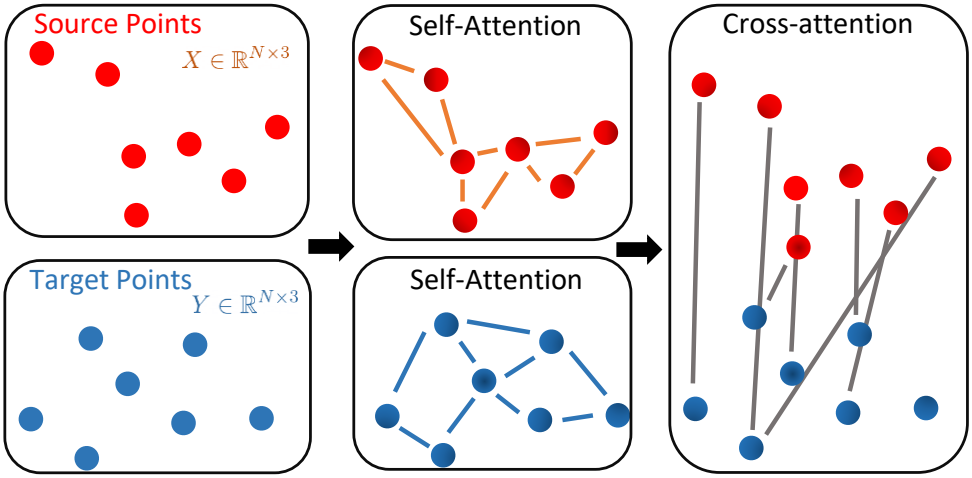


Figure 7.3: Graph-based Attention Representation. Points are represented as graph nodes. The nodes are defined by their position, *i.e.* 3D coordinates, and their feature vector (not represented in the image). The graphs connecting the points are created based on the k-NN ($k = 32$) between the points. In the self-attention layer, the k-NN distance is the Euclidean distance between points' spatial coordinates. In the cross-Attention graph, the k-NN distance is the dot product between points' feature vectors.

become the one-hot encoding of the point in Y with the highest similarity (*i.e.* dot product). Finally, each key point $x_i \in X$ is matched to the key point in Y with highest correspondence probability, resulting in the set of correspondence pairs $\{(x_i, y_{\arg\max_j \phi_{ij}}), i = 1, \ldots, N\}$. The ordered set of correspondences in Y is denoted as $\hat{Y} = \{y_{\arg\max_j \phi_{1j}}, \ldots, y_{\arg\max_j \phi_{Nj}}\}$.

7.2.4 Estimating Transformation Parameters

The previous step computes a correspondence for every point in X. In practice, only a fraction of points in X will have correspondences in Y, particularly in the case of partially overlapping point clouds. Sensor noise and varying point densities can also lead to encoding errors and erroneous correspondences. To mitigate the effect of correspondence outliers, a common practice is to use sample consensus algorithms such as RANSAC [149, 150]. A general version of this algorithm applied to this problem consists of three steps:

1. Create a hypothesis: Sample a minimal set of three correspondences from the set of correspondences and compute the transformation parameters using Eq. 7.4.

2. Score the hypothesis based on consensus: Compute the number of inlier correspondences, where a correspondence $(x_i, \hat{y}_i), x_i \in X, \hat{y}_i \in \hat{Y}$ is an inlier if $\|Rx_i + t - \hat{y}_i\| \leq \kappa$, where κ is the inlier threshold, R, t are the

hypothesis parameters computed in the first step.

3. Repeat the previous steps for L times and select the hypothesis with the highest number of inliers.

The number of tested hypotheses, denoted by H, offers a trade-off between computational performance and robustness to outliers. H can also be derived to achieve a desired confidence of the selected hypothesis [150], *i.e.* sampling an outlier-free set of correspondences. While previous works used RANSAC with a large set of putative correspondences [149], this may be unfeasible for real-time systems. In this work, negligible RANSAC computational cost is achieved by learning a small set of correspondences ($N = 512$), which allows to reduce the number of hypotheses being tested.

7.2.5 Training Process

The training process consists of optimising the encoder and attention networks to find accurate correspondences where they exist. To that end, the matching probabilities of ground-truth correspondences must be maximised and the matching probability of non-corresponding points must be minimised. This is achieved by directly minimising the following loss function:

$$\mathcal{L} = \frac{1}{N_c} \sum_{i=1}^{N} \delta_i \left[-\phi_{i\hat{j}} + \frac{\lambda}{N-1} \sum_{j=1, j\neq\hat{j}}^{N} \phi_{ij} \right], \tag{7.7}$$

where the binary variable δ_i indicates whether the source point x_i has a correspondence in Y and $\hat{j}$ represents the index of the corresponding point in Y. Additionally, $N_c = \sum_{i=1}^{N} \delta_i$ is the number of ground-truth correspondences and λ is a hyper-parameter scaling the contributions of incorrect matches into the loss function. A point in X is considered to have a correspondence in Y if, under the ground-truth transformation, it is within a distance smaller or equal to 1.6 meters from a point in Y. This inlier distance is arbitrary and was chosen based on the smallest encoder radius. Data augmentation is employed by applying random rotation transformations to both input point clouds and adjusting the ground-truth rotation matrix accordingly. The optimisation details are described in Section 7.3.3.

7.3 Performance Evaluation

In this section, the datasets and the evaluation metrics are described, followed by the implementation details. Next, the performance of the proposed method is compared against traditional baselines, including ICP [142], FPFH RANSAC [149] and TEASER [151]; and two state-of-the-art learning-based methods:

FCGF [154] and DGR [155]. Finally, an ablation study identifying the impact of the proposed attention network into the registration performance is presented.

7.3.1 Dataset

The KITTI Odometry dataset [47] is traditionally used to evaluate point cloud registration methods in outdoor environments. The evaluation in this chapter follows the evaluation protocol of recent methods [154, 155, 200, 201], which adopt sequences 0 to 5 for training, 6 to 8 for validation and 9 to 10 for testing. In each sequence, the samples are created by selecting pairs of points clouds obtained sequentially by a single vehicle such that the translation between the poses is less than 10m. The ground-truth pose is provided by GPS and refined using ICP to reduce misalignment.

The distribution of poses in the KITTI dataset is limited to the trajectory of a single vehicle as it navigates the environment. In practice, registration methods must be resilient to point clouds with arbitrary relative pose, where the overlap between point clouds may vary significantly across samples. To this end, the Cooperative Driving Dataset (CODD) [202], an open-source synthetic dataset containing lidar point clouds collected simultaneously from multiple vehicles, is introduced. This dataset is created using CARLA [165] and features a diverse range of driving environments, including rural areas, suburbs, and dense urban centres. The dataset consists of 108 sequences, which are split into three independent subsets for training, validation and testing, as detailed in Table 7.1. The samples are created by selecting all pair-wise combinations of point clouds obtained from vehicles driving simultaneously within a vicinity considering a maximum distance of 30m. Figure 7.4 presents the cumulative density plots of the relative distance (translation vector norm), rotation angle and overlap ratio of the pairs of point clouds in each dataset. The overlap ratio measures the overlap between point clouds as the percentage of points in the source point cloud that, when aligned, are within a distance smaller than γ to any point in the target point cloud [203]. The CODD dataset has a significantly broader distribution of relative distance, rotation angles and overlap ratio between the point cloud pairs, which provides representative scenarios for cooperative perception and multi-agent SLAM.

Table 7.1: Dataset Details

KITTI Odometry	**Train**	**Validation**	**Test**
# sequences	5	2	2
# samples (pairs of point clouds)	1358	180	555
CODD	**Train**	**Validation**	**Test**
# exclusive maps	6	1	1
# sequences	78	14	16
# samples (pairs of point clouds)	6129	1339	1315

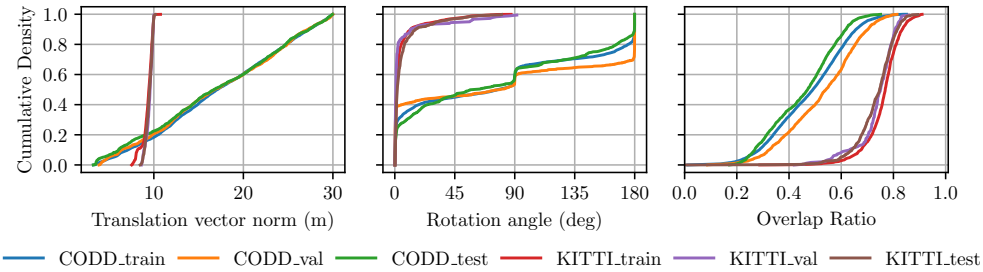


Figure 7.4: Cumulative density of the relative distance and rotation angle between coordinate systems of the pairs of point clouds in each subset.

7.3.2 Evaluation Metrics

Following previous studies [154, 155], the registration performance is evaluated in terms of the translation and rotation errors given by

$$\mathrm{TE} = \left\| \hat{t} - t^g \right\|_2 , \tag{7.8}$$

$$\mathrm{RE} = \arccos \frac{\mathrm{Tr}(\hat{R}^\mathsf{T} R^g) - 1}{2}, \tag{7.9}$$

where R^g, t^g denotes the ground-truth rotation matrix and translation vector, respectively. These metrics are reported considering their mean value over all dataset samples, denoted as Mean Translation Error (MTE) and Mean Rotation Error (MRE), respectively. The recall rate, measured as the ratio of successful registrations to the total number of samples, is also considered. The success criteria is TE $<$ 0.6m and RE $<$ 5deg following [155]. The runtime performance is evaluated as the average inference time for the registration of a pair of point clouds disregarding the data loading time.

7.3.3 Implementation Details

The proposed method is implemented using PyTorch [166], PyTorch Geometric [204], the CUDA implementation of SA and FP layers from [97] and the Open3D [205] Procrustes RANSAC implementation. The model is trained independently for each dataset using the Adam [191] optimiser with a learning

Table 7.2: Evaluation Results on KITTI Test Set

Model	**MTE [cm]**	**MRE [deg]**	**Recall**	**Time/sample [s]**
FPFH TEASER [151]	18.8	0.76	0.984	2.26
FCGF RANSAC [154]	23.7	0.034	0.975	26.29
DGR [155]	15.6	1.43	0.982	7.60
Proposed	26.1	0.74	0.949	0.41
Proposed + ICP	8.2	0.23	0.985	3.68

rate of 0.1, $\epsilon = 10^{-4}$, $\beta_1 = 0.9, \beta_2 = 0.999$ and a batch size of 6 (pairs of point clouds). The model is trained for twenty epochs and the learning rate is reduced in half after every five epochs. For evaluation, the model with the lowest validation loss is selected. The temperature hyper-parameter, described in Section 7.2.3, is set to $T = 10^{-2}$, and the loss scaling hyper-parameter is set to $\lambda = 10$. During inference, the RANSAC inlier threshold, denoted as κ, is set to 0.5m, and the maximum number of RANSAC iterations, denoted by H, is computed to achieve 0.999 confidence in the selected hypothesis within a limit of 10^5 iterations. The point clouds from both datasets are downsampled using voxel sizes of 0.3m following previous methods [154, 155]. The overlap ratio distance threshold, γ, is set to 0.3m, following the down-sampling voxel size. The experiments are carried out on a Xeon ES-1630 CPU and Quadro M4000 GPU with 8 GB of memory. For fair comparison, all baselines are also evaluated on the same hardware. The official implementation and pre-trained models (30cm voxel) are used for the evaluation of [154, 155] in the KITTI dataset; likewise, the official TEASER [151] implementation is adopted; and the Open3D [205] implementation of FPFH, RANSAC and ICP is used for the evaluation of the respective methods.

7.3.4 Performance on the KITTI Dataset

The evaluation results, presented in Table 7.2, show that the proposed method achieves competitive registration errors compared to other methods at a significantly lower inference time – more than five times faster than the fastest baseline. While the proposed method has a marginal increase in mean translation error compared to baseline methods, it achieves on-par recall rate relative to baseline methods. The mean translation and rotation errors of the proposed method can be further reduced using ICP for refinement (Proposed + ICP), at the cost of increased inference time. Figure 7.5 shows the cumulative distribution of rotation and translation errors, and inference times for different methods. Figure 7.5 shows qualitative results of the proposed method on the KITTI dataset.

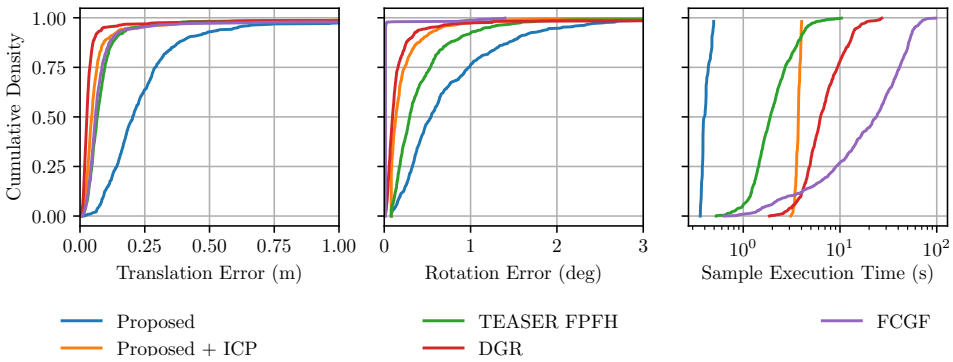


Figure 7.5: ECDF of the Translation Error, Rotation Error and Sample Execution Time for Different Methods on the KITTI Test Set.

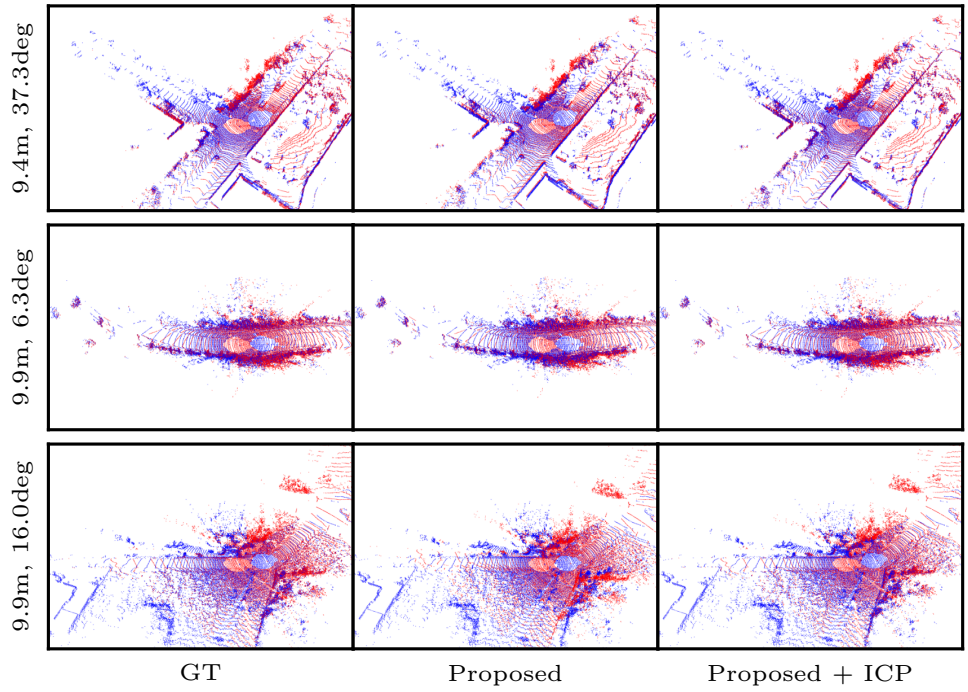


Figure 7.6: Qualitative of Results of the Proposed Method on the KITTI Test Set. Each row represents a different sample and the vertical label shows the ground-truth relative transformation between point clouds quantified by the norm of the relative translation vector in meters and relative rotation angle in degrees (axis-angle convention).

7.3.5 Performance on the CODD Dataset

The performance of the proposed method and baselines is evaluated on challenging low-overlapping pairs of point clouds. For a fair comparison, learning-based methods [154, 155] are also trained on the CODD dataset. Table 7.3 presents the results on the CODD test set, aggregated by the overlap ratio between point clouds in four progressively larger intervals – the last interval contains all samples. Traditional methods are not resilient to low overlapping points clouds, as the registration error increases significantly when considering lower overlap ratios, as shown in the first three rows of Table 7.3. In contrast, the learning-based baselines are reasonably robust to low overlapping point clouds and achieve high recall rates on all intervals. However, the latter methods demand substantial running times due to their complex encoders and the filtering of a high number of putative correspondences. In contrast, the proposed method achieves similar or better recall rates to the learning-based baselines with more than 35 times faster inference times. This is achieved by the efficient encoder design which outputs a small number of correspondences, which in turn reduces the RANSAC inference time. This efficient encoder strategy comes at the cost of a slight increase of the MTE and MRE metrics, as compared to DGR [155]. To mitigate this, ICP refinement is applied to the proposed model's output (Proposed + ICP), which allows achieving similar MTE and MRE for highly overlapping point clouds and outperforming all baselines on low-overlapping point clouds. Although the ICP refinement comes with an additional computational cost, this approach still achieve a nine-fold speed-up compared to competing learning-based baselines. Qualitative results are presented in Figure 7.8.

Figure 7.7 shows the Empirical Cumulative Density Function (ECDF) of the translation error, rotation errors, and inference time for different methods. The distributions indicate that the proposed method with ICP refinement has the best translation error across samples, closely matched by DGR [155], however with one order of magnitude smaller inference time. The inference time distributions show that the proposed method is the fastest among baselines, with an inference time of 320ms on average, with negligible standard deviation (17ms). Although the proposed method cannot perform inference at the frame rate of common lidar sensors, *e.g.* 10Hz, it must be noted that applications such as multi-agent SLAM and cooperative perception are not expected to receive sensor data at 10Hz for two reasons. First, bandwidth constraints limit the transmission rate of raw point cloud data and second, the redundancy in sequential scans is large when considering such frame rates, so the information gain is small. For example, [107] assumes a lidar frame transmission rate of 1 Hz for cooperative 3D object detection and [196] assumes a maximum transmitted

Table 7.3: Evaluation Results on CODD Test Set

	Overlap Ratio > 0.6			Overlap Ratio > 0.5			Overlap Ratio > 0.4			Overlap Ratio > 0			Time/sample [s]	
Method	**MTE [m]**	**MRE [deg]**	**Recall**	**MTE [m]**	**MRE [deg]**	**Recall**	**MTE [m]**	**MRE [deg]**	**Recall**	**MTE [m]**	**MRE [deg]**	**Recall**	**Mean**	**Std**
ICP [142]	1.69	36.11	0.67	4.81	68.67	0.38	9.11	66.42	0.17	16.14	74.84	0.07	0.38	0.048
RANSAC FPFH [149]	1.59	1.42	0.47	1.25	1.58	0.28	2.64	2.42	0.18	9.55	9.49	0.09	71.69	15.37
TEASER FPFH [151]	0.04	0.10	1.00	1.61	23.33	0.86	4.29	36.42	0.69	12.87	69.74	0.39	1.13	0.24
RANSAC FCGF [154]	0.09	0.01	1.00	0.10	0.01	1.00	0.12	0.01	1.00	1.70	0.11	0.91	16.5	22.4
DGR [155]	0.02	0.07	1.00	0.02	0.06	1.00	0.02	0.05	1.00	0.39	1.52	0.94	11.89	3.92
Proposed	0.14	0.21	1.00	0.19	0.25	0.99	0.22	0.29	0.98	0.28	0.41	0.94	0.32	0.017
Proposed + ICP	0.03	0.09	1.00	0.03	0.09	0.99	0.04	0.09	0.99	0.09	0.13	0.97	1.28	0.031
Proposed - Att	0.29	0.48	0.87	0.39	0.56	0.86	0.59	0.86	0.76	2.22	5.69	0.57	0.30	0.003

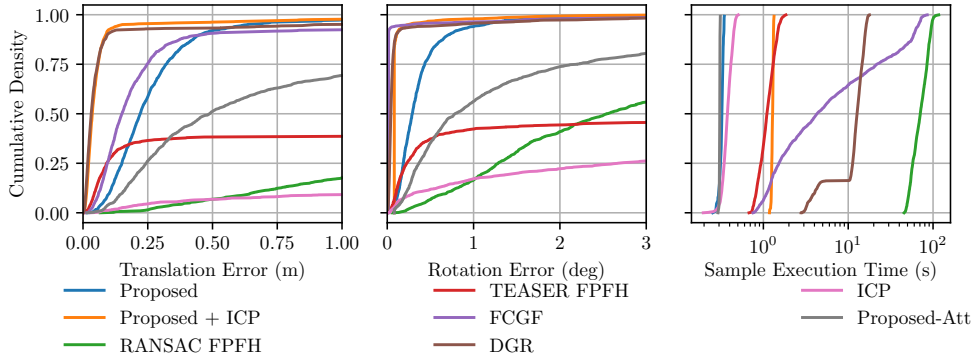


Figure 7.7: ECDF of the Translation Error, Rotation Error and Sample Execution Time for Different Methods on the CODD Test Set.

frame rate of 2Hz for cooperative SLAM. The proposed registration method can operate in real-time considering data input frequencies up to 3Hz.

7.3.6 Ablation Study

The impact of the proposed attention network into the registration performance is measured in terms of translation and rotation errors. This is achieved by removing the graph attention module, retraining the model and evaluating its performance on the CODD test set. The results, indicated in Table 7.3 "Proposed - Att", show that the graph-attention network plays a key role in improving the accuracy of the correspondences, resulting in lower translation and rotation errors. The benefits of the graph attention network is most significant for low-overlapping point clouds, as indicated by the last range group in Table 7.3, where the removal of the attention results in a 40% reduction of the registration recall and a significant increase in the mean translation and rotation errors.

7.4 Summary

This chapter investigated how to obtain the relative pose between on-board lidar sensors using the sensors' data alone. In doing so, a novel point cloud registration method focusing on fast inference of partially overlapping lidar point clouds was proposed. The performance evaluation considered the traditional KITTI benchmark dataset and the CODD dataset, a novel synthetic dataset

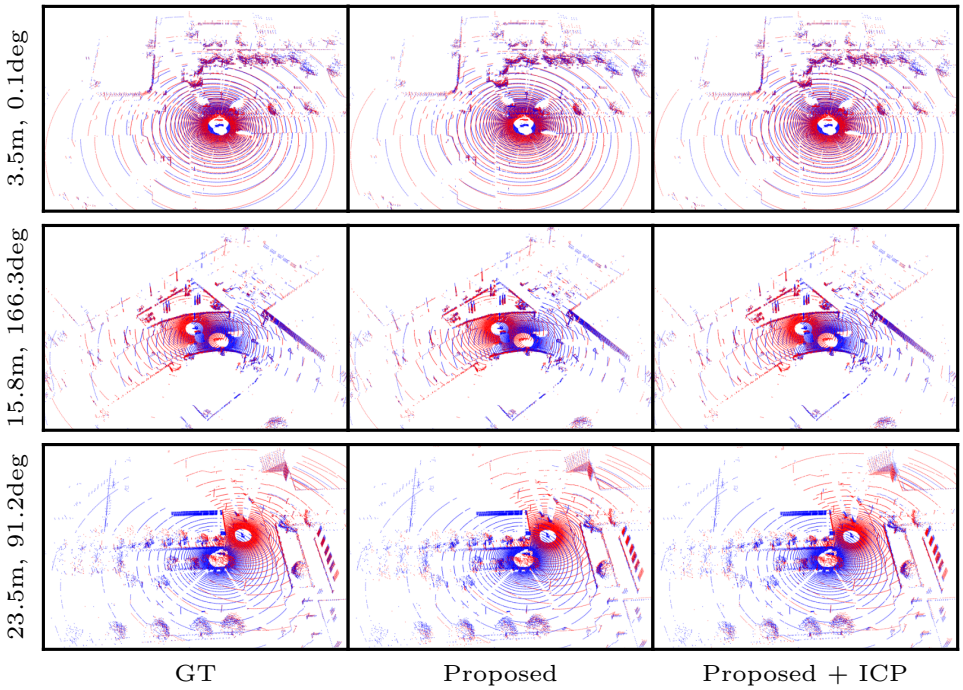


Figure 7.8: Qualitative of Results of the Proposed Method on the CODD Test Set. Each row represents a different sample and the vertical label shows the ground-truth relative transformation between point clouds quantified by the norm of the relative translation vector in meters and relative rotation angle in degrees (axis-angle convention).

featuring low overlapping point clouds with displacements of up to 30m. The proposed model can operate with latencies lower than 410ms and 320ms, depending on the dataset, a speed up of 5 and 35 times compared to competing methods in the KITTI and CODD dataset, respectively. The results show that the proposed model outperform baseline methods in terms of rotation and translation errors for pairs of point clouds with low overlap, achieving mean translation error below 30cm and angular errors below 0.41 degrees. Furthermore, ablation studies show that the graph attention module plays a key role in improving the quality of the correspondences in low overlapping point clouds, which results in higher registration performance.

Chapter 8

Conclusion

Following the research methodology proposed in Chapter 3, this thesis investigated different aspects of cooperative perception for driving applications through Chapters 4, 5, 6 and 7. This chapter discusses the key findings from those studies, how they address the research objectives from Chapter 3 and fill the research gaps identified in Section 2.7. Finally, the limitations of this research are presented along with suggestions for future work.

8.1 Discussion

8.1.1 Cooperative Object Classification

Chapter 4 analysed the impact of impairments in object classification performance for single-view methods and if cooperative object classification methods can mitigate these impairments. While previous multi-view object classification methods had already been devised, this study contributes to the literature by considering the effects of occlusions and sensor noise impairments and proposing a novel fusion method with greater resilience to these impairments. The experiments in this chapter show that single-view object classification methods are not resilient to occlusions and sensor noise. Specifically, the experiments indicate that introducing occlusions results in drops of up to 13% in terms of F1-score, and introducing sensor noise results in further performance drops of up to 28% for single-view methods. The novel cooperative classification method proposed in that chapter fuses images from multiple-views by concatenating the feature maps from each view and feeding this global representation to a fully connected classification layer. This cooperative method shows resilience to both occlusions and sensor noise, where the classification performance drops due to occlusion and sensor noise is less than 1% and 2%, respectively, in terms of the F1-score metric. When trained with occlusions, the proposed cooperative classification method shows greater generalisation to the size of occlusions and to the power of sensor noise compared to other cooperative baselines based in

voting (late-fusion) and feature-map pooling. Despite the limited application of this method in practical settings for the reasons discussed in Section 4.5, the results from this study showed the potential of cooperative perception and motivated the following studies in this thesis.

8.1.2 Cooperative 3D Object Detection

Chapter 5 investigated fusion schemes and the scalability of sensors for cooperative 3D object detection. This study expands on previous literature by proposing the use of infrastructure-based sensors and investigating how the number and pose of sensors affects the performance of cooperative object detection, which was a gap in the literature. Three fusion schemes were proposed, one based on early fusion (fusion of raw point clouds before the detection stage), late fusion (fusion of detected bounding boxes, after the detection stage) and a hybrid of the previous two. The experiments show that all three fusion schemes can mitigate the limitations from single-view sensing, namely occlusions and limited field-of-view, by incorporating information from multiple, spatially distributed sensors. In the context of the studied T-junction and roundabout scenarios, cooperative perception can increase the 3D object detection recall (ratio of detected objects to the total number of objects) compared to single-view sensing from 30% to more than 95%. Similarly, the precision (ratio of correct detections to the number of detections) can be enhanced from 80% to more than 95%, reducing the number of false positive detections, when considering multiple sensors.

Early fusion provides the best object detection performance (in terms of AP and recall rate), followed by hybrid fusion and late fusion schemes. While early fusion achieves more than 20% performance gain compared to late fusion, this comes at the cost of two orders of magnitude larger communication bandwidth. The hybrid fusion approach exploits another trade-off between object detection performance and communication bandwidth. In the studied scenarios, this fusion scheme can increase detection performance by more than 8% compared to late fusion with an increase of a single order of magnitude in communication bandwidth. The choice of fusion scheme depends on the application requirements in terms of bandwidth and detection performance (*e.g.* minimum object recall or precision rates). In terms of computational cost, the proposed model can operate with 298ms latency for late fusion and 380ms latency for hybrid/early fusion. The cooperative perception system proposed is agnostic to the object detection model, and thus, could be made more efficient by adopting newer and computationally efficient 3D object detection models [78].

The experiments in Section 5.4.4 suggest that the improvement in detection

performance gained by cooperative perception is associated with two factors. First, an increased field-of-view allows sensors to observe previously unseen areas. Second, an increased density of points on regions where sensors' field-of-view overlap. The latter can mitigate the effects of occlusions, reduce the number of false positives and increase the accuracy of the bounding box regression. Specifically, fusing the data from two sensors that observe the same region can increase the detection performance by up to 85% compared to using the observations from only one of the sensors in the studied scenarios. Still, the experiments in Section 5.4.3 show that the gain in detection performance with each additional sensor saturates as the number of sensors increases. Furthermore, the experiments show that the accuracy of an object's estimated 3D bounding box can be approximated using a logarithmic factor of the number of the points on the surface of the respective object. Although cooperative perception increases performance dependant upon the number of sensors and coverage of the environment, the results in the studied scenarios suggest these figures could generalise across environments provided sufficient number of sensors. The results in Chapter 5 illustrate how the detection performance varies depending on the number of sensors.

Since the publication of Chapter 5 results in [3], new studies exploring cooperative perception on infrastructure sensors for object detection and tracking have emerged. Particularly, Kloeker *et al.* [206] consider the low-level fusion of data from lidar-based infrastructure sensors to perform object detection and tracking. Their results show that using observations from eight lidar sensors can improve 3D object detection performance, measured in terms of Average Precision (AP), up to 120% compared to single sensor baselines. The rate of improvement observed in their study was larger than what was observed in Chapter 5 due to different environment and sensor configuration. Still, the general observation that cooperative perception significantly increases 3D object detection performance is consistent among both studies.

8.1.3 Infrastructure Sensor Pose Optimisation

Chapter 6 investigated how to optimise the pose of infrastructure sensors to maximise the visibility of target objects. A major category of sensor pose optimisation methods in the literature focus on maximising the coverage (visible ground area) of extensive 3D environments described by digital elevation maps. However, such a formulation does not consider the distribution of objects in the environment, and instead, assumes that an object would be visible if it is within a region covered by the sensors. As a result, these methods fail to detect and prevent occlusions between objects since they do not explicitly model the visibility of the target objects. Another category of methods considers the target

objects' visibility as a set of binary variables, which cannot describe degrees of visibility due to occlusions and due to its position *w.r.t.* the sensors. This study contributes to the literature by proposing an occlusion-aware visibility model based on a rendering engine that explicitly models the visibility of objects by counting the number of visible pixels per object.

The experiments in Chapter 6 show that the proposed sensor pose optimisation methods are capable of leveraging occlusion-aware visibility information to find sensor poses that allow for the visibility of all target objects in the considered T-junction scenario. While baselines that optimise sensors' pose based on ground coverage achieve a minimum visibility of 41 points using six sensors, the proposed methods based on gradient and IP optimisation achieve a minimum visibility of 178 and 590, respectively, for the same number of sensors. Although both of the proposed methods were able to guarantee the visibility of all objects, the Integer Programming-based method obtained the best performance in terms of minimum object visibility. This method obtained minimum object visibility up to 3.3 times higher than the gradient-based method and up to 7 times higher than gradient-based method without occlusion-awareness for the same number of sensors in the same scenario. These results shows that explicitly modelling occlusions can improve the minimum object visibility by more than 50% when compared to an occlusion-less model. The poor performance of the gradient-based method relative to the IP method can be explained by former method being prone to local-maxima which prevents finding the global optimum, and that this method is not able to directly optimise the sensor-rail assignment. The experiments revealed that explicitly modelling the visibility of target objects is critical when optimising the poses of sensors, particularly in cluttered environments where sensors are prone to severe occlusions. The proposed sensor pose optimisation methods can guide the deployment of sensor networks in traffic infrastructure to maximise the visibility of objects of interest.

8.1.4 Relative Pose Estimation

Chapter 7 investigated how to obtain the relative pose transformation between a pair of moving sensors using their data alone. This study expands the existing literature in cooperative perception by proposing a method that can obtain relative poses between sensors using a point cloud registration method. While some existing point cloud registration methods were deemed unsuitable for driving applications due to latency requirements, the proposed registration method allows for real-time inference considering frame rates of up to 3Hz. It also expands on the point cloud registration literature by evaluating traditional and state-of-the-art learning-based point cloud registration methods on low-

overlapping point clouds, which are representative of cooperative perception scenarios. The proposed method achieves computationally efficiency through a novel encoder that hierarchically subsamples the point clouds to reduce the number of key points at each layer. To improve the registration accuracy on point clouds with low overlap, a graph-based attention method refines point-wise features based on geometrical relationships between key points, and their features.

The evaluation results show that traditional registration methods are not resilient to low overlapping points clouds. In contrast, state-of-the-art learning-based methods are reasonably robust to low overlapping point clouds and achieve high recall rates on all overlap intervals. However, the latter methods demand substantial running times due to their complex encoders and the filtering of a high number of putative correspondences and are, thus, unsuitable for real-time applications. The proposed method achieves inference times between 410ms and 320ms, which are 5 and 35 times faster than competing point cloud registration baselines in the KITTI and CODD dataset, respectively. Furthermore, the proposed registration method can achieve mean translation errors below 30cm and rotation errors below 0.41 degrees for point clouds with translations of up to 30m, a decrease of 10cm and 1 degree when compared to state-of-the-art methods. While GPS/GNSS systems can exhibit translation errors in the order of meters in dense urban scenarios [115], this study suggests that point cloud registration can be used to obtain more accurate relative pose estimates in real-time.

8.2 Limitations

Although some of the limitations of the studies in this thesis have been discussed in their respective chapters, this section summarises the limitations of this research.

8.2.1 Generalisation

The preliminary concept of cooperative perception in Chapter 4 assumes that the pose of the object in each image is known and follow a canonical order, *e.g.* the first image shows the left view of the object, the second image shows the front view of the object, etc. In practice, this is an unrealistic assumption since the pose of the object is unknown a priori, which prevents the generalisation of this model in practical settings. Furthermore, the cooperative classification model assumes a fixed number of sensors, which provides limited usability in practice, where the number of sensors that observe a given object may change depending on the number of vehicles/sensors in the area.

The aforementioned limitations are addressed in the cooperative 3D object detection model in Chapter 5, which can fuse data from an arbitrary number of sensors and is agnostic to the order of sensors during data fusion. The evaluation of this model in a T-junction and a roundabout scenario shows that cooperative perception can increase the detection performance in terms of both recall and precision of the detected objects. The results show that the considered roundabout scenario requires two additional sensors to achieve the same level of performance achieved in the T-junction scenario due to factors such as a larger detection area. Although these suggest that cooperative perception can improve the detection performance in other driving environments, it is not possible to draw conclusions on whether the same level of performance gains will generalise to different types or sizes of junctions.

Another generalisation limitation of the object detection model in Chapter 5 regards the class of objects being detected. That detection model was designed to only detect a single class of objects: vehicles. However, the simulated dataset also includes other classes of objects, *e.g.* pedestrians and cyclists. These different object classes were included to increase the dataset diversity and prevent the model to overfit to the background environment. If these other objects were not included, the model could simply learn the environment background and estimate that any set of points different from that background are vehicles. In contrast, including all classes forces the model to distinguish between vehicles and non-vehicles. The results show that the model has high precision rates, which indicate a low number of false positives, and thus, that the model has learnt to correctly distinguish between vehicles and non-vehicle objects (pedestrians, cyclists and other background objects such as road, walls, trees, etc.). However, the results in that chapter can not be used to draw conclusions regarding the individual detection performance of pedestrians and cyclists. Considering the detection of bicycle and pedestrian classes individually would allow studying how cooperative perception can improve the detection of these smaller objects, which are notoriously more difficult to detect when compared to larger objects, such as vehicles.

The final generalisation limitation regards the usage of simulated data. Most of the studies in this thesis used simulated data, with the exception of Chapter 7 where the KITTI odometry dataset was used. Simulated datasets provide insights into the deployment of cooperative perception methods that are useful before deploying such methods in the real world. For example, using simulated data allows for the evaluation of cooperative 3D object detection with various number and pose of sensors, which in practice would have prohibitive due to the time required to capture and label the data for each new sensor configuration being tested. However, there may be a performance gap in the performance of object detection methods between simulated and real-world

datasets due to several factors, *e.g.* the presence of more realistic noise in the sensor measurements, temporal misalignment, calibration errors, etc. The lidar noise model used in Chapter 5 attempts to recreate the first factor but more studies using real world data need to be carried out to evaluate the gap between simulated and real-world datasets. The usage of simulated data was chosen because no real world cooperative perception datasets were available at the time the studies in this thesis were being conducted. Due to the nature of cooperative perception, many sensors and vehicles are required to curate a real world dataset, which prevented the usage of a real dataset in this thesis.

8.2.2 Network Delay and Temporal Misalignment

The data from a set of sensors must be aligned in space (*i.e.* in the same coordinate system) and time (synchronised) before it can be fused. The studies in this thesis considered the spatial alignment of lidar frames in Chapter 5 through an assumed registration of static sensors and in Chapter 7 using an efficient point cloud registration method to obtain the relative pose between sensors. However, the aforementioned studies considered that all lidar frames were perfectly synchronised, *i.e.* were captured at the same moment in time. The temporal misalignment between lidar frames can happen for multiple reasons, including network delays and simply out-of-sync clocks between sensors. While the effect of temporal misalignment may be negligible for infrastructure sensors, *e.g.* in Chapter 5, due to the local connectivity between sensors and the central system, this is not the case for on-board sensors as studied in Chapter 7. Future studies should be conducted to evaluate the impact of temporal misalignment between lidar frames, particularly considering the usage of cooperative perception for on-board sensors.

8.2.3 Visibility Assumptions

Chapter 6 considered the visibility metric as the number of points cast on the surface of objects of interest. While this metric has shown correlation with the accuracy of the detected bounding box (in terms of IOU) in Section 5.4.5, it does not fully explain the object detection accuracy. This is due to the fact that this metric is agnostic to the distribution of the points on the objects' surfaces. For example, an object may have a large number of points in a small portion of their surface, *i.e.* most of its visible surface is on the side or back of the object, which provides little information about the object's dimensions. While this behaviour is seldom observed in the conducted experiments, future studies should consider analysing how the points are distributed across an objects' surface, rather than the number of visible points alone to determine the object's visibility. For example, the optimisation problem defined in Section

6.1 could be extended to guarantee that at least two distinct surfaces of each object are visible.

8.3 Future Work

This section suggests venues of future work building upon the limitations identified in the previous section as well as new research directions motivated by the findings of this thesis.

8.3.1 Creating a Large-Scale, Real-World Cooperative Perception Dataset

Curating a real-world dataset for cooperative perception would address most of the limitations regarding generalisation. First, collecting data at different types of driving environments, *e.g.* roundabouts, four-way junctions, but also multiple instances of each type of environment, *e.g.* small roundabouts, large roundabouts, pass-through roundabouts, *etc.*, would probe how cooperative perception performance gains scale with different environments. Second, annotating the data for multiple classes of objects, including vehicles, pedestrians and cyclists, would allow investigating if cooperative perception can improve the detection performance of smaller objects, which are notably more challenging to be detected. Third, comparing the results of real-world scenarios with simulated scenarios with similar characteristics would uncover the performance gap between simulated and real-world datasets. Finally, such a dataset could also be used to investigate the resilience of cooperative perception methods under challenging weather conditions, such as rain, snow and fog, which are difficult to be accurately simulated [23]. Such a dataset could drive further developments in the cooperative perception research area, much like how the KITTI [47] and nuScenes [54] motivated researchers to tackle different problems in the driving domain. Although creating such a large-scale dataset would require a joint effort, a small scale cooperative perception dataset including a single junction has been recently released [207]. Such a dataset could be used for preliminary investigations of cooperative perception using real-world data.

8.3.2 Improving Processing Efficiency

A practical cooperative perception pipeline must be able to perform inference, *i.e.* transform sensor data into detections, in real-time to be effective. The study in Chapter 5 showed that the latency of the detection model was in the order of 300ms, assuming fixed infrastructure sensor and negligible communication delay. This latency can be further reduced by using the latest generation of Graphics Processing Units (GPUs) and newer, more efficient 3D object

detection models, such as [78] which can perform inference within 38ms per frame using high-performing GPUs. When considering the fusion of on-board sensors, the relative pose estimation step needs to be added to the pipeline before the data fusion can happen. This introduces another source of latency into the system, which should be minimised. A new and computationally efficient registration method for point clouds with low-overlap was introduced in Chapter 7 and has shown to be 5 and 35 times faster than competing methods. Still, this registration method achieves inference times between 320 and 410ms, which would add a significant latency to the overall cooperative perception pipeline. Although the inference time is expected to decrease when evaluating the method on newer GPUs, future studies should consider how to further reduce the overall latency time of the cooperative perception pipeline. One venue of investigation is to reduce redundancy in the pipeline. For example, both the 3D object detection model and the point cloud registration model require performing feature extraction on the input point clouds, however, this is done separately for each model. Using a shared feature extraction model for both tasks would reduce the computational complexity of the overall pipeline. Further studies are required to determine how to adapt the respective feature extraction modules as to avoid task-specific performance drop in either tasks.

8.3.3 Investigating Communication Delay and Reducing Communication Load

The study in Chapter 7 assumed negligible communication delay between sensors, which is a valid assumption for fixed infrastructure sensors in a dedicated network. In future studies, the communication delay introduced by the wireless communication system for on-board, mobile sensors should also be investigated. One research direction could tackle how to minimise the communication load between sensors and the central processing unit. Sharing the raw point cloud points between agents and the central system introduces large communication costs as noted in Table 5.1. However, point cloud compression methods can be investigated to reduce the communication costs, while maintaining the 3D object detection performance. In this direction, learning-based methods [208–210] seem to be a promising venue of investigation. Another direction is to investigate methods leveraging the position of each sensor to partition the spatial domain of the driving environment. For example, the driving environment can be divided into non-overlapping segments using Voronoi diagrams such that each vehicle only shares a subset of its data [211].

8.3.4 Improving Key-Point Sampling in Point Cloud Registration

The latency and registration performance of the point cloud registration model in Chapter 7 could be improved by considering a more efficient key-point sampling strategy. That model uses Farthest Point Sampling [198], which considers the spatial arrangement of the observed points to sample points with uniform density, to subsample the point clouds and reduce the computational complexity of the model. However, this strategy is agnostic to the semantic information of each point, ignoring its point-wise features. On the other hand, a sampling mechanism that considers the point features could prevent sampling non-informative points such as ground points, points on moving objects or points in regions with low point density, which are unlikely to have correspondences in the other point cloud. A potential venue of investigation includes measuring a "saliency" map of the points' features [212]. This would identify points with higher likelihood of being unique across point clouds, and thus, increase the likelihood of identifying correct correspondences. In doing so, the number of sampled points can be reduced and the temporal performance of the RANSAC loop can be improved. Alternatively, a model could directly learn a probability function that estimates the likelihood of a point being a "good" correspondence, in the sense of being uniquely identifiable across point clouds. The final key-points can then be obtained by sampling the original points sampled based on this learned probability function.

8.3.5 Leveraging Other Sensor Modalities

The majority of studies in this thesis considered the usage of depth-capable sensors, namely lidar sensors, for cooperative 3D object detection. These sensors are still considerably more expensive than colour cameras, although their cost is expected to drop with advances in solid-state lidar technology [180]. The price factor and the fact that there is an existing infrastructure of CCTV cameras installed in many driving environments motivates the usage of colour cameras for cooperative 3D object detection. As noted in Section 2.3.1, such sensors do not provide depth cues which are key to determine accurate 3D position of the objects in the driving environment. One research direction is to investigate the potential of colour cameras for cooperative 3D object detection by leveraging depth estimation networks [60]. Such networks estimate a depth map for a given colour image, which can be back-projected into a point cloud and be used by lidar-based 3D object detection methods. The drawback of such methods is their limited depth accuracy and generalisation capabilities. However, leveraging multiple wide-baseline cameras into the depth estimation method could be investigated as means to improve the accuracy of

such methods. Another potential research direction is to investigate how to directly fuse perspective views from wide-baseline cameras. For example, by considering the projection of features from the perspective view (image plane) to a common bird-eye view, and then fusing the bird-eye-view features from multiple cameras [213].

8.3.6 Considering Cyber-Security Aspects

Cooperative perception systems deployed in real-world infrastructure must be able to identify and prevent malicious attacks. Future research should identify potential threats and how to mitigate them. One research direction includes investigating how to model trust in the data that each "agent" or sensor provides. For example, the trust of a certain agent can be modelled by verifying how much of the data it provides is in consensus with the data from other agents or sensors [214]. In this fashion, a vehicle broadcasting erroneous or malicious data can be identified and its data ignored, preventing malicious usage of the network.

Appendix A

Result Reproduction

In an effort to promote transparent and reproducible research, the source code, trained models and datasets of most studies in this thesis are made publicly available through open access platforms. The source code of each study is hosted in a respective Github repository, which also contains instructions on how to set up the software environment/dependencies, and the usage of the code. The datasets and pre-trained models are made available through open access research repositories: Warwick Research Archive Portal (WRAP) and Zenodo. The hyper-links to each resource are presented below.
Chapter 4:

- Source code: `https://github.com/eduardohenriquearnold/coopObjectClassification`.
- Dataset: `https://wrap.warwick.ac.uk/160228/`.

Chapter 5:

- Source code: `https://github.com/eduardohenriquearnold/coop-3dod-infra`.
- Dataset and pre-trained models: `https://wrap.warwick.ac.uk/159053/`.

Chapter 7:

- Source code and pre-trained models: `https://github.com/eduardohenriquearnold/fastreg`.
- Source code of the CODD dataset generation: `https://github.com/eduardohenriquearnold/codd`.
- Dataset [202]: `https://zenodo.org/record/5720317#.YZ4JudDP1EY`.

Note that the source code used to create the Cooperative Driving Dataset (CODD), described in Section 7.3.1, can be used to create a custom version of the dataset with different sensor modalities or number of vehicles/pedestrians.

Bibliography

[1] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019. doi: 10.1109/TITS.2019.2892405.

[2] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. Cooperative Object Classification for Driving Applications. In *IEEE Intelligent Vehicles Symposium*, pages 2484–2489, 2019. doi: 10.1109/IVS.2019.8813811.

[3] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative Perception for 3D Object Detection in Driving Scenarios Using Infrastructure Sensors. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2020. doi: 10.1109/TITS.2020.3028424.

[4] Eduardo Arnold, Sajjad Mozaffari, Mehrdad Dianati, and Paul Jennings. Visual Sensor Pose Optimisation Using Rendering-based Visibility Models for Robust Cooperative Perception. *Under review, IEEE Transactions on Systems, Man and Cybernetics: Systems*, 2021.

[5] Eduardo Arnold, Sajjad Mozaffari, and Mehrdad Dianati. Fast and Robust Registration of Partially Overlapping Point Clouds. *IEEE Robotics and Automation Letters*, 2021. doi: 10.1109/LRA.2021.3137888.

[6] Francesco Bellotti, Riccardo Berta, Ahmad Kobeissi, Nisrine Osman, Eduardo Arnold, Mehrdad Dianati, Ben Nagy, and Alessandro De Gloria. Designing an IoT Framework for Automated Driving Impact Analysis. In *IEEE Intelligent Vehicles Symposium*, pages 1111–1117, 2019. doi: 10.1109/IVS.2019.8813989.

[7] Francesco Bellotti, Nisrine Osman, Eduardo H. Arnold, Sajjad Mozaffari, Satu Innamaa, Tyron Louw, Guilhermina Torrao, Hendrik Weber, Johannes Hiller, Alessandro De Gloria, Mehrdad Dianati, and Riccardo Berta. Managing Big Data for Addressing Research Questions

in a Collaborative Project on Automated Driving Impact Assessment. *Sensors*, 20(23), 2020. ISSN 1424-8220. doi: 10.3390/s20236773. URL https://www.mdpi.com/1424-8220/20/23/6773.

[8] Johannes Hiller, Sami Koskinen, Riccardo Berta, Nisrine Osman, Ben Nagy, Francesco Bellotti, Ashfaqur Rahman, Erik Svanberg, Hendrik Weber, Eduardo H. Arnold, Mehrdad Dianati, and Alessandro De Gloria. The L3Pilot Data Management Toolchain for a Level 3 Vehicle Automation Pilot. *Electronics*, 9(5), 2020. ISSN 2079-9292. doi: 10.3390/electronics9050809. URL https://www.mdpi.com/2079-9292/9/5/809.

[9] Eduardo Arnold, Jiaxin Chen, Ivan Croydon-Veleslavov, Anurag Deshpande, Tanaya Guha, Yixuan He, Ben Moseley, Gim Seng Ng, Luke Prince, Thomas Statham, and Peter Strong. Alan Turing Data Study Group Final Report: Greenvest Solutions, February 2021. URL https://doi.org/10.5281/zenodo.4534349.

[10] Sajjad Mozaffari, Eduardo Arnold, Mehrdad Dianati, and Saber Fallah. A Comparative Study of Ego-centric and Cooperative Perception for Lane Change Prediction in Highway Driving Scenarios. In *Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems - ROBOVIS,*, pages 113–121. INSTICC, SciTePress, 2021. ISBN 978-989-758-537-1. doi: 10.5220/0010655700003061.

[11] Sajjad Mozaffari, Eduardo Arnold, Mehrdad Dianati, and Saber Fallah. Early Lane Change Prediction for Automated Driving Systems Using Multi-Task Attention-based Convolutional Neural Networks. *IEEE Transactions on Intelligent Vehicles*, 2021. doi: 10.1109/TIV.2022.3161785.

[12] SAE J3016C. J3016C: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Standard, SAE International, Apr 2021. URL https://www.sae.org/standards/content/j3016_202104/.

[13] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. *IEEE International Conference on Intelligent Robots and Systems*, 2018.

[14] Jongmin Jeong, Tae Sung Yoon, and Jin Bae Park. Towards a Meaningful 3D Map Using a 3D Lidar and a Camera. *Sensors*, 18(8), 2018. ISSN 1424-8220. doi: 10.3390/s18082571.

[15] Marcel Sheeny, Andrew Wallace, and Sen Wang. 300 GHz Radar Object

Recognition Based on Deep Neural Networks and Transfer Learning. *IET Radar, Sonar & Navigation*, 14(10):1483–1493, 2020.

[16] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, and Andrew M. Wallace. RADIATE: A Radar Dataset for Automotive Perception in Bad Weather. In *IEEE International Conference on Robotics and Automation*, pages 1–7, 2021. doi: 10.1109/ICRA48506.2021.9562089.

[17] M. Weber, P. Wolf, and J. M. Zöllner. DeepTLR: A Single Deep Convolutional Network for Detection and Classification of Traffic Lights. In *IEEE Intelligent Vehicles Symposium*, pages 342–348, June 2016. doi: 10.1109/IVS.2016.7535408.

[18] B. Ranft and C. Stiller. The Role of Machine Vision for Intelligent Vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):8–19, 2016.

[19] S. Sivaraman and M. M. Trivedi. Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14 (4):1773–1795, December 2013.

[20] Stephen Hsu, Sunil Acharya, Abbas Rafii, and Richard New. Performance of a Time-of-Flight Range Camera for Intelligent Vehicle Safety Applications. In *Advanced Microsystems for Automotive Applications 2006*, pages 205–219. Springer, 2006.

[21] Omar Elkhalili, Olaf M Schrey, Wiebke Ulfig, Werner Brockherde, Bedrich J Hosticka, P Mengel, and L Listl. A 64× 8 pixel 3-D CMOS Time of Flight Image Sensor for Car Safety Applications. In *2006 Proceedings of the 32nd European Solid-State Circuits Conference*, pages 568–571. IEEE, 2006.

[22] David Gallup, Jan-Michael Frahm, Philippos Mordohai, and Marc Pollefeys. Variable Baseline/resolution Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587671.

[23] Mario Bijelic, Tobias Gruber, and Werner Ritter. A Benchmark for Lidar Sensors in Fog: Is Detection Breaking Down? In *IEEE Intelligent Vehicles Symposium*, pages 760–767, 2018.

[24] Velodyne HDL-64E Lidar Specification, . URL `http://velodynelidar.com/hdl-64e.html`.

[25] Velodyne VLS-128 Announcement Article, . URL http://www.repairerdrivennews.com/2018/01/02/velodyne-leading-lidar-price-halved-new-high-res-product-to-improve-self-driving-cars/.

[26] Leddar Solid-State Lidar technology. URL https://leddartech.com/technology-fundamentals/.

[27] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[28] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[29] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks. *Conference on Neural Information Processing Systems*, 28:91–99, 2015.

[31] Berthold Klaus Paul Horn. Extended Gaussian Images. *Proceedings of the IEEE*, 72(12):1671–1686, 1984.

[32] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[33] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape Distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[34] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015.

[35] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *IEEE International Conference on Intelligent Robots and Systems*, pages 922–928, September 2015. doi: 10.1109/IROS.2015.7353481.

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 77–85, July 2017. doi: 10.1109/CVPR.2017.16.

[37] X. Chen, Y. Chen, and H. Najjaran. 3D Object Classification With Point Convolution Network. In *IEEE International Conference on Intelligent Robots and Systems*, September 2017. doi: 10.1109/IROS.2017.8202239.

[38] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and Multi-view CNNs for Object Classification on 3D Data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, June 2016. doi: 10.1109/CVPR.2016.609.

[39] David G Lowe. Object Recognition From Local Scale-Invariant Features. In *IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[40] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the Fisher Kernel for Large-Scale Image Classification. In *European Conference on Computer Vision*, pages 143–156. Springer, 2010.

[41] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning Methods for Generic Object Recognition With Invariance to Pose and Lighting. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–104, 2004.

[42] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *IEEE International Conference on Computer Vision*, pages 945–953, December 2015. doi: 10.1109/ICCV.2015.114.

[43] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[44] David Meger, Christian Wojek, James Little, and Bernt Schiele. Explicit Occlusion Reasoning for 3D Object Detection. In *British Machine Vision Conference*, pages 113.1–113.11, Dundee, 2011. British Machine Vision Association. ISBN 978-1-901725-43-8. doi: 10.5244/C.25.113.

[45] Ozgur Yilmaz. Classification Of Occluded Objects Using Fast Recurrent Processing. In *International Conference o Machine Learning and Applications*, pages 805–812, 2015.

[46] Benjamin Chandler and Ennio Mingolla. Mitigation of Effects of Occlusion on Object Recognition With Deep Neural Networks Through Low-Level Image Completion. *Computational intelligence and neuroscience*, 2016, 2016.

[47] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[48] Dominic Zeng Wang, I. Posner, and P. Newman. What Could Move? Finding Cars, Pedestrians and Bicyclists in 3D Laser Data. In *IEEE International Conference on Robotics and Automation*, pages 4038–4044, May 2012. doi: 10.1109/ICRA.2012.6224734.

[49] A. Azim and O. Aycard. Layer-based Supervised Classification of Moving Objects in Outdoor Dynamic Environment Using 3D Laser Scanner. In *IEEE Intelligent Vehicles Symposium*, pages 1408–1414, June 2014. doi: 10.1109/IVS.2014.6856558.

[50] Jens Behley, Volker Steinhage, and Armin B Cremers. Laser-based Segment Classification Using a Mixture of Bag-of-Words. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4195–4200, 2013.

[51] Ying Zhang, Mohammad Pezeshki, Philemon Brakel, Saizheng Zhang, César Laurent, Yoshua Bengio, and Aaron Courville. Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks. In *Interspeech 2016*, pages 410–414, 2016. doi: 10.21437/Interspeech.2016-1446. URL `http://dx.doi.org/10.21437/Interspeech.2016-1446`.

[52] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

[53] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[54] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. Nuscenes: A Multimodal Dataset for Autonomous Driving. In

IEEE Conference on Computer Vision and Pattern Recognition, pages 11621–11631, 2020.

[55] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset, 2020.

[56] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D Object Detection for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, June 2016. doi: 10.1109/CVPR.2016.236.

[57] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5632–5640, July 2017. doi: 10.1109/CVPR.2017.597.

[58] Y. Xiang, Wongun Choi, Y. Lin, and S. Savarese. Data-driven 3D Voxel Patterns for object category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1911, June 2015. doi: 10.1109/CVPR.2015.7298800.

[59] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1827–1836, July 2017. doi: 10.1109/CVPR.2017.198.

[60] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into Self-Supervised Monocular Depth Prediction. In *IEEE International Conference on Computer Vision*, October 2019.

[61] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-End Pseudo-LiDAR for Image-Based 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[62] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving. In *International Conference on Learning Representations*, 2020.

[63] Xinshuo Weng and Kris Kitani. Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud. In *IEEE International Conference on Computer Vision Workshops*, Oct 2019.

[64] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D Object Proposals for Accurate Object Class Detection. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Conference on Neural Information Processing Systems*, pages 424–432. Curran Associates, Inc., 2015.

[65] Cuong Cao Pham and Jae Wook Jeon. Robust Object Proposals Re-Ranking for Object Detection in Autonomous Driving Using Convolutional Neural Networks. *Signal Processing: Image Communication*, 53:110–122, April 2017. ISSN 0923-5965. doi: 10.1016/j.image.2017.02.007. URL `http://www.sciencedirect.com/science/article/pii/S0923596517300231`.

[66] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-Aware Convolutional Neural Networks for Object Proposals and Detection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 924–933, March 2017. doi: 10.1109/WACV.2017.108.

[67] Kevin S Chan. *Multiview Monocular Depth Estimation Using Unsupervised Learning Methods*. PhD thesis, Massachusetts Institute of Technology, 2018.

[68] Greire Payen de La Garanderie, Amir Atapour Abarghouei, and Toby P. Breckon. Eliminating the Blind Spot: Adapting 3D Object Detection and Monocular Depth Estimation to 360° Panoramic Imagery. In *European Conference on Computer Vision*, September 2018.

[69] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016. doi: 10.15607/RSS.2016.XII.042.

[70] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds. In *European Conference on Computer Vision Workshops*, September 2018.

[71] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar

3D Vehicle Detection. In *IEEE International Conference on Intelligent Transportation Systems*, pages 3266–3273, 2018.

[72] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia, and Arturo De La Escalera. BirdNet: a 3D Object Detection Framework from LiDAR information. In *IEEE International Conference on Intelligent Transportation Systems*, pages 3517–3523, 2018.

[73] B. Li. 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1513–1518, September 2017. doi: 10.1109/IROS.2017.8205955.

[74] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. In *IEEE International Conference on Robotics and Automation*, pages 1355–1361, May 2017. doi: 10.1109/ICRA.2017.7989161.

[75] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[76] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18, October 2018. doi: 10.3390/s18103337. URL `https://www.mdpi.com/1424-8220/18/10/3337`.

[77] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[78] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3DSSD: Point-based 3D Single Stage Object Detector. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[79] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10526–10535, 2020. doi: 10.1109/CVPR42600.2020.01054.

[80] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3DETR: Multi-Representation, Multi-Scale, Mutual-Relation 3D Object Detection With Transformers. In

IEEE Winter Conference on Applications of Computer Vision, pages 772–782, January 2022.

[81] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *IEEE International Conference on Computer Vision*, pages 945–953, December 2015. doi: 10.1109/ICCV.2015.114.

[82] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *IEEE International Conference on Robotics and Automation*, 2018.

[83] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR*, abs/1602.07360, 2016. URL `http://arxiv.org/abs/1602.07360`.

[84] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An Efficient Probabilistic 3d Object Detector for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12677–12686, 2019.

[85] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-Time 3d Object Detection From Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[86] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and ZhaoXiang Zhang. RangeDet: In Defense of Range View for LiDAR-Based 3D Object Detection. In *IEEE International Conference on Computer Vision*, October 2021.

[87] S. L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro. Vehicle Detection and Localization on Bird's Eye View Elevation Images Using Convolutional Neural Network. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 102–109, October 2017. doi: 10.1109/SSRR.2017.8088147.

[88] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525, July 2017. doi: 10.1109/CVPR.2017.690.

[89] Martin Simon, Karl Amende, Andrea Kraus, Jens Honer, Timo Samann, Hauke Kaulbersch, Stefan Milz, and Horst Michael Gross. Complexer-YOLO: Real-time 3D object detection and tracking on semantic point

clouds. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[90] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[91] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real Time End-to-End 3d Detection, Tracking and Motion Forecasting With a Single Convolutional Net. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[92] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-End Multi-View Fusion for 3d Object Detection in Lidar Point Clouds. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020.

[93] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast Encoders for Object Detection From Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[94] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots. In *European Conference on Computer Vision*, pages 68–84, 2020.

[95] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions On Graphics*, 38(5):1–12, 2019.

[96] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, June 2015.

[97] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Conference on Neural Information Processing Systems*, pages 5099–5108, 2017.

[98] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: Sparse-to-Dense 3D Object Detector for Point Cloud. In *IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.

[99] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Conference on Neural Information Processing Systems*, volume 30, 2017.

[100] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[101] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. *IEEE International Conference on Intelligent Robots and Systems*, 2018.

[102] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection From RGB-D Data. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[103] J. Schlosser, C. K. Chow, and Z. Kira. Fusing LIDAR and Images for Pedestrian Detection Using Convolutional Neural Networks. In *IEEE International Conference on Robotics and Automation*, pages 2198–2205, May 2016. doi: 10.1109/ICRA.2016.7487370.

[104] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-Task Multi-Sensor Fusion for 3d Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[105] Xinxin Du, Marcelo H. Ang, Sertac Karaman, and Daniela Rus. A General Pipeline for 3D Detection of Vehicles. In *IEEE International Conference on Robotics and Automation*, pages 3194–3200, 2018.

[106] K. Shin, Y. P. Kwon, and M. Tomizuka. RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement. In *IEEE Intelligent Vehicles Symposium*, pages 2510–2515, 2019. doi: 10.1109/IVS.2019.8813895.

[107] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative Perception for Connected Autonomous Vehicles based on 3D Point Clouds. In *IEEE International Conference on Distributed Computing Systems*, July 2019.

[108] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-Cooper: Feature based Cooperative Perception for Autonomous Vehicle Edge Computing System Using 3D Point Clouds. In *IEEE/ACM Symposium on Edge Computing*, November 2019.

[109] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction. In *European Conference on Computer Vision*, pages 605–621. Springer, 2020.

[110] Nicholas Vadivelu, Mengye Ren, James Tu, Jingkang Wang, and Raquel Urtasun. Learning to Communicate and Correct Pose Errors. In *Conference on Robot Learning*, 2020.

[111] Stefan Wender and Klaus C. J. Dietmayer. Extending Onboard Sensor Information by Wireless Communication. In *IEEE Intelligent Vehicles Symposium*, pages 535–540, 2007. doi: 10.1109/IVS.2007.4290170.

[112] S. Kim, Z. J. Chong, B. Qin, X. Shen, Z. Cheng, W. Liu, and M. H. Ang. Cooperative Perception for Autonomous Vehicle Control on the Road: Motivation and Experimental Results. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5059–5066, Nov 2013. doi: 10.1109/IROS.2013.6697088.

[113] S. W. Kim, W. Liu, M. H. Ang, S. W. Seo, and D. Rus. Cooperative Autonomous Driving Using Cooperative Perception and Mirror Neuron Inspired Intention Awareness. In *International Conference on Connected Vehicles and Expo*, pages 369–376, November 2014. doi: 10.1109/ICCVE.2014.7297573.

[114] S. W. Kim, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus. The Impact of Cooperative Perception on Decision Making and Planning of Autonomous Vehicles. *IEEE Intelligent Transportation Systems Magazine*, 7(3):39–50, 2015. ISSN 1939-1390. doi: 10.1109/MITS.2015.2409883.

[115] Gianluca Falco, Marco Pini, and Gianluca Marucco. Loose and Tight GNSS/INS Integrations: Comparison of Performance Assessed in Real Urban Scenarios. *Sensors*, 17(2):255, 2017.

[116] Kai-Wei Chiang, Thanh Trung Duong, and Jhen-Kai Liao. The Performance Analysis of a Real-Time Integrated INS/GPS Vehicle Navigation System with Abnormal GPS Measurement Elimination. *Sensors*, 13(8):10599–10622, 2013. ISSN 1424-8220. URL `https://www.mdpi.com/1424-8220/13/8/10599`.

[117] C. Shooter and J. Reeve. INTERSAFE-2 architecture and specification. In *IEEE International Conference on Intelligent Computer Communication and Processing*, pages 379–386, August 2009. doi: 10.1109/ICCP.2009.5284730.

[118] B. Rebsamen, T. Bandyopadhyay, T. Wongpiromsarn, S. Kim, Z. J. Chong, B. Qin, M. H. Ang, E. Frazzoli, and D. Rus. Utilizing the Infrastructure to Assist Autonomous Vehicles in a Mobility on Demand Context. In *TENCON 2012 IEEE Region 10 Conference*, pages 1–5, November 2012. doi: 10.1109/TENCON.2012.6412285.

[119] Michael Gabb, Holger Digel, Tobias Müller, and Rüdiger-Walter Henn. Infrastructure-supported Perception and Track-level Fusion using Edge Computing. In *IEEE Intelligent Vehicles Symposium*, pages 1739–1745, June 2019. doi: 10.1109/IVS.2019.8813886. ISSN: 2642-7214.

[120] Yicong Wang, Gustavo de Veciana, Takayuki Shimizu, and Hongsheng Lu. Deployment and Performance of Infrastructure to Assist Vehicular Collaborative Sensing. In *IEEE Conference on Vehicular Technology*, pages 1–5, 2018.

[121] N. Jayaweera, N. Rajatheva, and M. Latva-aho. Autonomous Driving without a Burden: View from Outside with Elevated LiDAR. In *IEEE Conference on Vehicular Technology*, pages 1–7, April 2019. doi: 10.1109/VTCSpring.2019.8746507.

[122] Joseph O'rourke. *Art Gallery Theorems and Algorithms*, volume 57. Oxford University Press Oxford, 1987.

[123] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic Camera Placement for Image-Based Modeling. In *Computer Graphics Forum*, volume 19, pages 101–110. Wiley Online Library, 2000.

[124] Pankaj K Agarwal, Esther Ezra, and Shashidhara K Ganjugunte. Efficient Sensor Placement for Surveillance Problems. In *International Conference on Distributed Computing in Sensor Systems*, pages 301–314. Springer, 2009.

[125] V. Akbarzadeh, C. Gagne, M. Parizeau, M. Argany, and M. A. Mostafavi. Probabilistic Sensing Model for Sensor Placement Optimization Based on Line-of-Sight Coverage. *IEEE Transactions on Instrumentation and Measurement*, 62(2):293–303, 2013.

[126] T. T. Nguyen, H. D. Thanh, L. Hoang Son, and V. Trong Le. Optimization for the Sensor Placement Problem in 3D Environments. In *IEEE International Conference on Networking, Sensing and Control*, pages 327–333, 2015. doi: 10.1109/ICNSC.2015.7116057.

[127] A. Saad, M. R. Senouci, and O. Benyattou. Toward a Realistic Approach for the Deployment of 3D Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, pages 1–1, 2020.

[128] Vahab Akbarzadeh, Julien-Charles Lévesque, Christian Gagné, and Marc Parizeau. Efficient Sensor Placement Optimization Using Gradient Descent and Probabilistic Coverage. *Sensors*, 14(8):15525–15552, 2014.

[129] S. Temel, N. Unaldi, and O. Kaynak. On Deployment of Wireless Sensors on 3-D Terrains to Maximize Sensing Coverage by Utilizing Cat Swarm Optimization With Wavelet Transform. *IEEE Transactions on Systems, Man, and Cybernetics*, 44(1):111–120, 2014. doi: 10.1109/TSMCC.2013.2258336.

[130] K. Chakrabarty, S. S. Iyengar, Hairong Qi, and Eungchun Cho. Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks. *IEEE Transactions on Computers*, 51(12):1448–1453, 2002. doi: 10.1109/TC.2002.1146711.

[131] Eva Hörster and Rainer Lienhart. On the Optimal Placement of Multiple Visual Sensors. In *ACM International Workshop on Video Surveillance and Sensor Networks*, pages 111–120, 2006.

[132] Jose-Joel Gonzalez-Barbosa, Teresa García-Ramírez, Joaquín Salas, Juan-Bautista Hurtado-Ramos, et al. Optimal Camera Placement for Total Coverage. In *IEEE International Conference on Robotics and Automation*, pages 844–848, 2009.

[133] Ali O Ercan, Danny B Yang, Abbas El Gamal, and Leonidas J Guibas. Optimal Placement and Selection of Camera Network Nodes for Target Localization. In *International Conference on Distributed Computing in Sensor Systems*, pages 389–404. Springer, 2006.

[134] Jian Zhao, Sen-Ching S Cheung, and Thinh Nguyen. *Optimal Visual Sensor Network Configuration*. Academic press, 2009.

[135] Jian Zhao, Ruriko Yoshida, Sen-ching Samson Cheung, and David Haws. Approximate Techniques in Solving Optimal Camera Placement Problems. *International Journal of Distributed Sensor Networks*, 9(11):241913, 2013.

[136] P. L. Chiu and F. Y. S. Lin. A Simulated Annealing Algorithm to Support the Sensor Placement for Target Location. In *Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 867–870 Vol.2, 2004.

[137] Yi Yao, Chung-Hao Chen, Besma Abidi, David Page, Andreas Koschan, and Mongi Abidi. Can You See Me Now? Sensor Positioning for Automated and Persistent Surveillance. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(1):101–115, 2009.

[138] Ce Li, Bing Lu, Yachao Zhang, Hao Liu, and Yanyun Qu. 3D Reconstruction of Indoor Scenes via Image Registration. In *Conference on Neural Information Processing Systems*, 2018.

[139] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019.

[140] Milad Ramezani, Georgi Tinchev, Egor Iuganov, and Maurice Fallon. Online LiDAR-SLAM for legged robots with robust registration and deep-learned loop closure. In *IEEE International Conference on Robotics and Automation*, pages 4158–4164, 2020.

[141] Brian C Hall et al. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 10. Springer, 2003.

[142] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 698–700, 1987.

[143] Carl Olsson, Fredrik Kahl, and Magnus Oskarsson. Branch-and-bound Methods for Euclidean Registration Problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):783–794, 2008.

[144] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3D Registration Efficiently and Globally Optimally. In *IEEE International Conference on Computer Vision*, pages 1457–1464, 2013.

[145] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan Registration for Autonomous Mining Vehicles Using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007. doi: https://doi.org/10.1002/rob.20204. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20204`.

[146] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: science and systems*, page 435. Seattle, WA, 2009.

[147] A. Makadia, A. Patterson, and K. Daniilidis. Fully Automatic Registration of 3D Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1297–1304, 2006. doi: 10.1109/CVPR.2006.122.

[148] Lukas Bernreiter, Lionel Ott, Juan Nieto, Roland Siegwart, and Cesar Cadena. PHASER: A Robust and Correspondence-Free Global Pointcloud Registration. *IEEE Robotics and Automation Letters*, 6(2):855–862, 2021.

[149] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. doi: 10.1109/ROBOT.2009.5152473.

[150] Martin A Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography. *ACM Communications*, 24(6):381–395, 1981.

[151] H. Yang, J. Shi, and L. Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Transactions on Robotics*, 2020.

[152] Laurent Kloeker, Christian Kotulla, and Lutz Eckstein. Real-Time Point Cloud Fusion of Multi-LiDAR Infrastructure Sensor Setups with Unknown Spatial Location and Orientation. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1–8, 2020.

[153] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNET: Global Context Aware Local Features for Robust 3d Point Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018.

[154] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully Convolutional Geometric Features. In *IEEE International Conference on Computer Vision*, pages 8958–8966, 2019.

[155] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep Global Registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2514–2523, 2020.

[156] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. PCRNet: Point Cloud Registration Network using PointNet Encoding. *arXiv preprint arXiv:1908.07906*, Aug 2019.

[157] Yue Wang and Justin M Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3523–3532, 2019.

[158] C. R. Maurer, G. B. Aboutanos, B. M. Dawant, R. J. Maciunas, and J. M. Fitzpatrick. Registration of 3-D Images Using Weighted Geometrical Features. *IEEE Transactions on Medical Imaging*, 15(6):836–849, 1996. doi: 10.1109/42.544501.

[159] Huanshu Wei, Zhijian Qiao, Zhe Liu, Chuanzhe Suo, Peng Yin, Yueling Shen, Haoang Li, and Hesheng Wang. End-to-End 3D Point Cloud

Learning for Registration Task Using Virtual Correspondences. In *IEEE International Conference on Intelligent Robots and Systems*, 2020.

[160] Alvaro Parra Bustos and Tat-Jun Chin. Guaranteed Outlier Removal for Point Cloud Registration With Correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2868–2882, 2017.

[161] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. Limo: Lidar-monocular Visual Odometry. In *IEEE International Conference on Intelligent Robots and Systems*, pages 7872–7879, 2018.

[162] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. An Online Multi-Robot SLAM System for 3D LiDARs. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1004–1011, 2017.

[163] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL `http://www.blender.org`.

[164] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. BlenSor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*, pages 199–208, 2011.

[165] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Conference on Robot Learning*, November 2017.

[166] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Conference on Neural Information Processing Systems*, pages 8026–8037, 2019.

[167] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501*, 2020.

[168] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[169] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Semantic Image Segmentation via Deep Parsing Network. In *IEEE International Conference on Computer Vision*, pages 1377–1385, 2015.

[170] David Opitz and Richard Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[171] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[172] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *IEEE International Conference on Robotics and Automation*, pages 1817–1824, 2011.

[173] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. ObjectNet3D: A Large Scale Database for 3D Object Recognition. In *European Conference on Computer Vision*, 2016.

[174] Xiaodan Jin and Keigo Hirakawa. Approximations to Camera Sensor Noise. In *Image Processing: Algorithms and Systems XI*, volume 8655, page 86550H. International Society for Optics and Photonics, 2013.

[175] Rui Yue, Hao Xu, Jianqing Wu, Renjuan Sun, and Changwei Yuan. Data Registration with Ground Points for Roadside LiDAR Sensors. *Remote Sensing*, 11(11):1354, 2019.

[176] M. Knorr, W. Niehsen, and C. Stiller. Online Extrinsic Multi-Camera Calibration Using Ground Plane Induced Homographies. In *IEEE Intelligent Vehicles Symposium*, pages 236–241, June 2013. doi: 10.1109/IVS.2013.6629476.

[177] Federico Castanedo. A Review of Data Fusion Techniques. *The Scientific World Journal*, 2013, 2013.

[178] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object Detection With Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009.

[179] A Stevens, Mehrdad Dianati, Konstantinos Katsaros, Chong Han, Saber Fallah, C Maple, F McCullough, and A Mouzakitis. Cooperative Automation Through the Cloud: The CARMA project. In *ITS European Congress*, 2017.

[180] Velodyne Velarray Announcement, . URL https://velodynelidar.com/newsroom/velodyne-displays-solid-state-highest-performing-lidar-for-adas/.

[181] Terry Collins. STREET LIGHTING INSTALLATIONS: For Lighting on New Residential Roads and Industrial Estates. Technical report, Durham County Council Neighbourhood Services, December 2014.

[182] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. doi: 10.1017/CBO9780511811685.

[183] Craig Glennie and Derek D Lichti. Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning. *Remote Sensing*, 2(6):1610–1624, 2010.

[184] Luis Enrique Ortiz, Elizabeth V Cabrera, and Luiz M Gonçalves. Depth Data Error Modeling of the ZED 3D Vision Sensor From Stereolabs. *ELCVIA: Electronic Letters on Computer Vision and Image Analysis*, 17(1):0001–15, 2018.

[185] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[186] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[187] Ivan Tomljenovic and Adam Rousell. Influence of Point Cloud Density on the Results of Automated Object-Based Building Extraction From ALS Data. In *AGILE International Conference on Geographic Information Science*, 2014.

[188] Karl Granström, Marcus Baum, and Stephan Reuter. Extended Object Tracking: Introduction, Overview, and Applications. *Journal of Advances in Information Fusion*, 12(2), 2017.

[189] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic Traffic Simulation using SUMO. In *IEEE International Conference on Intelligent Transportation Systems*, 2018.

[190] Lance Williams. Casting Curved Shadows on Curved Surfaces. *SIGGRAPH Computer Graphics*, 12(3):270–274, aug 1978. doi: 10.1145/965139.807402.

[191] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[192] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[193] Johnforrest, Stefan Vigerske, Haroldo Gambini Santos, Ted Ralphs, Lou Hafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, rlougee, jpgoncal1, h-i gassmann, and Matthew Saltzman. coin-or/Cbc: Version 2.10.5, March 2020. URL `https://doi.org/10.5281/zenodo.3700700`.

[194] Tulio A. M. Toffolo and Haroldo G. Santos. python-mip, July 2020. URL `https://python-mip.com/`.

[195] Ruihao Li, Sen Wang, and Dongbing Gu. DeepSLAM: A Robust Monocular SLAM System with Unsupervised Deep Learning. *IEEE Transactions on Industrial Electronics*, 2020.

[196] Patrik Schmuck and Margarita Chli. CCM-SLAM: Robust and Efficient Centralized Collaborative Monocular Simultaneous Localization and Mapping for Robotic Teams. *Journal of Field Robotics*, 36(4):763–781, 2019. doi: https://doi.org/10.1002/rob.21854.

[197] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration. In *IEEE International Conference on Computer Vision*, pages 12–21, 2019.

[198] Carsten Moenning and Neil A Dodgson. Fast Marching Farthest Point Sampling. Technical report, University of Cambridge, Computer Laboratory, 2003.

[199] Tian Xie and Jeffrey C. Grossman. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Physical Review Letters*, 120:145301, Apr 2018. doi: 10.1103/PhysRevLett.120.145301. URL `https://link.aps.org/doi/10.1103/PhysRevLett.120.145301`.

[200] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast Global Registration. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision*, pages 766–782, Cham, 2016.

[201] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2020.

[202] Eduardo Arnold. Cooperative Driving Dataset (CODD), 2021. Available at `https://github.com/eduardohenriquearnold/CODD`.

[203] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, April 2013. ISSN 1573-7527. doi: 10.1007/s10514-013-9327-2. URL `https://doi.org/10.1007/s10514-013-9327-2`.

[204] Matthias Fey and Jan E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *International Conference on Learning Representations Workshops*, 2019.

[205] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847*, 2018.

[206] Laurent Kloeker, Christian Geller, Amarin Kloeker, and Lutz Eckstein. High-Precision Digital Traffic Recording with Multi-LiDAR Infrastructure Sensor Setups. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1–8, 2020.

[207] Matthew Howe, Ian Reid, and Jamie Mackenzie. Weakly Supervised Training of Monocular 3D Object Detectors Using Wide Baseline Multi-view Traffic Camera Data. In *British Machine Vision Conference*, 2021.

[208] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[209] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2020.

[210] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep Compression for Dense Point Cloud Maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, April 2021. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2021.3059633. URL `https://ieeexplore.ieee.org/document/9354895/`.

[211] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y. Ethan Guo, Feng Qian, and Z. Morley Mao. EMP: Edge-Assisted Multi-Vehicle Perception. In *International Conference on Mobile Computing and Networking*, page 545–558. Association for Computing Machinery, 2021. ISBN 9781450383424. doi: 10.1145/3447993.3483242. URL `https://doi.org/10.1145/3447993.3483242`.

[212] Georgi Tinchev, Adrian Penate-Sanchez, and Maurice Fallon. SKD: Keypoint Detection for Point Clouds Using Saliency Estimation. *IEEE Robotics and Automation Letters*, 6(2):3785–3792, 2021. doi: 10.1109/LRA.2021.3065224.

[213] Li Haoran, Duan Zicheng, Ma Mingjun, Chen Yaran, Li Jiaqi, and Zhao Dongbin. MVM3Det: A Novel Method for Multi-view Monocular 3D Detection. In *IEEE International Conference on Robotics and Automation*, 2022.

[214] Braden Hurl, Robin Cohen, Krzysztof Czarnecki, and Steven Waslander. TruPercept: Trust Modelling for Autonomous Vehicle Cooperative Perception from Synthetic Data. In *IEEE Intelligent Vehicles Symposium*, pages 341–347, 2020. doi: 10.1109/IV47402.2020.9304695.