A Robust Object Detection System for Driverless Vehicles through Sensor Fusion and Artificial Intelligence Techniques

by

Esraa Alaaeldin Hassan Khatab

A thesis submitted in partial fulfilment for the requirements for the degree of Doctor of Philosophy at the University of Central Lancashire

November 2023

RESEARCH STUDENT DECLARATION FORM

Type of Award **Doctor of Philosophy**

School **School of Engineering and Computing**

1. Concurrent registration for two or more academic awards

I declare that while registered as a candidate for the research degree, I have not been a registered candidate or enrolled student for another award of the University or other academic or professional institution

2. Material submitted for another award

I declare that no material contained in the thesis has been used in any other submission for an academic award and is solely my own work

3. Collaboration

Where a candidate's research programme is part of a collaborative project, the thesis must indicate in addition clearly the candidate's individual contribution and the extent of the collaboration. Please state below:

N/A

4. Use of a Proof-reader

No proof-reading service was used in the compilation of this thesis.

Signature of Candidate 

Print name: ESRAA KHATAB

Acknowledgments

I would like to take this opportunity to express my deepest gratitude to the individuals who have supported and guided me throughout my doctoral journey, without whom this accomplishment would have not been possible.

First and foremost, I would like to extend my heartfelt appreciation to my esteemed supervisor, Prof Ahmed Onsy. I am particularly grateful to him for offering me the opportunity to pursue my Ph.D. degree under his supervision. His belief in my potential and his confidence in my abilities have been the driving force behind my academic success. Prof Onsy's unwavering support, invaluable guidance, and immense knowledge have played a pivotal role in shaping my research and enriching my academic experience. His dedication, patience, and insightful feedback have consistently pushed me to strive for excellence, and I am truly honoured to have had the privilege of working with him.

I would also like to express my sincere gratitude to my supervisor, Prof Martin Varley. His expertise, constructive criticism, and valuable suggestions have been instrumental in refining my research and broadening my understanding of the subject matter. I am grateful for his ongoing support and mentorship throughout my doctoral journey.

I am immensely grateful to the University of Central Lancashire for providing me with the opportunity to pursue my Ph.D. degree. The resources, facilities, and stimulating research environment have significantly contributed to the development of my research skills and the overall success of my study. I am particularly grateful for the flexibility offered during my research journey, allowing me to spend a year in the United Kingdom and then return to continue working in parallel in Egypt. This unique experience has broadened my perspectives and enhanced my understanding of cross-cultural research.

The path to completing my Ph.D. has not been without challenges, and the past couple of years, in particular, have been exceptionally demanding. The global pandemic disrupted our lives and introduced new obstacles that tested our resilience. I want to acknowledge the immense stress and uncertainty I faced during this time, but I am also grateful for the lessons it taught me about adaptability and perseverance. The support and understanding I received from my supervisors, family, and friends were instrumental in navigating these difficulties.

To my family, I owe a debt of gratitude that words alone cannot express. I would like to thank Omar, my beloved husband for his unwavering support, encouragement, and belief in my abilities. His constant reassurance and understanding during the long hours I spent engrossed in research were invaluable. I am deeply grateful to my children; Malik for his patience, understanding, and maturity beyond his age. He has shouldered a significant burden and managed to adapt to the challenges posed by my travels and stressful periods. His love and understanding have been a constant source of motivation and strength. Malika, her arrival brought immeasurable joy to my life and presented new challenges that required careful balancing between my academic pursuits and motherhood. Malika's presence has been a constant source of inspiration.

I would also like to express my appreciation to my mother, father, sisters, and my whole beloved family for their unyielding support, encouragement, and presence throughout this journey. Their belief in my abilities and their unwavering confidence in my success have been a driving force behind my achievements. Their sacrifices and understanding have enabled me to focus on my research and overcome various hurdles.

Finally, I would like to extend my thanks to all my friends and colleagues who have provided valuable insights, offered their assistance, and contributed to the intellectual discussions that have shaped my research. Your presence and support have been instrumental in shaping my academic and personal growth.

Thank you all.

Abstract

Since the early 1990s, various research domains have been concerned with the concept of autonomous driving, leading to the widespread implementation of numerous advanced driver assistance features. However, fully automated vehicles have not yet been introduced to the market. The process of autonomous driving can be outlined through the following stages: environment perception, ego-vehicle localization, trajectory estimation, path planning, and vehicle control. Environment perception is partially based on computer vision algorithms that can detect and track surrounding objects. The process of objects detection performed by autonomous vehicles is considered challenging for several reasons, such as the presence of multiple dynamic objects in the same scene, interaction between objects, real-time speed requirements, and the presence of diverse weather conditions (e.g., rain, snow, fog, etc.). Although many studies have been conducted on objects detection performed by autonomous vehicles, it remains a challenging task, and improving the performance of object detection in diverse driving scenes is an ongoing field. This thesis aims to develop novel methods for the detection and 3D localization of surrounding dynamic objects in driving scenes in different rainy weather conditions.

In this thesis, firstly, owing to the frequent occurrence of rain and its negative effect on the performance of objects detection operation, a real-time lightweight deraining network is proposed; it works on single real-time images separately. Rain streaks and the accumulation of rain streaks introduce distinct visual degradation effects to captured images. The proposed deraining network effectively removes both rain streaks and accumulated rain streaks from images. It makes use of the progressive operation of two main stages: rain streaks removal and rain streaks accumulation removal. The rain streaks removal stage is based on a Residual Network (ResNet) to maintain real-time performance and avoid adding to the computational complexity.

Furthermore, the application of recursive computations involves the sharing of network parameters. Meanwhile, distant rain streaks accumulate and induce a distortion similar to fogging. Thus, it could be mitigated in a way similar to defogging. This stage relies on a transmission-guided lightweight network (TGL-Net). The proposed deraining network was evaluated on five datasets having synthetic rain of different properties and two other datasets with real rainy scenes.

Secondly, an emphasis has been put on proposing a novel sensory system that achieves real-time multiple dynamic objects detection in driving scenes. The proposed sensory system utilizes a monocular camera and a 2D Light Detection and Ranging (LiDAR) sensor in a complementary fusion approach. YOLOv3- a baseline real-time object detection algorithm- has been used to detect and classify objects in images captured by the camera; detected objects are surrounded by bounding boxes to localize them within the frames. Since objects present in a driving scene are dynamic and usually occluding each other, an algorithm has been developed to differentiate objects whose bounding boxes are overlapping. Moreover, the locations of bounding boxes within frames (in pixels) are converted into real-world angular coordinates. A 2D LiDAR was used to obtain depth measurements while maintaining low computational requirements in order to save resources for other autonomous driving-related operations. A novel technique has been developed and tested for processing and mapping 2D LiDAR measurements with corresponding bounding boxes. The detection accuracy of the proposed system was manually evaluated in different real-time scenarios. Finally, the effectiveness of the proposed deraining network was validated in terms of its impact on objects detection in the context of de-rained images.

Results of the proposed deraining network were compared to existing baseline deraining networks and have shown that the running time of the proposed network is 2.23× faster than the average running time of baseline deraining networks while achieving 1.2× improvement when tested on different synthetic datasets. Moreover, tests on the LiDAR measurements showed an average error of ±0.04m in real driving scenes. Also, both deraining and objects detection are jointly tested, and it was demonstrated that performing deraining ahead of objects detection caused 1.45× enhancement in the object detection precision.

List of Abbreviations

ADAS	Advanced Driver Assistance Systems
AEB	Automatic Emergency Braking
AI	Artificial Intelligence
AOS	Average Orientation Similarity
AP	Average Precision
AV	Autonomous Vehicle
BEV	Bird's Eye View
CAD	Computer-Aided Designs
CAN	Controller Area Network
CNN	Convolutional Neural Network
DARPA	Defence Advanced Research Projects Agency
DR	Dead Reckoning
DL	Deep Learning
DM	Driver Monitoring
DNN	Deep Neural Network
ELU	Exponential Linear Unit
FN	False Negative
FoV	Field of View
FP	False Positive
FPS	Frame Per Second
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
HFOV	Horizontal Field of View
HIN	Half Instance Normalization
HOG	Histogram of Oriented Gradients
HVS	Human Visual System
IMU	Inertial Measurement Unit
IoT	Internet of Things
IoU	Intersection over Union
IQA	Images Quality Assessment
LiDAR	Light Detection and Ranging
LKA	Lane Keep Assist
mAP	mean Average Precision
MCA	Morphological Component Analysis
ML	Machine Learning

MSE	Mean Square Error
MTBF	Mean Time Before Failure
MVG	Multivariate Gaussian
NIQE	Naturalness Image Quality Evaluator
NSS	Natural Scene Statistics
OA	Overall Accuracy
ODD	Operational Driving Domain
PA	Park Assist
PPV	Positive Predictive Values
PRN	Progressive Residual Network
PSNR	Peak Signal to Noise Ratio
RADAR	Radio Detection and Ranging
ResNet	Residual Network
RGB	Red, Green, Blue
RGBD	Red, Green, Blue, Depth
ReLU	Rectified Linear Unit
ROI	Region of Interest
SAE	Society of Automotive Engineers
SAM	Supervised Attention Module
SIFT	Scale-Invariant Feature Transform
SOTA	State Of The Art
SSD	Single-Shot multi-box Detector
SSEQ	Spatial-Spectral Entropy-based Quality
SSIM	Structural Similarity Index Measure
SVM	Support Vector Machine
TJA	Traffic Jam Assist
TN	True Negative
ToF	Time of Flight
TP	True Positive
YOLO	You Only Look Once

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Overview and Problem Statement

Autonomous driving, also known as self-driving or driverless vehicles, is considered a current trend in research and development. Autonomous driving refers to the technology and systems that allow vehicles to operate and navigate without human intervention. Major research centres and automotive companies are contributing to the boosting of the performance of autonomous driving on a regular basis. The Society of Automotive Engineers (SAE) has ranked autonomous driving in 6 levels, ranging from Level 0 (no automation) to Level 5 (automation under any circumstances) [1]. Level 5 is still under research and not available publicly.

Broggi et al. [2] have examined different tests performed by AVs in different scenarios with roads of different traffic intensities. The test Autonomous Vehicle (AV) was called PROUD (Public Road Urban Driverless), which was considered a breakthrough in autonomous car technology. Moreover, Jo et al. [3] targeted the development of AVs that are generalized for any driving environment. They used FlexRay as a software platform and communication protocol instead of traditional Controller Area Network (CAN) units. FlexRay is faster and more reliable compared to CAN technology. Some of the technologies that are applied in AVs are Lane Keep Assist (LKA), Park Assist (PA), Automatic Emergency Braking (AEB), Driver Monitoring (DM), Traffic Jam Assist (TJA), Dead reckoning (DR). Further technologies are under continuous development in order to move towards higher automation levels, such as:

- Sensors' development and optimization
- Stretching Internet of Things (IoT) towards Internet of Vehicles (IoV), where the vehicles themselves gather traffic information, weather conditions, parking spaces, etc.
- Enhancing and integrating different Artificial Intelligence (AI) techniques with different complex driving operations, such as objects detection, path planning and optimization, decision-making
- Advancing AV-based cyber security to protect autonomous driving systems from being hacked by malicious parties.

Figure 1.1 illustrates the main modules constituting the autonomous driving system [4]. Different sensors are mounted on AVs to gather data about the surrounding environment. They create a detailed picture of the environment, allowing the vehicle to identify different objects, such as pedestrians, road markings, and other vehicles. Different categories of sensors, their advantages, limitations, and applications will be discussed in Chapter 2. Afterwards, data gathered by the sensors is processed in the perception block, which processes data into useful, meaningful information. Advanced AI algorithms process the gathered sensory data to understand the environment, make decisions, and plan optimal routes.

Moreover, the planning module uses the perception module's output to perform short- and long-path planning. High-definition maps are used to enhance the vehicle's understanding of its location and surroundings; this helps the vehicle navigate accurately and safely. Finally, the control module ensures the AV follows the path provided by the planning module by controlling different vehicle parts. For the aforementioned hierarchy of operations, the process of environment perception and object detection should be accurate and reliable, even in variable operational domains; in other words, a step forward towards Level 5 autonomy.

In order to advance towards Level 5 AVs, they must achieve robust and reliable performance in different operational circumstances, most importantly, adverse weather conditions. The presence of rain, snow, and fog greatly degrades the quality of data gathered by the sensory systems of AVs. These weather conditions degrade scene contrast and visibility, which causes deterioration in the ability of the vehicle to detect surrounding objects. Adverse weather conditions could be classified into steady (fog, mist, and haze) and dynamic (rain and snow) [5]. Dynamic adverse weather conditions are more challenging to mitigate as they don't affect the whole image equally. Moreover, they come in different directions, orientations, and intensities. For example, rain consists of countless drops of diverse sizes and shapes [6].

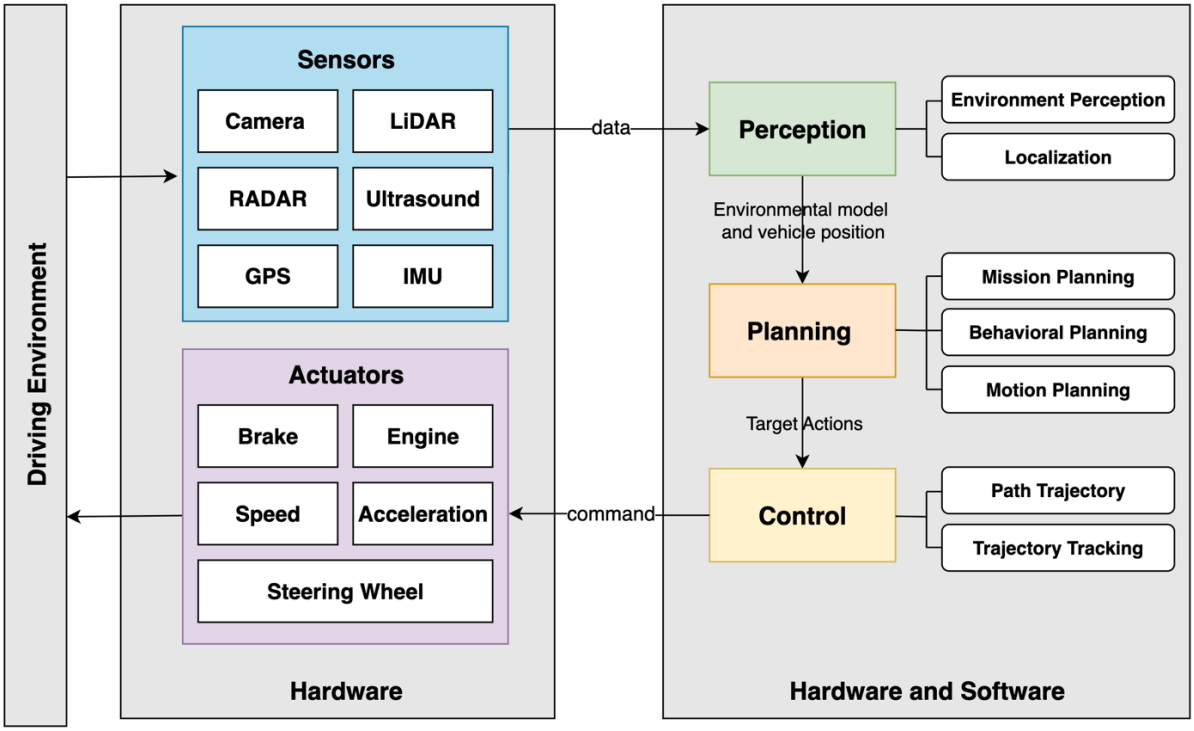

Figure 1.1: Block diagram of autonomous driving [7]

1.2 Motivation

Current challenges in AV development are summarized in Figure 1.2. Also, as shown in Figure 1.1, environment perception comes ahead of the subsequent AV operations (i.e., planning and control) [7]. For AVs, data are collected by multiple sensors; after processing these data, different features of objects in the surrounding environment are extracted. Both static and dynamic objects are detected and then tracked. Moreover, vehicle behaviour and scene understanding are performed. The main functions of environment perception are based on lane and road detection, traffic sign recognition, dynamic objects detection and tracking, and scene understanding [8]. Therefore, objects detection is an essential operation as it affects many subsequent ones [8].

Most available sensory systems deployed on AVs are high-cost sensors that capture a considerable amount of data. The high cost of sensors hinders further research on the development of enhanced perception systems. Moreover, although the acquisition of huge amounts of perceived data enhances the surrounding understanding, it increases the computational complexity and running time of the system.

Also, driving environments are considered dynamic and surrounded by different road elements that are in continuous movement and interact with each other. Dynamic road elements include pedestrians, cyclists, and other vehicles. They interact with each other, causing occlusion and truncations of their detections captured by the camera. These interacting objects burden the integration of multi-sensory data.

Level 5 AVs are meant to operate under diverse circumstances, such as slippery roads, road work, accidents, and diverse weather conditions. Much research has addressed different environment perception algorithms performed by AVs; on the other hand, other research has aimed to mitigate the effect of rain on visual data. However, some aspects have not been addressed, such as (*1*) addressing the deraining task in single images that lack temporal data, (*2*) real-time constraints that AVs should follow, and (*3*) different rain degradation effects.

Additionally, numerous deraining algorithms have led to a decline in the subsequent object detection performance [9, 10]. Hence, it becomes crucial to assess deraining techniques in conjunction with the performance of object detection.

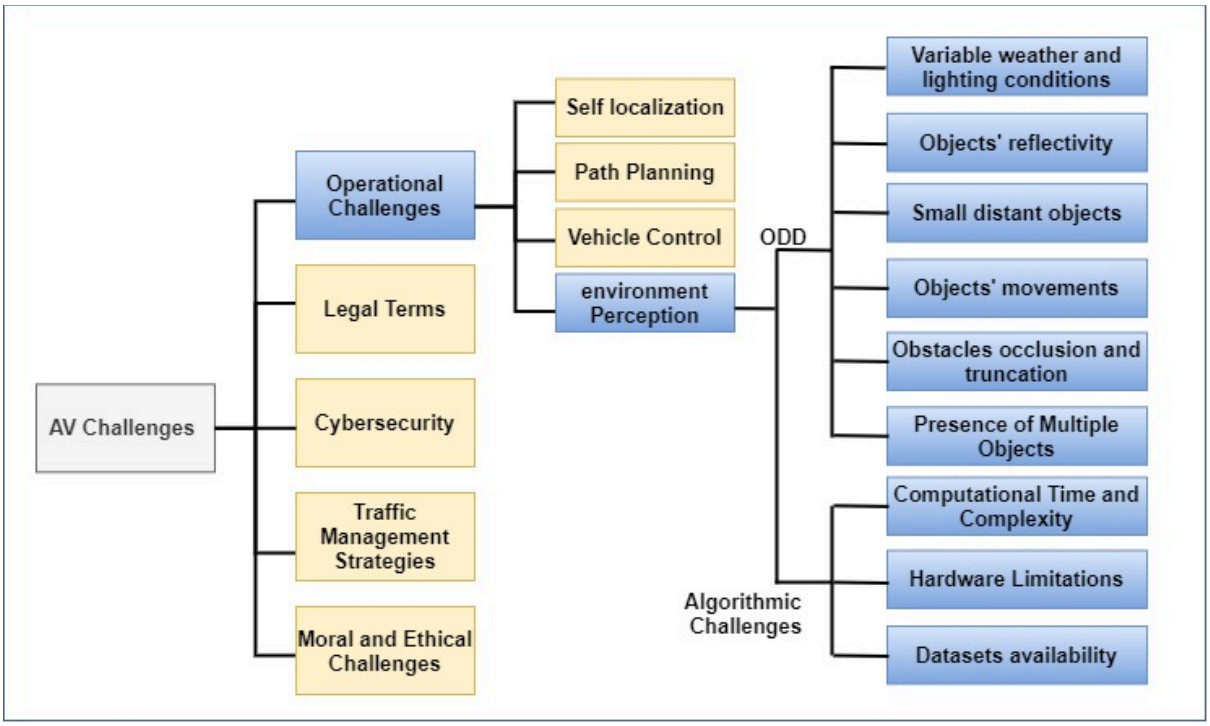

Figure 1.2: Taxonomy of challenges facing the widespread and the operation of AVs

1.3 Thesis Objectives and Contribution

This study has two main objectives. The first objective is to investigate the feasibility of employing a low-cost 2D Light Detection and Ranging sensor (LiDAR) in addition to a typical monocular camera to achieve real-time 3D object detection of multiple dynamic road elements. The second objective is to boost the object detection operation's performance to enable it to perform under different rainy weathers.

Therefore, the main contribution of this thesis is an object detection framework that remedies previous methods' shortcomings to produce a bespoke solution for AVs object detection tasks. The main challenges that were addressed are (*1*) maintaining real-time performance, (*2*) computational complexity, (*3*) the presence of multiple dynamic objects surrounding AVs, (*4*) the high cost of the sensory systems, and (*5*) rainy weather conditions with different degradation effects.

The main contributions of this thesis can be summarized as follows:

- Adapting different state-of-the-art deraining and denoising methods to successfully perform deraining on both synthetic and real-world rainy datasets (Chapter 3). Based on the available research, no work has been addressed before to mitigate different degradation effects in real time.
- Proposing a lightweight network for real-time rain streaks and rain accumulation removal from images captured by AVs (Chapter 3). The proposed network works upon single images and adopts a multi-stage operation. Additionally, it has been tested on different synthetic and real-world rainy datasets.
- Evaluating the running time of the developed network and comparing it to other significant deraining networks.
- Developing a 3D object detection framework that employs complementary sensor fusion between a 2D LiDAR and a monocular camera (Chapter 4). YOLOv3 object-detection framework has been applied to captured images to perform robust object detection.
- Both separate and interacting objects are detected and analysed by the video processing module. Afterwards, LiDAR measurements were processed to complement the detected objects with a depth dimension. Therefore, the output of the object detection framework is bounding boxes of detected objects with associated

depth measurements. Previous methods used 3D sensors (3D LiDARs or stereo cameras), which impose higher cost and computational requirements, or were not targeted for detecting multiple overlapping objects.

- Validation of the proposed model regarding the performance of object detection task by applying object detection algorithm (YOLOv3) over the de-rained images that are the output of the deraining framework. Moreover, deraining results from other deraining networks were also tested for object detection to be compared with the proposed model.

The current research has resulted in three journal papers listed in Appendix A.

1.4 Thesis Structure

The overall thesis layout is as follows:

Chapter 2 describes the specifications of autonomous driving along with its different technologies and challenges. Moreover, different sensory systems and environment perception approaches are elaborated and surveyed. The focus will be on dynamic object detection. In addition, different AI-based object detection techniques will be elaborated. Finally, the different degradation effects of rainy weather on captured images will be exploited while reviewing baseline deraining networks.

Chapter 3 presents the effect that diverse rainy conditions impose on captured images from driving scenarios. A novel deraining network is further developed to perform single image deraining in real time.

Chapter 4 investigates a low-cost framework that performs real-time multiple dynamic objects detection. The prototype includes a monocular camera along with a 2D LiDAR to perform the detection of numerous dynamic objects in 3D within real-world driving situations. The video-processing module outputs bounding boxes of detected objects, while LiDAR measurements complement these bounding boxes with a depth dimension. The system has the capability to accurately localize multiple objects even if they are occluding each other.

Chapter 5 presents the experimentation details of the proposed approaches. Datasets used to evaluate of the proposed methods will be described, followed by the evaluation protocols for

object detection as well as deraining operations performed by AVs, along with ablation studies and experimental results. Proposed approaches are compared to current baseline ones; multiple experiments have been performed to evaluate the proposed deraining network's performance over both synthetic and real rainy datasets. Additionally, its running time has been compared to baseline deraining algorithms. Finally, object detection has been performed on de-rained images to further asses the joint deraining and object detection operations.

Chapter 6 concludes and summarizes the thesis. Additionally, the chapter provides recommendations for future work.

Furthermore, a list of appendices is provided, which includes the following:

- Appendix A: List of Publications
- Appendix B: Hokuyo UTM-30LX Specifications
- Appendix C: Vehicles' Ground Clearance
- Appendix D: Code Samples for LiDAR Measurements
- Appendix E: Code Samples for LiDAR and Camera Integration
- Appendix F: Auxiliary Material

And, finally, the list of references is presented.

Chapter 2 Literature Review

2.1 Introduction

The process of object detection undertaken by AVs holds significant importance as it precedes various autonomous driving operations such as object tracking, trajectory estimation, and collision avoidance. The presence of dynamic road elements (pedestrians, cyclists, vehicles) adds complexity due to their constantly changing positions and behaviours. This chapter provides an all-encompassing examination of state-of-the-art object detection technologies, concentrating on the sensory systems and algorithms employed.

First, a concise introduction to the operations and challenges inherent in autonomous driving is presented. Then, diverse sensory systems utilized on existing AVs are elaborated, discussing their advantages, constraints, and applications. Furthermore, the sensory systems used in various research endeavours are examined. Moreover, given the notable impact that AI has on objects detection tasks, different DNN-based networks are also elaborated, along with related research on this topic. Finally, due to the significant impact imposed by rainy weather on the object detection operation, different rain degradations are discussed along with the existing baseline deraining algorithms.

2.2 Autonomous Driving

AVs, commonly referred to as self-driving or driverless vehicles, have the potential to revolutionize the transportation pattern in the world. One of the most prominent advantages of autonomous driving lies in its ability to alleviate the demands of the driving task itself. Autonomous driving extends mobility access to a broader demographic, grants passengers additional free time during their journeys (the average driver in England spends 235 hours driving every year), reduces emissions, alleviates traffic congestion, and, most significantly, enhances road safety [11]. Since the early 1990s, autonomous driving has attracted various research domains, paving the way for several highly advanced driver assistance functionalities that have now reached widespread adoption.

Statistics related to road accidents reveal that a significant 76% of all accidents stem exclusively from human error, with a considerable 94% having an element of human error in their occurrence, according to the National Highway [11]. Among the primary reasons

behind road collisions are distractions, haste, and misinterpretation of other road users' actions [12].

The process of autonomous driving can be outlined in the following stages [8, 12, 13]:

- Perceiving the environment by the detection of roads, signs, pedestrians, cyclists, other vehicles, obstacles, etc., and understanding their motion
- Determining the location of the detected objects
- Localization of the ego-vehicle itself
- Path planning and trajectory estimation
- Giving the appropriate controlling commands to the actuators

2.2.1 SAE Classification

Given the variations in terminologies used to depict autonomous driving, the Society of Automotive Engineers (SAE) has established a classification system that ranks vehicle automation and intelligence levels. [14]. It provides a taxonomy of six levels of driving automation (level 0 – level 5). Defining an automation level is related to the driving automation features that can be engaged in a driving scenario at any instance. The six SAE levels are shown in Figure 2.1 and summarized below:

- Level 0 (No Automation): The human driver is responsible for all driving aspects, including control and monitoring.
- Level 1 (Driver Assistance): The vehicle can assist with either longitudinal or lateral control but not both simultaneously; an example is adaptive cruise control.
- Level 2 (Partial Automation): The vehicle can control both longitudinal and lateral control simultaneously; however, the driver must remain engaged and monitor the environment.
- Level 3 (Conditional Automation): The vehicle can perform object and event detection and response, while the human override is mandatory whenever the vehicle is unable to execute a particular task.
- Level 4 (High Automation): The vehicle is entirely autonomous in the vast majority of situations, but there is still an option to switch to manual driving under challenging scenarios.

- Level 5 (Full Automation): The vehicle is fully automated under any Operational Driving Domains (ODDs), and the vehicle does not feature any typical driving controls (e.g., steering wheels, brake pedals, etc.).

For an easier illustration, the UK's Centre for Connected and Autonomous Vehicles described levels 0-2 as "Hands-on Assisted Driving," level 3 as "Hands off, Eyes on (the road)," and levels 4 and 5 as "Hands off, Eyes off" [15].

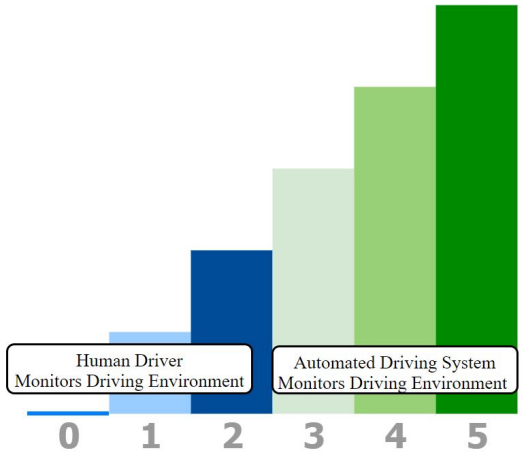

Figure 2.1: SAE J3016 automotive automation standard [14]

2.2.2 Competitions Promoting Autonomous Driving

Numerous competitions have been organized to stimulate advancements in the autonomous driving sector. The first was the European PROMETHEUS project, which featured a route from Munich to Odense in 1995 [16-18]. Furthermore, a particularly notable competition was organized by the Defence Advanced Research Projects Agency (DARPA) [19-21]. A brief overview of autonomous vehicle initiatives and competitions can be found in [13]. The most recent ongoing challenge is the AutoDrive challenge, co-sponsored by the SAE and General Motors (GM) [22]. This challenge is geared towards achieving navigation within an urban driving environment, following the SAE standards description of level 4 automation (J3016) [14].

2.2.3 Towards Autonomous Driving in Industry

Parallel to the academic field, automobile manufacturers have embraced the advancement of autonomous vehicles [23]. As a result, significant technological progress has been witnessed in this research arena. [24-27]. Many automotive companies have focused on enhancing the

autonomous capabilities of their vehicles. Commencing in 2013, Volvo embarked on the development of the "Drive Me" project, successfully creating Level 3 AVs. In 2017, Volvo initiated a large-scale trial of autonomous driving involving real-road experiences by customers. Similarly, in 2017, Audi introduced a Level 3 AV, becoming the world's first production car crafted specifically for conditional automated driving at Level 3. The highest autonomy level AVs are developed by Waymo, which not only develops the highest-level autonomous vehicles but also manufactures a suite of self-driving hardware. On the other hand, Tesla's new cars are equipped with hardware capable of facilitating autopilot functionalities and gradually introducing self-driving features through software updates, positioning them between levels 2 and 3 of autonomy. Notably, it's important to highlight that most autonomous vehicle manufacturers employ the same types of sensors, except for Tesla, which abstains from using a LiDAR.

While level 5 autonomous vehicles are not yet available in the commercial market, all manufacturers strive to achieve this objective. Notably, they are using similar sensory categories, except for Tesla, which opts not to utilize LiDAR technology as the manufacturers see it as a high-cost system that could be replaced by vision-based systems. Table 2.1 provides an overview of the concept cars manufacturers are targeting for production in the upcoming years.

Table 2.1: Levels 4 and 5 concept cars

Level 4	Level 5
• Symbioz	• Mercedes Benz S-class
• Yandex Taxi	• VW Sedric
• Volvo 360c	• Rinspeed Snap
• Ford fusion	• Rinspeed Oasis
• Rolls Royce 103EX	• Rinspeed Microsnap
• Chrysler Pacificas	• Rinspeed ∑tos
• Toyota Edge	

2.2.4 Challenges Facing the Spread of Autonomous Driving

Despite the significant progress within the realm of the autonomous driving field, numerous concerns impede its widespread adoption [28, 29] :

- *Legal terms*: They range from the need for a special driving license to much more obscure topics such as responsibility in the event of an accident. Legal terms must consider public policies, traffic codes, and technical standards.
- *Cybersecurity*: Autonomous driving is a complex "system-of-systems" that requires a comprehensive and shared framework. The UK Department for Transport established key cybersecurity principles for Connected and Automated Vehicles. Also, the SAE developed a cybersecurity Guidebook for Cyber-Physical Vehicle systems [30]
- *Moral and ethical challenges*: Driving scenarios are considered diverse and may include moral dilemmas as well as basic driving rules. AVs must make rapid decisions in these diverse conditions; therefore, moral algorithms must be established in order to solve such dilemmas according to acceptable norms.
- *Traffic management strategies*: To enable AVs to interact in real road scenarios effectively, they must have with a high-precision digital map. Also, recent approaches believe that AVs should be cooperative; Vehicle-to-Vehicle communication is encouraged in order to effectively share critical information between vehicles (autonomous or traditional), making use of the advanced sensors mounted on AVs.

2.3 Sensory Systems

The initial stage in autonomous driving involves environment perception through suitable sensory systems. These sensory systems can be categorized into three primary categories [31], as shown in Table 2.2.

Table 2.2: Automotive sensing categories

Sensory system	Definition	Sensors
Self-seeing	Estimating the current state of the ego-vehicle: its velocity, yaw angle, acceleration, and steering angle, using proprioceptive sensors and the Controller Area Network	Proprioceptive sensors: • Odometers • IMUs • Gyroscopes
Localization	Estimating the current location of the vehicle	External sensors: GNSS Dead reckoning: IMUs
Environment Perception	Perceiving external data such as road markings, obstacles, traffic signs, etc.	Exteroceptive sensors: • Camera • RADAR • LiDAR • Ultrasonic sensors

Sensors can also be classified into active and passive [32, 33]. Active sensors transmit electromagnetic waves and then measure the return signal reflected to them. Examples of such sensors include Ultrasonic, Radio Detection and Ranging (RADAR), and LiDAR. On the other hand, passive sensors rely on detecting existing electromagnetic waves in the environment without emitting their own signals. Infrared- and light-based cameras are examples of passive sensors [34-45], as well as acoustic systems [46-48].

2.3.1 Exteroceptive Sensors

As outlined in Table 2.2, exteroceptive sensors play a vital role in sensing and perceiving the environment. Table 2.3 concisely summarises the prevalent exteroceptive sensors utilized in autonomous driving, including their advantages, limitations, and applications. This subsection elaborates their integration into various AV operations, both as standalone components and when combined.

i. Camera

There are two main camera categories that can be used on outdoor AVs:

- Monocular cameras: They are the primary camera sensors used for object detection in [44]. Also, in [49], multiple monocular cameras were used in multi-object tracking. Various algorithms have been developed to perform object detection and localization [50, 51]; however, the results suffered from relatively low accuracy in depth estimation especially at longer ranges.

- Stereo-cameras: They have been used in [52] to perform road surface condition monitoring. Also as an extension to the work proposed in [51], Chen et al. have proposed 3D object proposals using stereo imagery in [53]. Moreover, in [54, 55], authors have achieved 3D object detection in RGB-D images, demonstrating the great potential of 3D deep learning to perform 3D detections. Another significant boost to the stereo-vision applications was the work proposed in [56, 57], where authors transformed the data obtained from a stereo-camera into a 3D point cloud, emulating the 3D point cloud typically generated by a 3D LiDAR. This approach facilitated the application of various pre-existing LiDAR-based detection algorithms.

Even though cameras provide feature details of surrounding environments, monocular cameras suffer from a lack of direct and accurate depth measurements. On the other hand, stereo-cameras can perform depth estimation via the processing of two separate images. However, stereo-imagery requires high on-board computational resources for computer vision processing; also, camera pairs may become un-calibrated during driving and may need manual adjustment.

Table 2.3: Sensors, advantages, limitations, applications

Sensor	Advantages	Limitation	Application
Monocular camera	• Low cost • Different Fields of View (FoV) • High-resolution cameras provide a more extended range • Provide features data	• High computational requirements • It does not provide straightforward distance calculations • Limited by weather and lighting conditions • Can't calculate the velocity of objects	• Object detection & classification • Traffic signals recognition • Road and lane detection
Stereo-camera	In addition to the advantages of monocular cameras, stereo-cameras also provide: • Depth calculation • 3D-localization of objects • Enhanced object detection	• More expensive than monocular cameras • Higher computational requirements • Limited by weather and lighting conditions • Can't calculate the velocity of objects	• Object detection and classification • 3D localization • 3D mapping
Short-range RADAR	• Large FoV • Easier to develop • Resistant to bad weather	• Large package size • Shorter sensing range	• Blind spot detection • Parking aid
Long-range RADAR	• Higher accuracy • Better resolution • Smaller package size	• More data losses • Narrow Field of View at short distances	• Speed calculation of detected vehicles • Used on highways and cross-traffic alert systems
Ultrasonic	• Direct distance estimation • Can operate in harsh weather conditions • Can detect near objects (<2m)	• Can only detect near objects • Low angular resolution	• Parking assistance • Near object detection

ii. LiDAR

LiDAR, which stands for Light Detection and Ranging, leverages the Time of Flight (ToF) principle to measure the distance between the sensor and detected objects. LiDARs emit laser pulses; these waves reflect off objects and are received by the LiDAR, which calculates ToF. These reflected and returned pulses are processed into a 3D visualization known as a 'point cloud'.

LiDARs boast a maximum operational detection range of up to 200m [58] and remain resilient in varying weather and lighting conditions. Diverse types of LiDARs project differing numbers of laser beams. Two-dimensional LiDARs employ a solitary beam projected onto a rotating mirror, whereas 3D LiDARs employ multiple laser diodes that spin at high speeds. The greater the number of laser diodes, the greater the number of measurements gathered thereby enhancing the precision of perceptual tasks [59].

In some applications, multiple 2D LiDARs (each with a single beam) have been utilized for vehicle detection [60] and pedestrian detection [61, 62] through the application of pattern recognition techniques. However, this approach confines detection to a limited set of object classes.

Multi-beam LiDARs, on the other hand, are used for 3D object detection [63, 64]. Examples of 2D LiDARs used in previous research are LRS-1000, LMS-291, and UTM-30LX, while examples of multi-beam LiDARs are IBEO LUX, Velodyne, and Quanergy. Three-dimensional LiDARs are more computationally extensive and suffer from higher costs.

There are three primary approaches to achieving 3D detection using point clouds [65]:

- Projecting a point cloud onto a 2D plane to implement 2D detection frameworks, subsequently obtaining 3D localization from the projected images.
- Utilizing volumetric techniques involving voxelization [66, 67]. However, 3D convolutional operations pose high computational demands.
- Employing PointNets [68-70] to directly use raw point clouds for predicting 3D bounding boxes. This approach also involves significant computational overhead and increases runtime.

iii. RADAR

It boasts remarkable resilience to diverse weather and light scenarios, excelling in multi-depth detection and long-range capabilities, as used in [71, 72]. In another study [73], a compact automotive RADAR sensor was employed to assess its proficiency in executing fundamental tasks like vehicle odometry, road structure mapping and tracking moving objects.

iv. Ultrasonic

Operating on the ToF principle, it measures the duration taken for sonic waves to travel to an object and return as an echo. Ultrasonic sensors are resilient and capable of determining distances to objects regardless of their colour, material, or prevailing weather conditions. Nevertheless, their applications are confined to short-distance measurements and parking systems.

2.3.2 Proprioceptive Sensors

To achieve accurate path planning for the ego vehicle, its state must be continuously monitored using its proprioceptive sensors: Global Navigation Satellite System (GNSS), Inertial Measurement Unit (IMU), and wheel odometry. The GNSS, a satellite-based navigation system, imparts context and real-time information to AVs. Most of the available GNSS sensors employed in AVs incorporate an IMU, which, when fused with other sensors, equips the AV to measure metrics like velocity, Euler angles and angular velocity. This integration facilitates ego-motion correction. A famous example of GNSS/ IMU is the X-sense MTi-G.

Furthermore, wheel odometry records the rotation rates of the vehicle's wheels, thereby estimating its speed and acceleration [64]. Developers of Advanced Driver-Assistance systems (ADAS) assert that even level 3 AVs necessitate the integration of high-precision proprioceptive sensors. These sensors provide the AV with accurate location information and the intended path, ensuring operational competence even in scenarios where the lane view might be obstructed.

2.3.3 Sensor Fusion

Relying solely on a single type of sensor has demonstrated inefficiency and unreliability. Thus, the necessity of sensor fusion arises to transcend individual sensors' limitations by harnessing multiple sensors' strengths. Consequently, sensor fusion amplifies reliability and accuracy, and diminishes measurement uncertainty. The classification of sensor fusion can be based on:

- Relationship between input data sources, as proposed by Durrant-Whyte [74]: (*a*) Complementary, (*b*) Redundant, and (*c*) Cooperative.
- Types of input and output data, as proposed by Dasarthy [75]: (*a*) Data-In-Data-Out, (*b*) Data-In-Feature-Out, (*c*) Feature-In-Feature-Out, (*d*) Feature-In-Decision-Out, and (*e*) Decision-In-Decision-Out.
- Different Abstraction levels, as proposed by [76]: (*a*) Signal-level (*b*) Pixel-level (*c*) Characteristic-level (*d*) Symbol-level.
- Architecture type [77]: (*a*) Centralized, (*b*) Decentralized, and (*c*) Distributed.

Different sensors data can be fused together, below are examples of the most widely applied sensor fusion combinations:

i. Camera and LiDAR fusion

The fusion between a camera and a LiDAR is primarily used to achieve obstacle detection and avoidance in various approaches, as presented in [69, 78-86]. In [85, 86], data from both 3D LiDARs and cameras were integrated to achieve comprehensive obstacles perception. The LiDAR's role was to ascertain the precise object positions, whereas the camera contributed features and classification information. In a study by Han et al. [86], a decision-level sensor fusion approach was employed on a Velodyne 64-beam LiDAR along with an RGB camera. This combination aimed to enhance the detection of dim objects such as pedestrians and cyclists. Similarly, in studies by Chen, Xiaozhi et al. and Qi et al. [69], Regions of Interest (ROIs) were extracted from images, with the LiDAR providing 3D localization.

Furthermore, a three-dimensional object detector operating within the Bird's Eye View (BEV) space has been introduced in work by Liang et al. [82]. This approach involves merging image features by learning to map them onto the BEV space. The integration is achieved through continuous convolutions that fuse image and LiDAR feature maps across varying resolution levels. Likewise, in the study conducted by Xiaozhi et al. [78], a deep fusion benchmark was devised, surpassing the prevailing state-of-the-art performance by 25% and 30% in terms of Average Precision (AP). Similarly, in [79, 80], two-dimensional Convolutional Neural Networks (CNNs) were employed on both BEV and image feature maps, followed by fusion at the intermediate stage through region-wise convolutional feature maps.

Several approaches have been directed towards the detection of specific classes of objects. For instance, in [87], a pedestrian detection system was introduced. This system employed pedestrian pattern matching and recognition techniques, augmented by context information, to estimate potential danger linked to the identified pedestrians. Their experimental results yielded positive detection rates of 82.29% with 1.11% false positives per frame. These outcomes were juxtaposed with scenarios involving solely cameras (73.97% positive detections, 5.27% false positives) and only LiDAR (74.56% positive detections, 13.3% false positives).

In addition, Garcia et al. [81] addressed the vehicle detection challenge. This study capitalized on context and real-time Global Positioning System (GPS) information to enhance the detection process. The fusion results yielded a positive vehicle detection rate of 92.03%, with a mere 0.59% missed detections per frame. These outcomes were contrasted with those achieved using solely a camera (47.72% positive detections, 1.13% missed detections) and solely a LiDAR (91.03% positive detections, 8.19% missed detections). Another noteworthy example pertains to passive beacon detections, as discussed in the study by Rövid & Remeli [84]. In this context, the authors proposed a delineation methodology that improved performance by fusing camera and LiDAR data.

Beyond the scope of object detection and localization, the fusion of LiDAR and camera sensors has enabled the functionalities of urban mapping and vehicle positioning, as evidenced by studies such as [85, 88, 89]. Shi et al. [88] introduced an innovative approach that amalgamated a panoramic camera with a 2D laser scanner for urban mapping and localization. Their outcomes exhibited an average error of 0.2m in the horizontal plane along a 580m trajectory. In the work by Zhang, J. & Singh [89], a distinctive strategy involving a rotating motor-mounted Hokuyo UTM-30LX LiDAR and a monocular camera achieved 3D mapping. The LiDAR was set to rotate by 180°. Xue et al. [85] pursued the integration of LiDAR and camera sensor data to facilitate effective autonomous positioning and obstacle perception through geometric and semantic constraints. Despite its complexity, the algorithm's real-time constraints were not met, and the detection of faint objects was not considered.

Although the fusion of camera and LiDARs achieved promising results, existing approaches suffered from one or more of the following:

- Detection of specific objects classes: [80, 81, 84-87],
- Employing 3D LiDARs: [69, 79, 80, 82, 85, 86]
- Not real-time: [78, 80]

ii. Camera and RADAR fusion

The incorporation of RADARs into AVs brings the added advantage of reliable and precise obstacle detection even in challenging scenarios such as low visibility, adverse weather conditions, object occlusions, and distant targets. Some research has addressed the problem of detection of specific object classes; an example is a study by Palffy et al. [90], where

particular attention was given to addressing the issue of pedestrians crossing roads, mainly when they are obscured by surrounding vehicles. The study leveraged RADAR's capability to detect pedestrian reflections through multi-path propagation, even when entirely occluded. Similarly, in the research conducted by Chadwick et al. [91], RADAR integration with cameras was showcased to enhance the detection performance of small and distant objects. Their investigation focused on two object classes: vehicles and pedestrians/cyclists.

The integration of camera and RADAR data has demonstrated efficacy not only in object detection, but also in classification. Jhan et al. [92] exemplified this by combining data from both camera and RADAR sensors to achieve object classification and relative distance measurement. The approach introduced in their study operates in a real-time environment facilitated by its swift detection capabilities. However, their approach has been only evaluated on short distances and not proven accurate on long distances.

Furthermore, an innovative avenue of research involves deploying Deep Learning (DL) methodologies on RADAR signals to identify the presence of objects in the surrounding scene and subsequently determine their three-dimensional positions. Zhang et al. [72] explored this novel approach by utilizing a deep learning framework as opposed to conventional signal processing techniques on RADAR data. Their proposed system was evaluated in the context of car detections within noisy environments and was not tested on other different objects having different features (e.g., dim objects, small/distant objects, etc.).

As shown, most of the existing camera-RADAR fusion approaches are dedicated to the detection of specific object classes or objects in a short range, which makes this direction need more development and exploration in order to be appropriate for AVs environment perception.

iii. Camera, LiDAR and RADAR fusion

This method is often regarded as the costliest approach to sensor fusion, and numerous researchers have explored various implementations of this sensor combination. For instance, in the study by Steyer et al. [93], pre-fused measurement data was leveraged to introduce an innovative grid-based object-tracking methodology. RADAR Doppler velocity estimates were seamlessly integrated into the input data in order to enhance the accuracy of motion estimation. Their approach was evaluated using actual sensor data, with a specific focus on autonomous driving in complex urban settings.

Furthermore, in the work by Ahrabian et al. [94] a sensor fusion strategy was employed, encompassing a RADAR, ZED stereo-camera, and an HDL-32E Velodyne LiDAR. The researchers developed an algorithm capable of tracking potential objects within the scene without being restricted to specific object types (object agnostic). Similarly, using the identical LiDAR and stereo-camera as outlined by Ahrabian et al. [94] along with the addition of a 150 GHz RADAR, Daniel et al. [95] investigated their approach within a simulated fog-laden environment. This experimentation underscored the significance of including a RADAR sensor as an integral component within the automotive industry.

In the study by Kunz et al.[21], the emphasis was on centralized sensor-independent fusion schemes designed to facilitate straightforward sensor substitution and ensure redundancy through the utilization of probabilistic and versatile interfaces. Their approach involved the integration of three IBEO LUX 3D LiDARs, monocular cameras, and multiple RADAR sensors. Another strategy, devised by Cho et al. [96], allowed the system to seamlessly switch between a point model and a 3D-box model, depending on the proximity of objects to the vehicle.

Fusion of different data modalities by applying sensor fusion on a bigger number of different sensors provides the system with a big size of variant features of the surrounding environment, enhancing the accuracy and precision of the operation of objects detection performed by AVs. However, this also implies the burden of the high cost of the sensors employed. Moreover, processing big amounts of data leverages the need for heavy complex computations that need powerful processing units to overcome the computational needs.

2.4 Object Detection Using Artificial Intelligence Techniques

The object detection phase facilitates a semantic comprehension of the captured data, involving the identification of both the class and the spatial location of identified objects. To ensure the development of a dependable autonomous vehicle equipped with the ability to comprehend its surroundings, the selection of appropriate object detection techniques becomes crucial for efficient performance across the desired environments. The diversity of driving scenarios presents distinct challenges in object detection:

- Variable weather and lighting conditions (for example, fog, snow, rain, morning, night, etc.)

- Object's reflectivity (dim vs. highly reflective)
- Detection of small/distant objects
- Object's movements
- Movement and speed of the ego-vehicle
- Obstacles occlusion and truncation
- Presence of multiple objects.

Other challenges in choosing the suitable algorithm include:

- Computation time and complexity
- The required computation hardware for the algorithm.
- Availability of datasets and training data

Object detection falls into two primary categories: 2D and 3D. The choice of the detection technique predominantly depends on the intended application, the sensors employed, and the subsequent tasks following the detection process (e.g., sensor fusion). For example, video surveillance applications may suffice with 2D detections alone. On the other hand, the functioning of autonomous vehicles depends on the accurate 3D localization of detected objects. Various sensors can be used to achieve object detection, but the visual system retains its prominence as a primary data source as it can perform object classification as well.

In autonomous driving, the task of object detection encompasses diverse classes that can be categorized as either static or dynamic objects. Static objects include entities like traffic lights, signs, buildings, bridges, and curbs, while dynamic objects include pedestrians, cyclists, animals, and various vehicles. Detecting static objects is generally perceived as more straightforward due to their predicted shapes and fixed positions.

Two fundamental approaches to object detection exist: Machine Learning (ML) and Deep Learning (DL), as shown in Table 2.4. Illustrative examples of ML techniques encompass the Scale-Invariant Feature Transform (SIFT) [97], Histogram of Oriented Gradients (HOG) [98], and the Viola-Jones object detection framework founded on Haar-like features [99] . Meanwhile, instances of DL approaches and architectures include Region proposals (R-CNN) [100], Fast R-CNN [101], Faster R-CNN [102], Single-Shot multi-box Detector (SSD) [103], and You Only Look Once (YOLO) [104]. Given the variability of object classes, lighting conditions, and backgrounds, relying solely on feature extraction achieves

less robust performance. DNNs play a pivotal role in object detection tasks, offering the ability to learn and extract intricate features. Additionally, their training algorithms prevent the need for manual feature design.

Table 2.4: Machine Learning vs. Deep Learning

Criteria	**Machine Learning (ML)**	**Deep Learning (DL)**
Description	Uses statistical algorithms and models to improve computer systems' performance on a specific task by learning from data without being explicitly programmed	Uses neural networks with multiple layers to learn from huge amounts of data to perform complex tasks
Data requirements	Can be trained on small to medium-sized datasets	Requires a significant amount of data to train
Performance	DL algorithms outperform ML algorithms; however, they require a lot of computation power	
Feature Engineering	Requires manual feature engineering to extract relevant features from the data	It can automatically extract features from the data, which reduces the need for manual feature engineering
Applications	Predictive modelling, clustering, and classification tasks	Image and speech recognition, natural language processing, and robotics tasks

2.4.1 Object Detection using DNNs

Convolutional neural networks (CNNs) have revolutionized object detection by enabling automated and highly accurate detection of objects in complex scenes. Figure 2.2 presents a chronological illustration that highlights key milestones in the implementation of object detection for AVs. It shows milestones of object detection for AVs, including the state-of-the-art DL-based approaches (R-CNN [100], Fast-CNN [100], Faster-CNN [102], SSD [103], YOLO [104]), CNN-based Networks ([53, 67-69, 79, 105-110]), and reliable real-time object detection approaches ([50, 51, 53, 86, 111-113]). Handcrafted features dominated the period prior to 2013. A transition took place in 2013 with the development of CNNs, which contributed to the emergence of multiple real-time object detectors for driving scenes

Two prominent approaches that leverage Convolutional Neural Networks (CNNs) for object detection are Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO). These methods have played a pivotal role in advancing object detection system's accuracy, efficiency, and real-time capabilities. Since the development of R-CNN networks [100], much research has turned towards the employment of DNNs in object detection [100-104].

i. Single Shot MultiBox Detector (SSD)

SSD is a real-time object detection method that combines object localization and classification in a single pass through the network. It is designed to be efficient while maintaining high accuracy. Key features of SSD include:

- Anchor Boxes: SSD employs anchor boxes of various aspect ratios and scales at different locations within an image. These anchor boxes act as references for predicting object positions and sizes.
- Multi-layer detection: SSD uses multiple convolutional layers with varying receptive fields to capture objects of different sizes. Detection predictions are generated at different scales to handle objects at various resolutions.
- Real-time performance: SSD's single-pass architecture and multi-layer predictions enable real-time object detection suitable for applications like video analysis, robotics, and autonomous vehicles.

ii. You Only Look Once (YOLO)

YOLO is recognized for its real-time object detection capabilities achieved through a single forward pass of a CNN. YOLO's unique grid-based approach allows it to make predictions at multiple spatial scales within a single iteration. Key features of YOLO include:

- Grid-based detection: YOLO divides the input image into a grid, with each grid cell responsible for predicting objects within its boundaries. Each cell predicts bounding box coordinates and class probabilities.
- Single forward iteration: YOLO's single-pass mechanism directly outputs object predictions, making it exceptionally fast and suitable for real-time applications.
- Speed-accuracy balance: While YOLO offers real-time performance, it may not match the accuracy of some other methods. However, YOLOv3 have improved accuracy while maintaining impressive speed.

Table 2.5 provides a comprehensive comparison of the performance of the three primary DL-based object detection methods (CNNs, SSD, YOLO) considering their speed (Frames per Second (FPS)) when executed on a Graphics Processing Unit (GPU), along with their mean Average Precision (mAP) of detections over different objects classes. A comprehensive exploration of object detection using deep learning is covered in [114]. It is essential to highlight that SSDs demonstrate limitations in detecting small objects. Conversely, numerous CNN-based detection networks have been developed due to the diverse advantages of CNN-based approaches (such as hierarchical feature representation, increased expressive capability, and the combination of several tasks together), as depicted in Table 2.6. CNN-based detection has been effectively applied to achieve the following:

- Vehicle detection (ex: [115])
- Pedestrian detection (ex: [115-117]). Furthermore, a survey was conducted to explore the utilization of deep learning techniques for pedestrians' detection and tracking in [118]. Additionally, a comprehensive survey on camera-based pedestrian detection methods can be found in [119].
- Cyclists' detection (ex: [120, 121])

Table 2.5: Different deep learning approaches

	Approach	**Speed (FPS)**	**Testing dataset**	**mAP (%)**
CNN	R-CNN [100]	6	VOC	53.3
	Fast R-CNN [101]	0.5	VOC	70
	Faster R-CNN (VGG16) [102]	7	VOC	73.2
SSD	SSD300 [103]	59	VOC	74.3
	SSD512 [103]	22	VOC	76.8
YOLO	YOLOv1 [104]	45	VOC	63.4
	YOLOv1-Tiny [122]	155	VOC	52.7
	YOLOv2 [123]	67	VOC	76.8
	YOLOv2 [123]	40	VOC	78.6
	YOLOv2 [122]	40	COCO	48.1
	YOLOv2-Tiny [122, 123]	244	COCO	23.7
	YOLOv3 [122, 124]	51	COCO	57.9
	YOLOv3-Tiny [122]	220	COCO	33.1

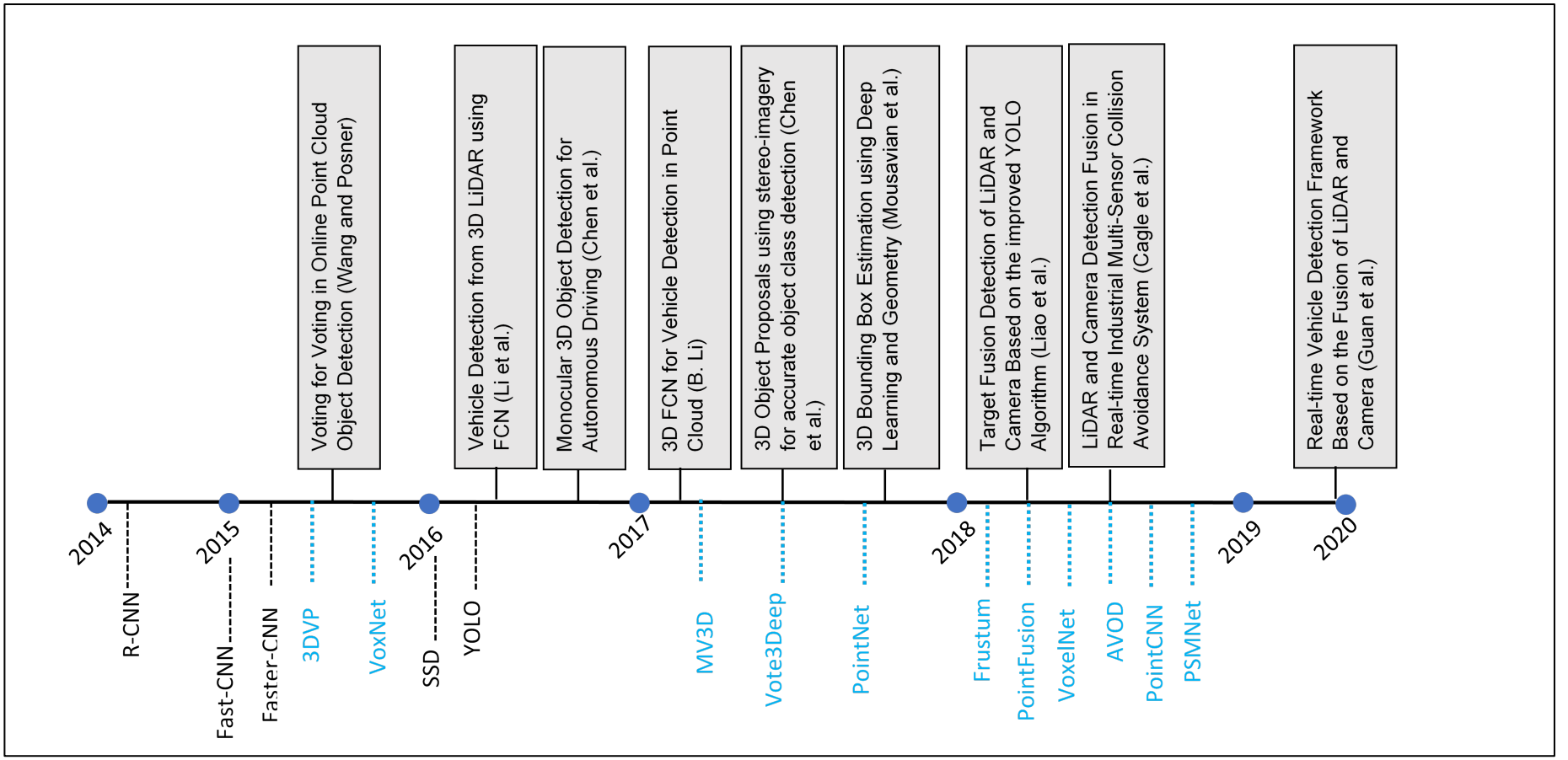

Figure 2.2: Milestones of object detection for AVs

Due to the exceptional speed of YOLO, several research efforts have been dedicated to expanding its capabilities:

- *YOLO9000* [123]*:* In this work, YOLO9000 was trained on both the COCO detection dataset and the ImageNet classification dataset. This simultaneous training approach enables YOLO9000 to predict unlabelled data. Validation on the ImageNet dataset yielded a mAP of 19.7%, while for 156 classes not present in the COCO dataset, it achieved a mAP of 16.0%. As a result, YOLO9000 can perform real-time detection for over 9000 object categories.
- *YOLO-LITE* [122]*:* This variant was developed to cater to portable devices without GPUs. Trained on the PASCAL VOC and COCO datasets, it achieved mAPs of 33.81% and 12.26%, respectively. Its notable advantage lies in its high FPS, which reaches 21 FPS on non-GPU platforms.
- *Complex-YOLO* [125]*:* It is an extension of YOLOv2 that integrates the Euler-Region-Proposal-Network to leverage 3D point cloud data from multi-beam LiDARs, such as the 64-beam Velodyne LiDAR. This extension enables 3D detections for all eight classes in the KITTI dataset, running at a speed of over 50 FPS on an NVIDIA TitanX GPU. The achieved APs for the main object classes in the KITTI dataset were as follows:
    - Cars: 67.72% (easy), 64.0% (moderate), 63.01% (hard)
    - Pedestrian: 41.79% (easy), 39.7% (moderate), 35.92% (hard)
    - Cyclists: 68.17% (easy), 58.32% (moderate), 54.3% (hard)

Table 2.7 presents an overview of recent advancements in utilizing deep learning for object detection tasks in autonomous vehicles. As illustrated, many approaches are dedicated to the detection of specific classes of objects, such as pedestrians [86, 126], traffic cones [83], and vehicles [112]. Some approaches suffered from slow inference time [50, 51, 53, 112, 127]. The slow operation of some existing approaches makes them hard employ on AVs, as autonomous driving requires fast real-time computations in order to achieve real-time decisions.

Table 2.6: CNN-based object detection networks

Network	Approach	Time Analysis	Modality	Performance
3DVP [107]	Translation of point clouds into voxels	-	RGB images	• 3D AP on KITTI (car: 87.46%, 75.77%, 65.38%) • AOS (86.92%, 74.59%, 64.11%) • AP on OutdoorScene (car: 90.0%, 76.5%, 62.1%)
MV3D [78]		0.36s/image (TitanX GPU)	LiDAR point cloud, RGB images	3D detection AP on KITTI dataset (easy IoU: 56.56% - hard IoU: 96.52%)
Vote3Deep [109]		-	LiDAR point cloud, RGB images	2D AP on KITTI dataset (easy, moderate, hard): • Car: 76.79%, 68.24%, 63.23% • Pedestrian: 68.39%, 55.37%, 52.59% • Cyclists: 79.92%,67.88%, 62.98%
VoxelNet [67]		0.033ms (TitanX GPU)	LiDAR point cloud	3D AP on KITTI (easy, moderate, hard): • Car: 77.47%, 65.11%, 57.73% • Pedestrian: 39.48%, 33.69%, 31.51% • Cyclists: 64%, 52.18%, 46.61%
PointCNN [105]	point cloud processing using multi-layer perceptrons	0.012s per batch for inference of batch size 16 (GPU NVIDIA Tesla P100)	LiDAR point clouds	• ModelNet40 dataset: ○ Pre-aligned: mA (88.8%), OA (92.5%) ○ Unaligned: mA (88.1%), OA (92.2%) ScanNet dataset mA (55.7%), OA (79.7%)
PointNet [68]		-	LiDAR point cloud	• ModelNet40 dataset: mA (86.2%), OA (89.2%) • Stanford dataset: 3D detections mAP 24.24%
Frustum [69]		4 FPS (GPU NVIDIA GTX 1080i)	LiDAR point cloud, RGB images	• mAP on SUN-RGBD: 54% • 3D AP on KITTI (easy, moderate, hard): ○ Car: 81.2%,70.39%,62.19% ○ Pedestrian: 51.21%, 44.89%, 40.23% ○ Cyclists: 71.96%, 56.77%, 50.39%
PointFusion [108]	Faster R-CNN, PointNet	-	LiDAR point cloud, RGB images	• mAP on SUN-RGBD: 45.38% • 3D AP on KITTI (easy, moderate, hard): ○ Car: 77.92%, 63%, 53.27% ○ Pedestrian: 33.36%, 28.04%, 23.38% ○ Cyclists: 49.34%, 29.42%, 26.98%
VoxNet [106]	Volumetric occupancy grid using a supervised 3D CNN	0.002s/2000 points (GPU Tesla K40)	LiDAR point clouds, RGBD images, CAD data	Average Accuracy on Sydney Urban Objects dataset: • ModelNet10: 0.92 • ModelNet40: 0.83 • NYUv2: 0.71
AVOD [79]	Combined fusion approaches, Faster R-CNN	0.1s/ frame (GPU TitanXP)	LiDAR point cloud, RGB images	3D AP on KITTI (easy, moderate, hard): • Car: 81.94%, 71.88%, 66.38% • Pedestrian: 50.8%, 42.81%, 40.88% • Cyclists: 64%, 52.18%, 46.61%
PSMNet [110]	Spatial Pyramid pooling and 3D CNN	Real-time	LiDAR point cloud, RGB images	• 92.03% positive detections, 0.59 misses per frame on real driving conditions

Table 2.7: DL-based object detection approaches

Approach	Base Architecture	Target Objects	Time Analysis	Input Data	Accuracy
Liu et al. [126]	YOLOv2	Pedestrians (2D)	73 FPS on NVIDIA GPU GTX 1080Ti	RGB images	90.9% AP on INRIA & Caltech datasets
Han et al. [86]	Improved YOLO	Dim objects (3D)	13 FPS on NVIDIA GPU	64-beam LiDAR & RGB camera	82.9% mAP on a 6-category dataset of 3000 frames
Wei et al. [83]	YOLO and SVM	Traffic cones (3D)	5Hz on NVIDIA Jetson TX2 GPU	8-beam LiDAR & RGB camera	True positive rate, false negative rate: (3-20m range: 97.6%, 2.4%), (20-40m range: 94.8%, 5.2%) on a collected dataset
Li et al. [112]	Fully Convolutional Networks	Vehicles (3D)	-	64-beam LiDAR	AP on KITTI dataset (easy, medium, hard): (84.2%, 75.3%, 68%)
Wang et al. [127]	Sliding Window Approach, CNN	Car, pedestrians, cyclists	0.5s/100k points on a multi-core CPU (i7)	3D LiDAR	AP on KITTI dataset (easy, medium, hard): Car: 47.99%, 56.8%, 42.57%, Pedestrian: 35.74%, 44.48%, 33.72%, Bicyclists: 31.24%, 41.43%, 28.6%
Chen et al. [53]	CNNs	3D objects	2s/image on NVIDIA GPU TitanX	Stereo-camera	3D AP of cars on KITTI (89.49%, 81.21%, 74.32%)
Mousavian et al. [50]	Deep CNNs	3D objects	1.1s/ image	RGB Camera	AP cars (92.98%, 89.04%, 77.17%), AOS cars (92.9%, 88.75%, 76.76%) on KITTI dataset
Chen et al. [51]	CNNs	3D objects	1.8s inference time on a single core	Monocular Camera	AP on KITTI dataset (easy, medium, hard): Car: 92.33%, 88.66%, 78.96%, Pedestrian: 80.35%, 66.68%, 63.44%, Cyclists: 76.04%, 66.36%, 58.87%

2.4.2 Three-dimensional Object Detection

Table 2.8 provides a compilation of the latest and widely recognized 3D object detection networks and frameworks, along with their respective limitations. Most approaches that are focusing on 3D detections are based on processing 3D point clouds along with RGB images. Existing approaches suffer from one or more of the following limitations:

- Not real-time
- Detection of static objects
- Detection of specific objects classes

Table 2.8: Three-dimensional object detection networks and frameworks

Research	Modality	Limitation
Multi-task multi-sensor fusion for 3D object detection [128]	RGB + 3D point cloud	Expensive 3D LiDAR Not real-time
Frustum pointnets for 3D object detection from rgb-d data [69]	RGB-D	0.12s per frame Not real-time
Pointfusion: deep sensor fusion for 3D bounding box estimation [108]	RGB + 3D point cloud	1.3s per frame Not real-time
RoarNet: a robust 3D object detection based on region approximation refinement [129]	RGB + 3D point cloud	Expensive 3D LiDAR Not real-time
A frustum-based probabilistic framework for 3D object detection by fusion of LiDAR and camera data [130]	RGB + 3D point cloud	Only for detecting static objects
SEG-VoxelNet for 3D vehicle detection from RGB and LiDAR data [131]	RGB + 3D point cloud	Only detects vehicles Not real-time
MVX-Net: multimodal voxelnet for 3D object detection [132]	RGB + 3D point cloud	Not real-time
3D-cvf: generation joint camera and LiDAR features using cross-view spatial feature fusion for 3D object detection [133]	RGB + 3D point cloud	NVIDIA GTX 1080Ti, inference time $75ms$ per frame (13.33 FPS)
PI-RCNN: an efficient multi-sensor 3D object detector with point-based attentive cont-conv fusion module [134]	RGB + 3D point cloud	Not real-time
Image guidance-based 3D vehicle detection in traffic scene [135]	RGB + 3D point cloud	Only vehicles, 4FPS
Epnet: enhancing point features with image semantics for 3D object detection [136]	RGB + 3D point cloud	Not real-time

Although much research applied sensor fusion between cameras and LiDARs to achieve 3D detections, not enough research has considered employing 2D LiDARs in addition to cameras to achieve 3D detections. This is because attaining straightforward depth measurements of surrounding objects from 2D LiDARs is a challenging operation due to the

single horizontal plane measurements. Moreover, objects features cannot be directly acquired from 2D LiDAR measurements; as a result, in order to achieve 3D detections using 2D LiDARs and cameras, complementary fusion should be applied.

2.5 Object Detection in Rainy Weather Types

To recall, the difference between levels 4 and 5 AVs is that level 5 AVs can operate under any ODD. The most advanced currently available AVs are at level 4 autonomy. Consequently, the operation of AVs must demonstrate reliability across all demanding scenarios to gain public confidence and justify the investment. Therefore, one of the tasks that should be performed, even under challenging weather conditions, is object detection.

Diverse weather conditions can degrade image quality, leading to distortion and impairment. These conditions can be categorized as either stable or dynamic [137]. Steady weather conditions encompass fog, mist, and haze, whereas dynamic weather conditions involve rain and snow. Rainy weather conditions, characterized by larger and variable droplets, have a more pronounced impact on scene visibility. These conditions reduce contrast and scene visibility, significantly affecting the ability of AVs to detect objects.

Our study focuses on addressing the deterioration caused by rain in captured videos of the environment. This is particularly crucial as rain is the most prevalent dynamic weather condition, particularly in regions like the United Kingdom [138]. Despite the progress made by researchers in the field of deraining techniques, real-time video deraining for autonomous vehicles remains a relatively unexplored area.

2.5.1 Rain Degradation

i. Types of rain

Rain can be defined by the existence of numerous drops that come in diverse sizes, intricate forms, and varying velocities. Rain contributes to two distinct types of reductions in visibility. *Rain streaks* incline to generate specular highlights and obstruct and distort background scene elements. Conversely, *distant rain streaks accumulations* generate atmospheric veiling effects that scatter light and obscure the line of sight, resembling the impact of fog [139]. Since the attributes of rain streaks and the accumulation of rain are distinct, the distortions they introduce to images differ. Consequently, researchers tackle each type of rain degradation as a distinct problem.

Figure 2.3 shows the impact of different rain degradation forms on images. The green box shows the effect of heavy rain streaks; the yellow box shows the impact of the presence of rain streaks of different sizes overlapping on each other. The blue and red boxes show the impact of the accumulation of rain streaks resembling a foggy/veiling effect; it can be noted that the veiling effect becomes stronger on further objects [140].

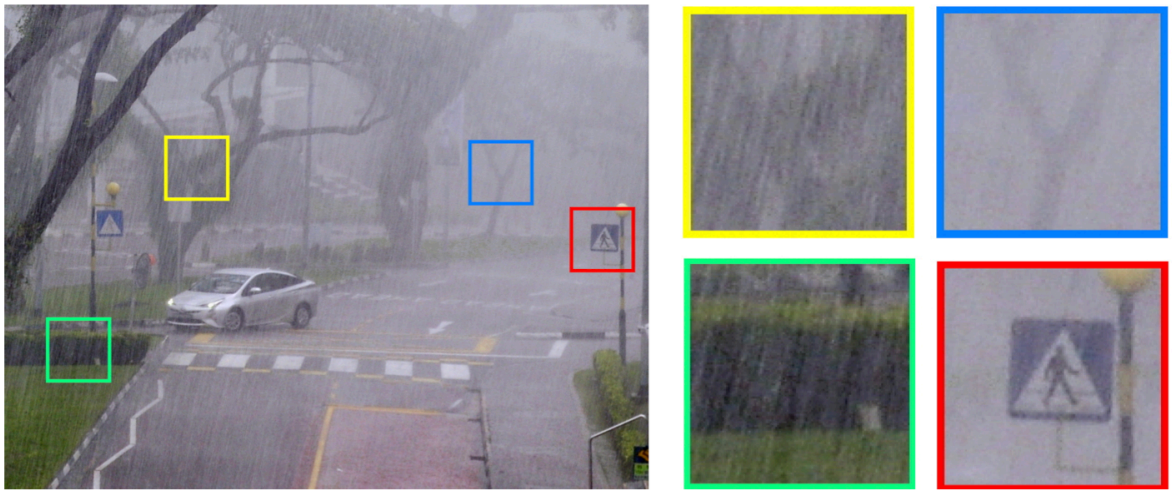

Figure 2.3. An example of the presence of different rain degradations in the same frame

ii. Rain effect on video frames

The presence of raindrops blocks the light reflected from the surrounding objects. Also, rain distorts the intensity in images and video frames [5]. Deraining techniques can be divided into two primary categories: video-based and single-image-based methods. Video-based methods often exploit temporal information within consecutive frames. In contrast, single image-based deraining poses a more challenging task due to the absence of temporal redundancy knowledge and the limited information available per frame [141, 142]. As a result, more significant research effort has been dedicated to devising diverse algorithmic approaches to tackle this issue. Furthermore, existing single image-based methods can be categorized into three distinct groups according to Wang, Hong et al. [6]:

- Filter-based methods [143-146]: It is based on identifying different rain properties and designing appropriate filters to achieve rain-free images.
- Prior-based methods [147-152]: It is based on leveraging prior knowledge of both rainy and clear images, prior-based methods encompass a range of techniques, among which is Morphological Component Analysis (MCA) [153], histogram of oriented

gradients (HOG) [154], structural similarities, and sparse representation models [152];

- Deep learning-based methods [155, 156].

iii. Rain effect on LiDAR

Much research has studied the operation of the Hokuyo UTM-30LX 2D LiDAR used in this thesis [157-159]. Authors in [158] examined its performance under different environments; they performed tests in several scenes. It has been shown that raindrops were not detected and did not impose a noticeable difference on the detection of background objects. One of the explanations for this phenomenon is that raindrops in the air are too small that it does not impose an effect on the laser beams. Moreover, measurements captured by the Hokuyo UTM-30LX tend to stabilize for about 4 minutes before preheating.

iv. Rain effect on object detection

The performance of object detection methods is heavily influenced by how diverse the training and testing data are. According to extensive experiments performed by Hnewa and Radha in [5], the performance of object detection frameworks which are trained on non-rainy datasets degrades when tested on rainy datasets. This degradation is due to the fact that rain hides and distorts important visual features which are used for detection.

The Average Precision (AP) of the detection of small and distant objects declines more significantly than that of bigger objects. Large objects, such as vehicles, occupy larger regions in a frame; therefore, even when rain exists, there are still enough features that can be extracted and detected. Also, objects made of reflective material are better detected even if distorted with rain [5].

v. Related work and deraining challenges

Numerous methods have been suggested to tackle rain detection and removal, as outlined in Table 2.9. This challenge can be approached either from a video-based perspective or as an image-based task. However, a significant portion of these approaches did not take into account the specific requirements of AV systems and encountered the following deficiencies:

- High computation time: [160-162]
- Only addressed the problem of rain detection: [163, 164]

- Assumes static scenarios: [165], also ref. [166]'s approach does not work with highly dynamic raindrops. The approach of [167] fails to separate dynamic textures and moving objects.
- Not applicable for real-time scenarios:
    - The approach of [168] requires the interaction of a human operator to process the captured frames. Also, it is bound by certain scenes (lane and building scenes).
    - The approaches of [161, 169] exploit temporal information during the recovery process.
    - The approach of [170] makes use of previously cleaned frames.
    - The approach of [162] suffers from high computational time.
- Limited to certain degradation factors: [9, 160, 171, 172]

According to [173], no existing de-raining method directly aids in the objects detection task that happens afterwards. This encouraged developing new robust algorithms to account for high-level vision problems on real-world driving scenarios.

2.5.2 State-of-the-Art Deraining Networks

Traditional deraining models, which are based on the detection of the statistical characteristics of rain, depend on performing linear mapping transformations. Therefore, they are not resilient to dynamic rain variations, such as different streak directions, intensities, and orientations [174-177]. Recently, CNN-based Deep-Learning (DL) techniques [140, 161, 178] have emerged and shown significant improvements compared to traditional methods.

Table 2.9: Rain detection and removal approaches

Paper	Input	Method	Limitations
[160]	Rain streaks (images)	Single-step rain detection and removal by analysing rain's local features (orientation, amplitude, etc) and applying anisotropic filter.	Low speed, temporal dependency
[164]	Rain drops (images)	Cellular Neural Networks, and SVM. Pattern recognition using SVM produces a model for the classification phase using Cellular Neural Networks.	Detection only, no removal performed
[165]	Any rain condition (videos)	They used the significant difference in time evolution between the rain and non-rain pixels in videos, and then analysed this difference with the help of skewness.	static scenes
[166]	Rain drops (videos)	Based on motion and intensity of temporal derivatives of the input video. The authors assume that pixels without raindrops exhibit swifter motion compared to those with raindrops, and the temporal shift in intensity of non-raindrop pixels is greater than that of raindrop pixels.	Temporal dependency, static scenes
[168]	Rain drops (videos)	Hough transformation and Sobel filters	Human operator
[169]	Rain streaks (videos)	Spatial and temporal information, incorporating motion segmentation for dynamic scenes. Rain removal filters are employed following the application of photometric and chromatic constraints used for rain detection.	Temporal dependency
[161]	Rain streaks (images)	De-rainNet: leveraging domain knowledge in image processing to enhance deraining using a moderately sized CNN	Low speed
[170]	Rain streaks and accumulation (videos)	A rain-free frame is predicted based on a single rainy frame and then used as a reference along with previously restored clean frames to enhance the accuracy of obtaining a clean frame.	Temporal dependency
[162]	Rain Streaks (videos)	Multiscale convolutional sparse coding technique for eliminating rain effects.	Low speed
[171]	Rain streaks (images)	Frequency division, and non-negative matrix factorization	Limited to rain streaks removal
[172]	Rain Streak (videos)	Super pixel segmentation to conduct scene decomposition into units consistent with depth.	Limited to rain streaks removal

The majority of cutting-edge algorithms are rooted in convolutional neural networks (CNNs). Various modules can be integrated into deraining networks, including:

- Residual blocks are used in [179-183]. They are also called skip connections or shortcut connections; they are a building block that allows the network to learn the residual between the input and output of a layer. In other words, it models the difference between the desired output and the actual output of a layer. By incorporating this difference, the network can more easily learn identity mappings and residual mappings, facilitating the training of very deep networks.
- Dilated convolution is used in [178, 184]. Dilated convolutions are a powerful tool in deep learning architectures, especially for tasks that require capturing contextual information at different scales, by efficiently expanding the receptive field without increasing computational overhead.
- Dense blocks are used in [185]. They are a unique innovation that enhances feature reuse, gradient flow, and parameter efficiency in deep neural networks. Their design philosophy of densely connecting layers has led to improved performance and training stability in modern convolutional architectures.
- Recurrent layers are used in [10, 186]. The core idea behind recurrent layers is to introduce connections that loop back on themselves, allowing information to be passed from one step to the next. This enables the network to maintain some form of memory or context as it processes sequential data.

Certain methods employed lightweight networks, either in a cascaded setup or within a Laplacian pyramid framework, with the goal of improving computational efficiency [187, 188]. However, this compromise led to a decline in deraining performance. In contrast, deep networks are intricate, complicating individual module analysis.

Listed below are baseline deraining networks used for single image deraining and have achieved significant performance on rainy datasets.

i. PRN and PreNet

In 2019, Ren et al. [141] established a straightforward foundation for deraining networks. Their innovation involved a novel network design that systematically unravelled a shallow Residual Network (ResNet). They harnessed the potential of recursive computations by employing a progressive ResNet (PRN). Furthermore, they introduced recursive layers that

capitalized on the interdependencies of deep features across stages (PreNet). Given its simplicity, efficiency, and efficacy, this model serves as a fitting baseline for forthcoming deraining studies.

ii. MSPFN

In 2020, Jiang et al. [180] tackled the issue of removing rain streaks from single images. They explored the concept of multi-scale collaborative representation for rain streaks, considering various input image scales and hierarchical deep features within a unified framework. They captured the overall textures of comparable rain streaks located at different positions by employing recurrent computations.

iii. MPRNet

In 2021, Zamir et al. [189] introduced a multi-stage network that follows a progressive approach to learn the restoration of degraded images. This strategy involves breaking down the entire process into smaller, more manageable sub-processes. Their framework incorporates sequential information exchange from early to late stages alongside lateral connections connecting feature processing blocks to preserve comprehensive information. This network addresses various image restoration tasks such as denoising, deblurring, and deraining. In the initial stages, encoder-decoders are utilized to extract multi-scale contextualized features. Subsequent stages operate at the original image resolution to produce spatially accurate outputs. Furthermore, they incorporated a Supervised Attention Module (SAM) between each pair of stages to refine features before transmitting them to the subsequent stage.

iv. HINet

In 2021, Chen et al. [190] introduced HINet, a robust and straightforward multi-stage network design. HINet comprises two subnetworks, both adopting the U-Net architecture [191]. A novel element, the Half Instance Normalization (HIN) block, was incorporated. The network's performance excelled over state-of-the-art methods across various image restoration tasks, including denoising, deblurring, and deraining.

2.6 Summary

This chapter provided a brief background on autonomous driving and its current challenges. Nowadays, most vehicles have different autonomous capabilities onboard, such as lane detection and parking assistance. On the other hand, both research and industry are moving

towards reaching fully autonomous vehicles (Level 5), which are capable of operating under any operational driving domain without the need for human interference.

Different types of sensors were studied (*Camera*, *LiDAR*, *RADAR* and *Ultrasonic*), and compared to highlight each sensor's advantages, limitations, and applications. Moreover, a review of the related research applying sensor fusion on different sensor combinations was made. In the proposed study, objects detection process is the primary concern as it comes ahead of other tasks performed by AVs. Objects detection should be performed while maintaining AV-related requirements, such as real-time performance, detection of multiple and dynamic objects, and reliable performance under different weather conditions.

Moreover, due to the significant effect weather conditions impose on the objects detection operation, different rain degradation types are elaborated, showing their effect on images. Also, existing baseline research towards rain detection and removal has been elaborated and compared while highlighting their limitations.

On the other hand, much research has been made towards enhancing the performance of the objects detection process; some approaches reached remarkable accuracy by employing expensive sensory systems while integrating them with powerful computational hardware. However, these systems impose high computational requirements; otherwise, they will slow down the whole system. Moreover, objects detection is a primary step before many others. Hence, the computational resources are shared among many different operations in addition to objects detection. Therefore, future research should aim to reduce the complexity and computational requirements of autonomous driving sub-operations. Other less expensive approaches exist; however, they still have their limitations, such as slow running time, detecting specific objects, and not considering the dynamic behaviour of both the surrounding objects and the ego vehicles themselves.

In this thesis, a real-time 3D objects detection framework is proposed which addresses many of the current systems' limitations while maintaining low computational requirements. Moreover, the visual objects detection operation is boosted by proposing a deraining network in order to maintain reliable objects detection performance in rainy weather conditions. The following chapters will demonstrate the contributions of this thesis while comparing the proposed work to related work in the field.

Chapter 3 Visual Rain Streak and Accumulation Removal using Artificial Intelligence

3.1 Introduction

Object detection stands as a foundational task for AVs, preceding activities such as object tracking, trajectory estimation, and collision avoidance. It involves the process of identifying and precisely localizing objects while determining their respective classes. Environmental perception is accomplished through the utilization of suitable exteroceptive sensory systems. Exteroceptive sensors encompass a range of devices such as monocular and stereo cameras, short- and long-range RADARs, ultrasonic sensors, and LiDARs.

In this thesis, a cost-effective approach is taken by combining a single-beam (2D) LiDAR with a monocular camera to achieve comprehensive 3D detection of dynamic objects in real driving scenarios, all while maintaining real-time processing capabilities. This study also addresses challenges encountered during dynamic object detection, such as complex object interactions leading to occlusion and truncation, as well as the dynamic variations in perspective and bounding box scales. Object detection from images relies on a deep-learning detector (YOLOv3), while LiDAR measurements undergo pre-processing and clustering. The system's output includes object classification and localization within bounding boxes, accompanied by a third dimension of depth information obtained from the LiDAR measurements. A schematic of the proposed approach is shown in Figure 3.1.

However, when object detection frameworks (image-based) are applied to unclear or distorted images, their performance greatly degrades. The reason behind this is that distortions (such as rain) cover important features that are needed by object detection frameworks to perform feature extraction.

Weather conditions that lead to image distortions can be categorized as either steady or dynamic, as outlined by Garg and Nayar [137]. Steady weather conditions encompass phenomena like fog, mist, and haze, while dynamic weather conditions comprise instances of rain and snow. Due to their larger and constantly changing droplet patterns, the dynamic weather conditions impose more substantial impairments on scene visibility. These conditions result in diminished scene clarity and contrast, thereby causing a notable decline in the effectiveness of object detection by AVs.

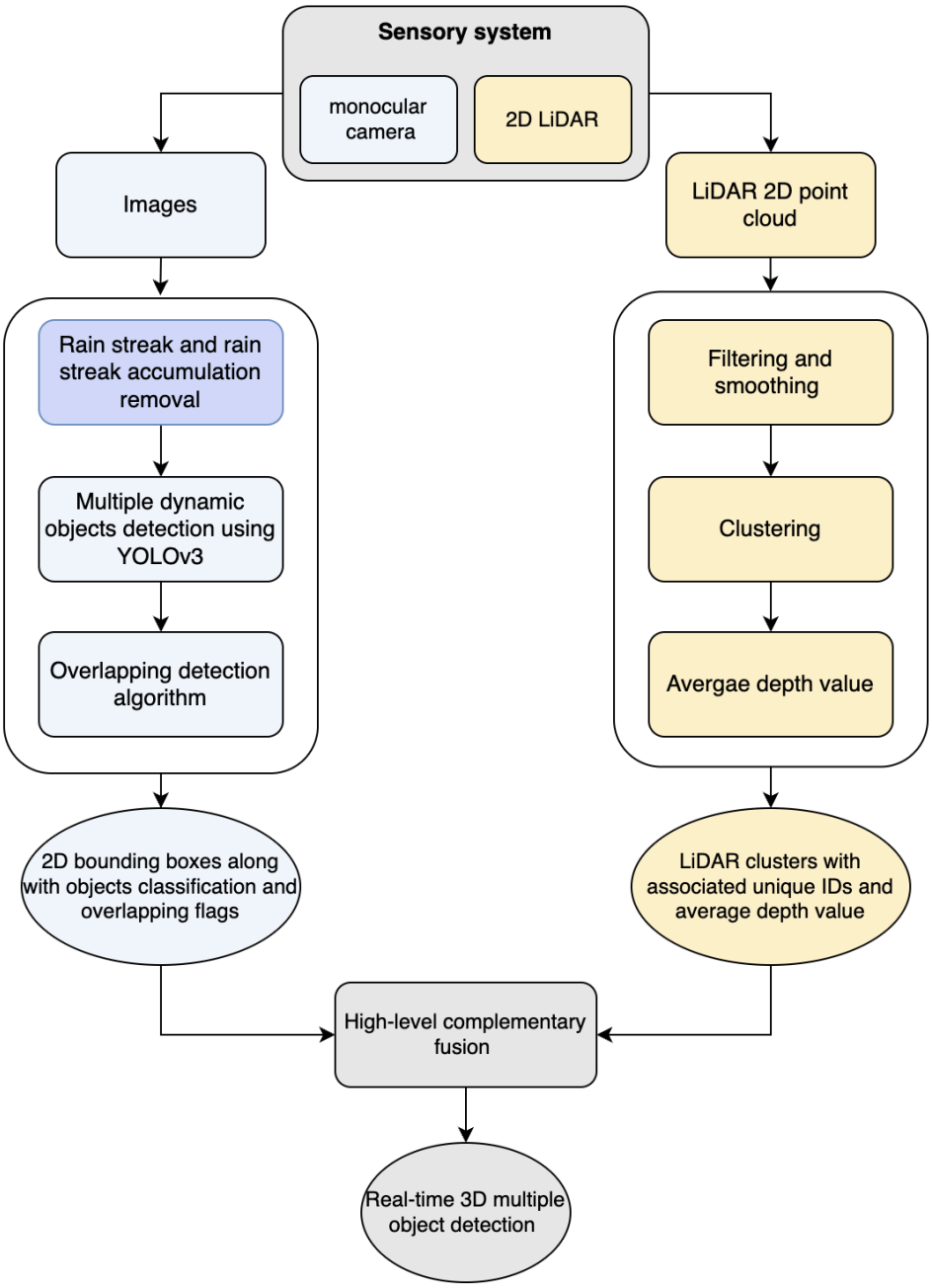

Figure 3.1: A schematic of the whole system

Rain is the most common dynamic challenging weather condition, especially in the United Kingdom [138], as the annual average rainfall percentage in the UK is around 33.7% [192]. Even though many researchers have achieved advancements in developing deraining methods, real-time video deraining for AV tasks has been largely understudied.

Rainfall can be defined as the existence of numerous water droplets characterized by diverse sizes, intricate forms, and fluctuating velocities. It randomly spreads with varying speeds over road elements (i.e., roadways, pavements, vehicles, pedestrians, cyclists, animals, etc.). Due to the high velocity of raindrops, their perspective projection forms rain streaks [193].

The presence of raindrops in captured video frames causes pixel intensity variations, as they block the light reflected by objects in the driving scene. Generally, rain streaks cause lower contrast and more whitened visual data [5].

As discussed in the Literature Review in Section 2.5, rain causes two different forms of visibility degradations. Rain streaks distort the images by producing specular highlights and distorting background scene features. On the other hand, rain streaks may gather and produce a hazy effect that resembles fog, hence decreasing the scene's vision [139]. Rain streaks and rain streak accumulation have different characteristics, and as a result, the distortions they cause to images are different. Researchers eventually deal with them as two distinct issues. This has encouraged the research towards developing a framework that mitigates both rain degradation types without negatively affecting the sequential object detection tasks.

Therefore, rainfall influences camera-captured videos and detections; the performance of object detection is degraded due to the degraded image quality. As a result, mitigating the effect of rain on the object detection framework is one of the most critical tasks that AVs should perform to reduce the gap towards achieving Level 5 autonomous driving (Automation under any ODD). Moreover, it is considered a challenging task due to:

- Real-time performance constraints (computation speed and complexity)
- Different rainfall characteristics
- Dynamic driving scenarios
- Further tasks (e.g., object detection) depend on the output of the deraining process.

Some studies have shown that many deraining algorithms [9, 194] actually degrade the detection performance compared to directly using the rainy images as input into the corresponding detection frameworks as they may over-smooth the input images, which distorts meaningful features in a scene, especially edges of objects [5]. Moreover, most deraining algorithms have been designed and tested using synthesized rainy datasets or mitigated a single rain degradation effect. Also, some of the SOTA deraining networks are heavily weighted, which leads to higher computational requirements and furthermore hinders low-cost real-time performance.

Current deraining algorithms suffer from unclear and unsatisfactory aspects [173], including but not limited to (*i*) rain modelling is over-simplified, i.e., considering and evaluating with a single type of rain degradations, (*ii*) limiting evaluation to synthetic images, which lacks

the complexity and characteristics of real rain, (*iii*) many algorithms have not been evaluated metrics related to human perception quality [195] or computer vision [208].

In this chapter, a lightweight multi-stage network for single-image deraining is presented; it mitigates the two main types of rain degradations in different directions, orientations, intensities, and accumulations. The proposed approach utilizes different progressive stages to reliably restore images affected by different rain degradations while maintaining low inference time. Lightweight networks have fewer parameters and less computation [196].

3.2 Overview of Residual Networks (ResNets)

Due to the vanishing gradient problem, training Deep Neural Networks (DNNs) is a challenging task; repeated multiplication that occurs when the gradient is backpropagated reduces the gradient, making it infinitely small. As a result, the deeper the network, the more saturated its performance becomes and the faster the degradation occurs [197]. The main idea of Residual Networks (ResNets) is using "identity shortcut connections" or "skip connections" that skips one or more layers [179], as shown in Figure 3.2. The addition between the output of a layer and its input improves the training of deep networks as these skip connections easily navigate gradients through them, resulting in faster training and higher accuracy from deep networks [9, 194]. Typical ResNets are implemented with double- or triple-layer skips that contain nonlinearities (ReLU) and batch normalization in between.

3.2.1 Progressive Residual Networks

In [141], Ren et al. have recursively unfolded a shallow ResNet with five Residual Blocks (ResBlocks) into multiple stages, taking advantage of recursive computations without increasing the network parameters; they introduced the Progressive Residual Network (PRN). In addition, they introduced a recurrent layer forming a Progressive Recurrent Network (PreNet) in order to employ the dependencies of deep features across stages.

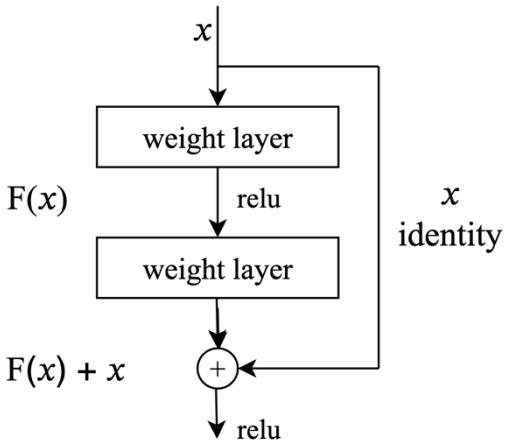

Figure 3.2: Residual learning: ResBlocks [179]

3.3 Methodology

Figure 3.3 illustrates the suggested deraining framework, which consists of the removal of (*i*) rain streaks and (*ii*) rain accumulation. Because the deterioration of rain accumulation is multiplicative and its removal will adversely affect the rain streak removal stage, generating more significant errors, separate training for the two stages has been performed. Thus, end-to-end training would have caused contamination of the recovery of rain streaks, which would have a more significant impact on the captured visual data [139]. On the other hand, both phases were jointly implemented during the testing phase.

When distant rain streaks accumulate, they produce an atmospheric veiling effect that is comparable to the degradation caused by fog, as shown in Equations 3.1 and 3.2 [198]. Therefore, rain accumulation removal can be achieved using techniques similar to defogging and dehazing. Removal of rain accumulation as an initial step negatively affects the whole deraining process, as it may lead to strengthening the appearance of existing rain streaks; all rain streaks, even already sharp and clearly visible ones, are boosted; eventually, they look different than those present in the datasets, either synthetic or real [198]. Therefore, in the proposed organization, another rain streak removal stage was re-applied after rain accumulation removal, as the rain accumulation removal stage will cause the rain streaks that may be missed in the first stage of rain streak removal to be more apparent.

$$I(x) = J(x)\alpha(x) + A\big(1 - \alpha(x)\big), \qquad 3.1$$

$$J(x) = \left(B + \sum_{t_r=1}^{s} \tilde{S}_{t_r} \circ R \right) \quad 3.2$$

Where $I(x)$ is the observed hazy image at pixel location x; J is the actual scene radiance; α is the transmission map; A is the global atmospheric light (indicating the intensity of a light source at an infinite distance); B is the background layer; $\tilde{S}_{t_r}$ represents a layer of rain streaks having the same direction; t_r is the index of rain-streak layers; s is the maximum number of rain-streak layers; and R is a region-dependent variable that indicates the locations of individually visible rain streaks; finally, $\circ$ is an element-wise multiplication operation.

The initial stage of the suggested framework involves conducting rain-streak elimination. This is succeeded by eliminating rain accumulations and, ultimately, applying rain-streak removal once again. This sequence is founded on the insight that the phase of rain accumulation removal brings out the presence of less conspicuous rain streaks, which might have been overlooked during the initial round of rain-streak removal. The diagram of the suggested deraining framework is illustrated in Figure 3.3.

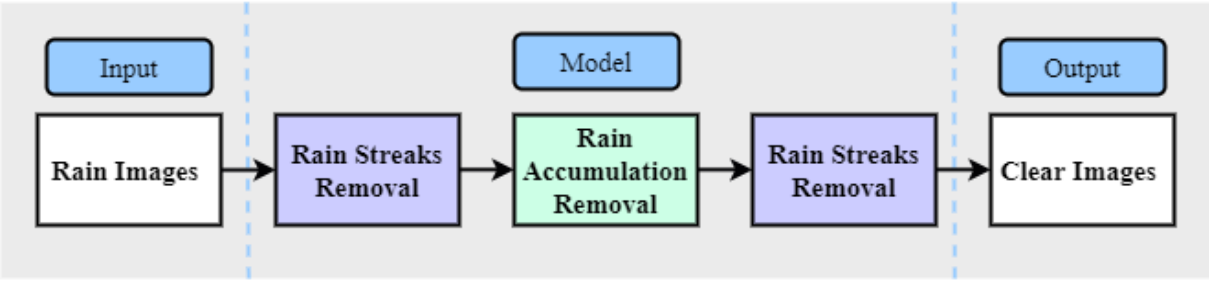

Figure 3.3: Proposed deraining framework

3.3.1 Rain Streaks Removal

The procedure for removing rain streaks relies on Residual Networks (ResNets), as introduced by He et al. [179]. This is done to preserve real-time efficiency without contributing to the computational intricacy of the object detection framework. The fundamental design of progressive residual networks has been embraced. Hence, instead of employing deeper and more intricate networks, the rain streak removal unfolds in a series of stages, with ResNet being employed in each stage. Furthermore, to prevent a surge in the count of network parameters and the vulnerability to overfitting, the approach involves recursive computations achieved by sharing the same network parameters across multiple stages.

A basic ResNet has three parts: (*i*) a convolutional layer f_{in}, (*ii*) multiple ResBlocks f_{res} to extract deep representation, and (*iii*) a convolutional layer f_{out}, as shown in Figure 3.4. Inference of the Progressive Residual Network (PRN) at stage t can be formulated as shown in Equations 3.3 and 3.4:

$$x^{t-0.5} = f_{in}(x^{t-1}, y)\,, \tag{3.3}$$

$$x^{t} = f_{out}(f_{res}(x^{t-0.5})) \tag{3.4}$$

Network parameters are reused across different stages, as f_{in}, f_{res}, and f_{out} are stage invariant. As shown in Equation 3.3, f_{in} takes the concatenation of the current estimation x^{t-1} and rainy image y as input, the inclusion of y further improves the deraining performance.

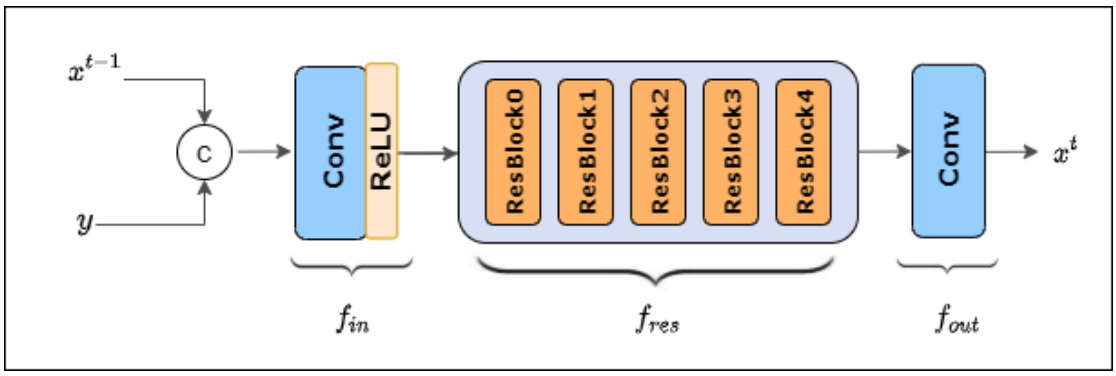

Figure 3.4: A basic ResNet

i. Network Architecture

The progressive network architecture is shown in Figure 3.5:

- Filter sizes, 3 × 3, Padding 1 × 1;
- f_{in} is a convolutional layer with ReLU non-linearity [199]. Due to the concatenation of three channels from y and another three channels from x^{t-1}, the convolution of f_{in} has six input channels and 32 output channels;
- f_{res} includes five ResBlocks;
- f_{out} is a single-layer convolution; it takes the output of f_{res} with 32 channels as input and outputs a 3-channel RGB image.

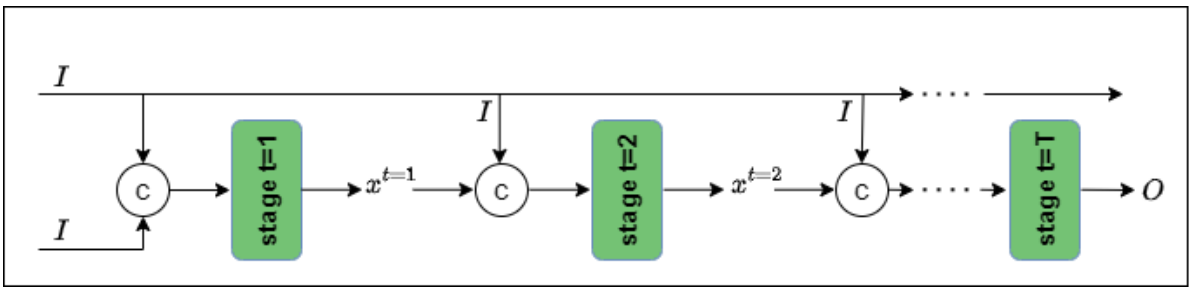

Figure 3.5: Progressive network architecture

ii. Loss Function

Although hybrid loss functions have been recently widely used to train deraining networks (e.g., MSE + SSIM, ℓ_1 + SSIM, and adversarial loss), they increase the burden of hyper-parameter tuning. As shown by Ren et al. [141], a single loss function, especially single negative SSIM, is sufficient to train PRN benefiting from the progressive nature architecture.

For a model having multiple stages (T), with a multiple number of outputs (x^1 , x^2, …, x^T), by only imposing supervision on the output of the final stage x^T [200]. MSE can be presented as:

$$\mathcal{L} = \|x^T - x^{gt}\|^2 \quad 3.5$$

Where, x^{gt} is the corresponding ground-truth clean image.

On the other hand, the negative SSIM loss can be represented as:

$$\mathcal{L} = -SSIM(x^T, x^{gt}) \quad 3.6$$

3.3.2 Rain Accumulation Removal

Rain streaks and the accumulation of rain have distinct visual distortions on captured images. Distant rain streaks accumulate, creating a distortion akin to fog. This fog-like effect not only degrades visibility but also conceals image details and the features of existing objects. Consequently, removing these distant rain streaks becomes imperative. Considering that the distortion caused by rain accumulation resembles that of fog (as represented by Equation 3.1), similar algorithms can be employed for the removal of rain accumulation. For instance, techniques such as those outlined in references (e.g. [139, 198, 201]).

i. Network Architecture

The rain accumulation removal network incorporates a lightweight neural network termed Transmission-Guided Light-Weight Network (TGL-Net), as put forth by Li, Zhan et al. in 2021 [202]. This network autonomously computes guided transmission maps instead of generating synthetic transmission maps using depth data and haze-free images. These maps are computed using the filter-refined dark-channel-prior (F-DCP) technique. This strategy empowers the network to be trained on both synthetic and real images without requiring extra manual adjustments or acquiring transmission information.

The architectural design of TGL-Net draws inspiration from the Very Deep Residual Encoder-Decoder Network (RED-Net) introduced by Mao et al. in 2016. While RED-Net employs a symmetric configuration with skip connections, TGL-Net deviates by featuring fewer layers and parameters, rendering it notably lightweight. TGL-Net comprises three main stages: downsampling, encoder-decoder, and upsampling, all of which are visually represented in Figure 3.6; skip connections are shown as dotted lines [202].

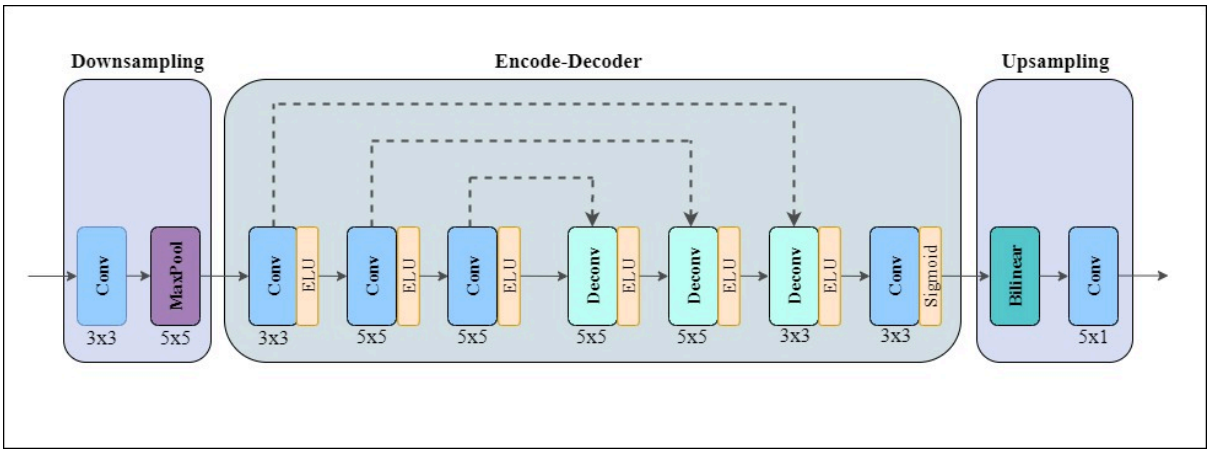

Figure 3.6: TGL-Net's symmetric architecture, comprised of three stages: downsampling, encoder-decoder, and upsampling [202]

Downsampling

The initial downsampling stage encompasses a convolutional layer (3 × 3) that is succeeded by a subsequent max-pooling layer. The role of the convolutional layer is to extract features from the images, after which the max-pooling layer reduces the total number 25 times by using a stride of 5. This optimization significantly enhances the computational efficiency of the network.

Encoder-Decoder

During the encoder-decoder phase, a set of three convolutional and three deconvolutional layers are interconnected via a condensed encoder-decoder connection. This stage serves the purpose of both feature extraction and transmission estimation. Within the encoder, the convolutional layers are responsible for extracting additional image features while simultaneously eliminating noise. Conversely, the deconvolutional layers restore the intricate details of transmission maps in the decoder.

In the encoder, the dimensions of the convolutional kernels follow a sequential pattern of 3 × 3, 5 × 5, and 5 × 5. These dimensions are mirrored in the decoder's corresponding deconvolutional kernels. Furthermore, zero padding has been incorporated to maintain consistency in the sizes of the output feature maps across each layer.

The arrangement of symmetrically connected convolutional and deconvolutional layers remains consistent with the original RED-Net [203]. This choice leverages the advantageous characteristics of residual networks as outlined by He et al. [179]. The incorporation of skip connections, which directly transmit signals to subsequent layers, mitigates the prevalent issue of vanishing gradients often encountered in deep neural networks (DNNs), as described by Glorot and Bengio in 2010 [204]. Additionally, the utilization of skip connections contributes to the preservation of valuable image features and intricate details.

In order to maintain a reliable computational efficiency, TGL-Net has a reduced encoder-decoder number of layers; it has a six-layer encoder-decoder phase compared to 20 or 30 layers in RED-Net.

Furthermore, an Exponential Linear Unit (ELU) activation has been implemented for each encoder and decoder layer. In contrast to the Rectified Linear Unit (ReLU), ELU offers the advantage of avoiding an excessive number of "dead nodes" while still preserving the positive linear attributes of ReLU activation, thus mitigating the vanishing gradient problem, as depicted in Figure 3.7. Additionally, the mean output value of ELU is proximate to zero, promoting accelerated convergence during training, as indicated by Clevert et al. in 2015 [205]. Consequently, the ELU function serves as the activation mechanism for the encoder and decoder layers within TGL-Net.

Subsequent to the encoder-decoder layers, a single convolutional layer is employed, which combines a three-channel feature map into a single-channel predicted transmission map.

Additionally, a non-linear Sigmoid activation function has been utilized. This sigmoid function enters a saturation region when the input reaches extreme values (larger than 5 or smaller than -5), potentially intensifying the vanishing gradient risk when more layers are added. However, the utilization of ELU units and skip connections within the encoder-decoder layers effectively addresses the vanishing gradient problem. The non-linear sigmoid function ultimately generates a preliminary estimate of a reduced transmission map that is smaller than the input image 25 times.

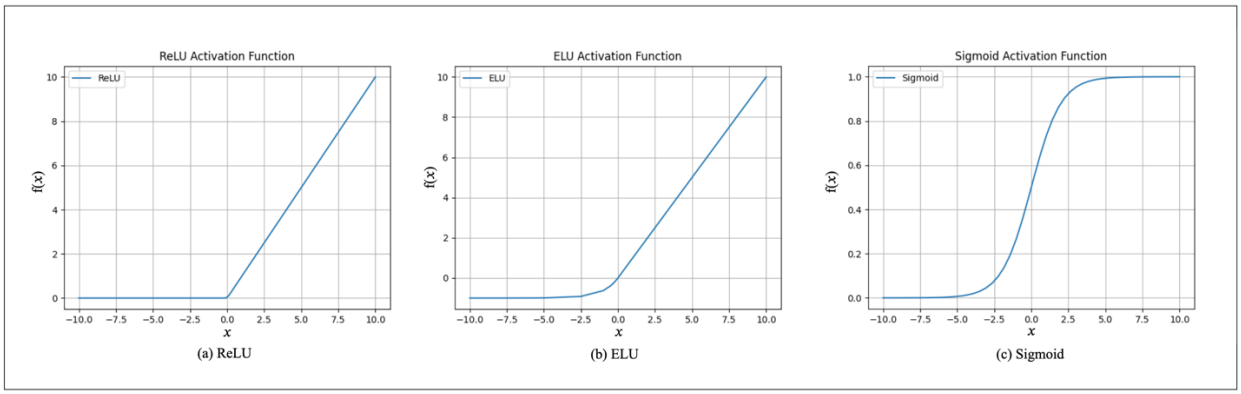

Figure 3.7: Activation functions

Upsampling

The upsampling phase is introduced to match the dimensions of the input image. This upsampling phase encompasses a dual-step process. Initially, image enlargement is executed through bilinear interpolation, a measure undertaken to uphold the network's computational efficiency. Subsequently, a convolutional layer is employed to further enhance the transmission's map output quality.

ii. Loss Function

In order to combine the errors of both the transmission maps and the outputs, a double-error loss function was used; it was proposed by Li et al. in [202]. The double-error loss function makes use of the errors of both a transmission map and clear output to supervise the training of the network. As a result, prior information is introduced, which achieves output images with enhanced details and richer information.

3.4 Summary

Within this chapter, a novel deraining network designed to address a diverse range of rain-related issues prevalent in images taken under rainy driving conditions is introduced. This network is devised to alleviate distinct rain effects, encompassing varied rain streak orientations, intensities, and rain accumulations. Comprising three primary phases, the proposed approach strategically tackles these challenges. The initial phase is dedicated to rain streak removal, targeting the elimination of the most conspicuous rain streaks within the image. Subsequently, the second phase focuses on rain accumulation removal, which is similar to a defogging process. Finally, the third phase involves a second rain streak removal operation, which removes any remaining rain streaks that may have been intensified by the prior rain accumulation removal phase.

The foundation of the network relies on fundamental lightweight architectures with few parameters to ensure real-time computational efficiency. This consideration is of utmost importance due to the stringent requirement for prompt processing in deraining networks, as well as in other subsequent operations conducted by autonomous vehicles (AVs)

This work has been submitted and published in the *Applied Sciences* Journal as "*A Lightweight Network for Real-Time Rain Streaks and Rain Accumulation Removal from Single Images Captured by AVs*".

Chapter 4 Real-time Object Detection Using a Multi-Sensory System

4.1 Introduction

Among the complexities, dynamic road objects like pedestrians and cyclists present a substantial challenge due to their dynamic behaviour. The operational reliability of object detection, crucial for AVs, is mainly tested in this context. The prevalent approach in both commercial AVs and research endeavours involves the utilization of costly sensors. Nonetheless, this reliance on expensive sensors could potentially impede the advancement of AV research and capabilities.

These detection techniques yield bounding boxes encapsulating the detected entities, accompanied by predicted class labels and associated confidence scores [206]. Within the realm of autonomous driving, the task encompasses the recognition of various object categories, which can be categorized into two main groups: static and dynamic objects. Static objects encompass elements such as traffic lights, signs, buildings, bridges, and curbs, while dynamic objects comprise pedestrians, cyclists, animals, and various types of vehicles. Detecting static objects is generally more straightforward due to their well-defined shapes and relatively predictable positions.

Despite the substantial interest researchers have shown in LiDAR-based 3D detections, point clouds still lack the texture information necessary for effective object classification. Furthermore, point clouds encounter issues such as sparsity and reduced density when detecting objects at a distance.

4.2 Sensory System

In this thesis, a low-cost sensory system has been employed and a novel fusion algorithm has been developed in order to perform real-time 3D object detection. A monocular camera is used in addition to a 2D LiDAR; both sensors complement each other.

4.2.1 Monocular Camera

Cameras play a pivotal role as the primary vision sensors utilized for object detection due to two key factors: (*1*) their affordability, making them a cost-effective option for integration into AVs, and (*2*) their ability to capture intricate texture details. Nonetheless, monocular

cameras face limitations in lacking a third dimension crucial for precise object detection. While 3D object detection from captured images can be accomplished by extending the detected 2D bounding boxes through techniques like reprojection constraints or regression models, the accuracy of depth estimation in these calculations remains less dependable.

In this thesis a monocular camera is used as a vision sensor for detecting and localizing objects (in a 2D plane) in captured video frames. Different monocular cameras could be used, and a simple update in the system would be applied regarding field of view and resolution.

4.2.2 Two-Dimensional LiDAR

In order to add a depth dimension to detected surrounding objects, a LiDAR has been chosen. LiDARs come in two categories: 3D and 2D. 3D LiDARs project multiple laser beams along the vertical plane, providing a large amount of data; on the other hand, 2D LiDARs only project a single beam at the same vertical point. As discussed in Chapter 2, much research has focused on using 3D LiDARs as they provide the system with more information on the surrounding objects. However, they impose high costs and high computational requirements. Therefore, in this research, a 2D LiDAR has been employed to evaluate its performance when used along with a monocular camera.

Below are five essential considerations for choosing a LiDAR to operate on an AV:

- *Resistance to ambient light*: Intense sunlight can interrupt this process by preventing the LiDAR from reading its own returning light pulses. This could cause the LiDAR to malfunction, resulting in a loss of navigation. Therefore, a LiDAR with high sunlight resistance must be chosen for outdoor applications.
- *Resistance to Environmental Noise:* Environmental factors such as rain and snow can interfere with LiDAR's ability to accurately detect objects (i.e., leading the system to detect objects that are not there). Therefore, it is important to choose a LiDAR capable of maintaining a high level of accuracy even in adverse conditions. Some advanced LiDARs mitigate possible interferences using multi-echo technology, as shown in Figure 4.1, ensuring reliability regardless of weather conditions and minimizing false alarms, saving time and money. Multi-echo is an essential feature as part of the energy from the pulse of the LiDAR may be reflected by nearby environmental factors (such as rain and snowflakes). At the same time, the rest of the

beam reaches an obstacle and is reflected by it. LiDARs with multi-echo technology evaluate these multiple “echoes” and ignore the closer, weaker reflections caused by environmental factors. This eliminates the effect of noise.

- *High Environmental Rating:* When choosing a LiDAR to work outdoors, it must be resilient enough to have a longer lifetime and resist environmental factors (such as rain). A LiDAR with rugged construction and a high enclosure rating will help reduce Mean Time Before Failure (MTBF) and ensure the longevity of the navigational system.
- *Temperature Range*: Extremely cold or hot temperatures could damage sensors (for example, causing LiDAR’s housing to crack). Because of this, LiDARs with a wider operating temperature range and a built-in temperature control system are advised.
- *Electromagnetic Considerations*: Outdoor environments may have a wide variety of signal emissions, which could contain varying strengths and types of electrical noise. This noise can induce itself onto the sensors and circuits of the control system, which could cause the system to behave erratically. Although this factor is often overlooked in the early design process, electromagnetic noise can become problematic once AVs are deployed into real-world environments, especially for vehicles operating near substations or powerlines.

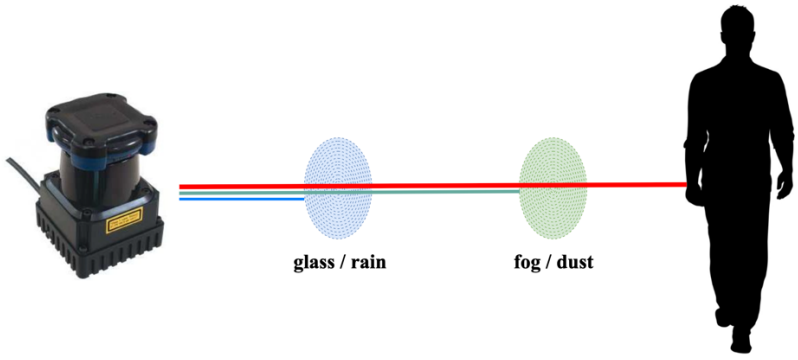

Figure 4.1: Multi-echo operation

According to the previously mentioned criteria for choosing a LiDAR, four LiDARs were considered good candidates, they are presented in Table 4.1.

Table 4.1 Comparison between three candidate LiDARs

	UXM-30LX-EW	**UTM-30LX-EW**	**UXM-30LXH-EWA**	**UTM-30LX**
Range	30m	30m	30m	30m
Field of view	190°	270°	190°	270°
angular resolution	0.25°	Approx. 0.25°	0.125°	Approx. 0.25°
Scan time	50ms/scan	25ms/scan	50ms/scan	25ms/scan
Multi-echo	✓	✓	✓	-

UTM-30LX-EW and UTM-30LX were the best candidates owing to their wider FOV and faster scanning time. Due to the availability, the LiDAR used in this research is the Hokuyo UTM-30LX (shown in Figure 4.2). It owns competitive features; however, it lacks a multi-echo feature. It is a 2D radial LiDAR that measures 1,081 distance points in a range from -135° to 135°, where orientation 0° corresponds to the front of the sensor, as shown in Figure 4.3. It takes 0.025 seconds/scan when operating in the continuous acquisition mode. The LiDAR has a resolution of 1mm in a range from 0.1 to 60m and requires a dedicated power source (12V, 1A) to operate correctly. More specifications are listed in Appendix B.

Figure 4.2: Hokuyo UTM 30LX [207]

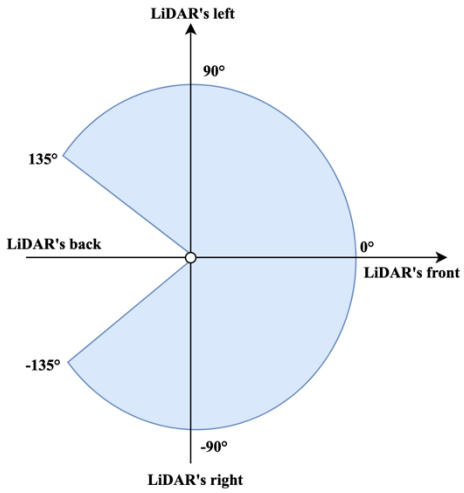

Figure 4.3: Hokuyo UTM 30LX angular range

4.2.3 Sensor Fusion

Relying solely on a single type of sensor has demonstrated its inadequacy and unreliability; thus, the integration of multiple sensors through sensor fusion becomes imperative to overcome these limitations. Through the utilization of multiple sensors, the process of sensor fusion enhances the trustworthiness and precision of measurements while diminishing their inherent uncertainty. To effectively harness the capabilities of the 2D LiDAR within diverse and challenging driving scenarios, where many distinct objects may potentially overlap and interact, the implementation of an overlapping detection mechanism becomes essential. This mechanism serves the purpose of identifying objects that are overlapping and forming clusters with one another. This algorithm plays a crucial role during the integration of LiDAR and camera data.

4.2.4 Sensors Placement

Sensors placement must consider the vehicles' ground clearance (i.e. ride height): it is the amount of space between the base of an automobile tyre and the lowest point (typically the axle) or, more appropriately, to the shortest distance between a flat-level surface, and the lowest part of a vehicle other than those parts designed to contact the ground (such as tyres, tracks, skis, etc.), as shown in Figure 4.4. Therefore, vehicle types are surveyed in order to estimate the average ground clearance of vehicles in the UK in order to place the LiDAR at a height that is between the maximum ground clearance and the minimum vehicle height (this is given in Appendix C). Therefore, it is concluded that the LiDAR's optimal height is

559mm away from the ground in order to be higher than the maximum ground clearance (maximum truck ground clearance is 559mm).

In the proposed setup, the camera system and the LiDAR must have a common horizontal centre point in order to be able to correctly map between the image pixels and LiDAR light beam angles.

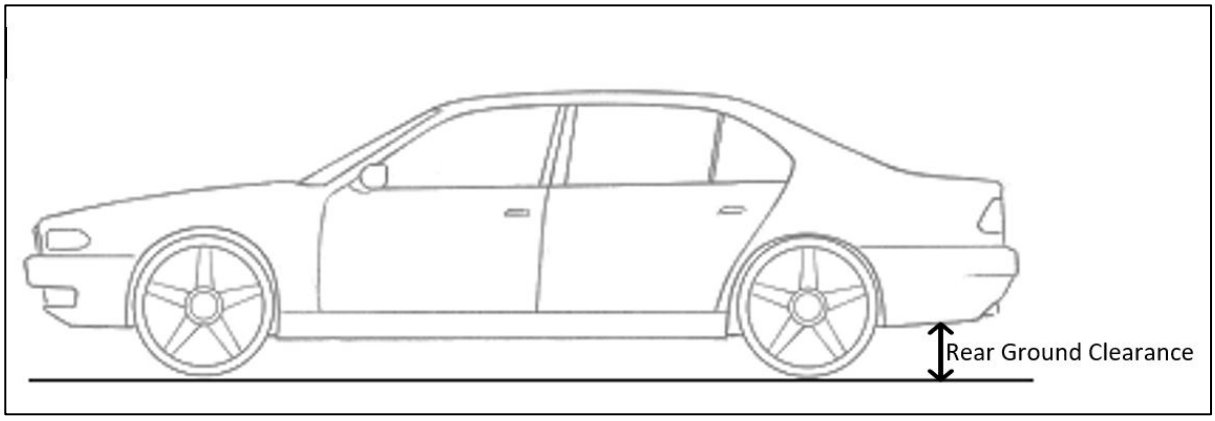

Figure 4.4: Vehicles ground clearance

4.3 Real-time Object Detection using Artificial Intelligence Techniques

4.3.1 Deep-Learning-Based Object Detection

Object detection techniques yield bounding boxes encompassing identified objects, accompanied by an anticipated class and a confidence score [206]. Various factors influence the selection of the appropriate object detection algorithm, and consequently, distinct driving scenarios introduce varying challenges to object detection. These challenges can include:

- Fluctuating weather and lighting conditions
- Reflective objects
- Differing object dimensions
- The obstruction and partial concealment of obstacles.

Table 2.5 summarises a comparison between the performance of the three main DL-based object detection approaches: CNNs, SSD, and YOLO. The table shows that there is a trade-off between the detection FPS and the mAP percentage. It can be observed that:

- SSDs achieve high mAP on high FPS; however, it is worth noting that SSDs perform poorly when detecting small/distant objects. This is because SSD detects small

objects only in higher-resolution layers. However, these layers contain low-level features, like edges or colour patches, that are less informative for classification.

- YOLO Tiny models achieve high-speed detections while degrading the mAP values.
- YOLOv2 and YOLOv3 achieve unparalleled speed while maintaining high mAP values on both VOC and COCO datasets [123, 124].

As a result of these comparisons, YOLOv3 will be employed as a real-time object detection network. YOLOv3 works upon Darknet, which is a neural network framework created by Joseph Redmon [208] (He is the author of these YOLO papers: [104, 123, 124]). Darknet is a framework to train neural networks. It is open-source, written by C/CUDA and serves as the basis for YOLO. The initial repository established by J. Redmon can be located in [209].

The YOLO algorithm is named "you only look once" because its prediction used 1×1 convolutions; this means that the size of the prediction map is exactly the size of the feature map before it. It is based on Convolutional Neural Networks (CNN) that can perform object detection in real time. It processes input images as a structured array of data and recognizes patterns between them.

The YOLOv3 architecture can be summarized as follows:

- Input: YOLOv3 takes an image input and divides it into a grid.
- Darknet: The input image goes through a backbone Darknet-based network.
- Multi-scale detection: YOLOv3 performs objects detection at three different image scales, and it generates bounding boxes and class probabilities at each scale.
- Bounding box prediction: YOLOv3 predicts multiple bounding boxes with associated confidence scores for each grid cell at each scale.
- Class prediction: YOLOv3 predicts the class probabilities for each bounding box, indicating the likelihood of an object being for a specific class.
- Non-Maximum Suppression (NMS): It only keeps non-redundant bounding boxes with the highest confidence scores for each detected object.
- Output: The final output is bounding boxes, each associated with a specific class and a confidence score.

YOLO performs at high speed while maintaining high detection accuracy. It looks at the whole image once at test time; therefore, its predictions are influenced by the full context of

the image. Different image regions are scored based on their resemblance to predefined classes.

YOLOv3 generates 2D bounding boxes in addition to the predicted class of the detected object. The model employed here has been pre-trained on the KITTI dataset, which is the most extensive dataset targeted for computer vision assessment related to autonomous driving.

4.3.2 Overlapping Detection

To ensure accurate usage of the 2D LiDAR across diverse and demanding driving scenarios and given the potential existence of multiple distinct objects that might overlap and interact (as shown in Figure 4.5), the implementation of an overlapping detection algorithm was imperative. This algorithm identifies objects that are in overlapping proximity and groups them into clusters. This algorithm's utilization will be during the integration of LiDAR and camera data, enhancing the precision of object location updates that necessitate distance computations. Multiple challenges face the overlapping checking process:

- The detection is performed in real-time scenarios, and the objects' number and locations are continuously changing,
- The regular overlapping checking operations cause higher computational complexity and time.
- There are multiple objects-overlapping scenarios: non-overlapping (Figure 4.6), overlapping between two or more objects in a single cluster (Figure 4.7), and the most common case in driving scenarios is overlapping between multiple objects in different clusters (Figure 4.8)

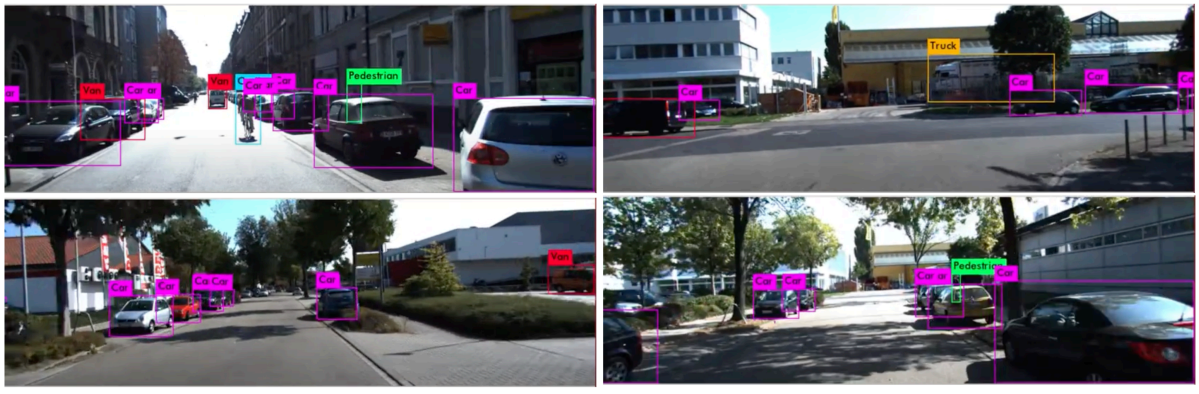

Figure 4.5: Overlapping samples from the KITTI dataset after YOLOv3 object detection

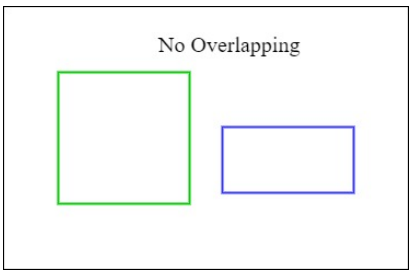

Figure 4.6: No overlapping scenario

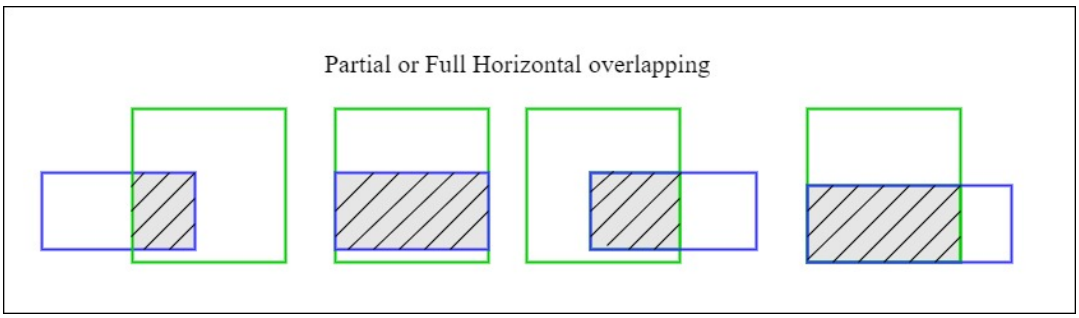

Figure 4.7: Partial and full overlapping scenarios

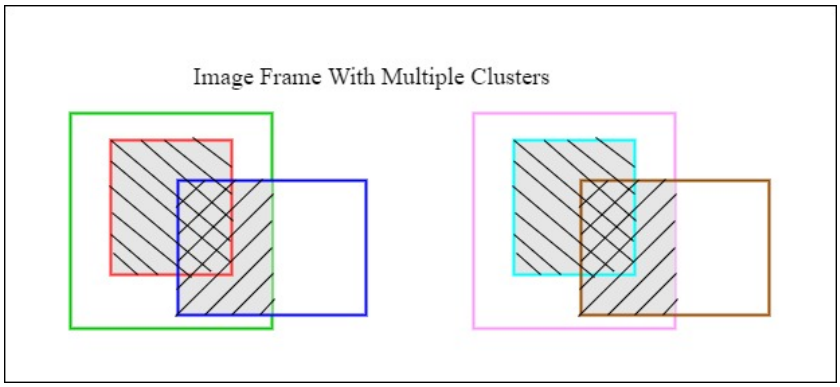

Figure 4.8: Overlapping in multiple clusters in the same image frame

In order to do the overlapping checking, the struct "*bbox_t*" is updated in order to include a vector of the overlapped horizontal pixels "*overlapped_pixels*" and another vector for the overlapped objects "*overlapped_objects*" and renamed it to "*bbox_ol_t*". The first step is to loop through the "*n*" detected objects at frame "*i*" in order to detect if the *n*'th object overlapped with any of the previous objects and update its related attribute of "*overlapped_pixels*" and "*overlapped_objects*".

4.4 LiDAR Data Processing

4.4.1 Conversion of Radial Measurements into Perpendicular Measurements

LiDAR works by calculating distances through angular rotations, resulting in radial measurements, as depicted in Figure 4.9 and Figure 4.3. To normalize the LiDAR measurements, it becomes necessary to convert these radial measurements into perpendicular measurements, as demonstrated in Equation 4.1.

$$\perp \text{ distance} = cos(\text{A}) \text{ x radial distance} \tag{4.1}$$

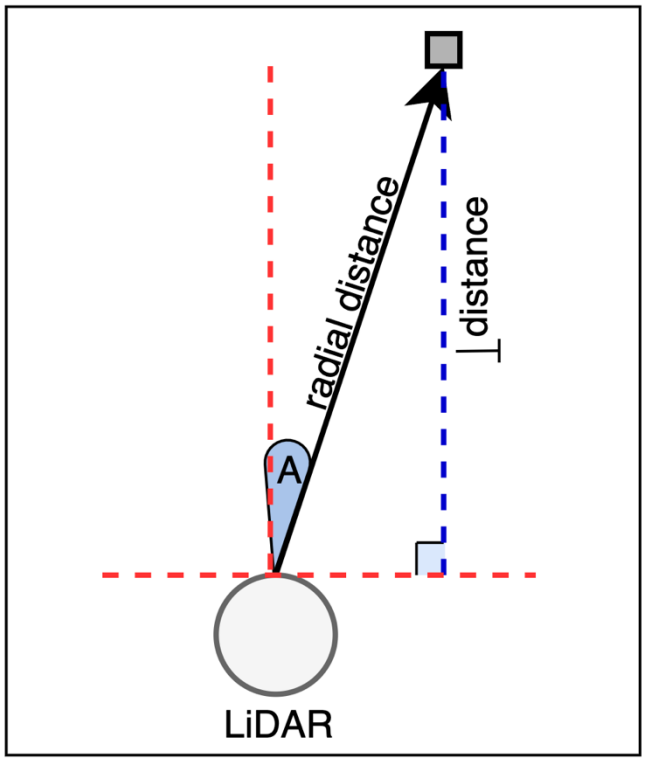

Figure 4.9: LiDAR radial to perpendicular measurements

4.4.2 Linearization and Smoothing

The data collected by the LiDAR for objects enclosed within bounding boxes cannot be directly regarded as a simple distance to be incorporated as a third dimension of the detected objects. This complexity arises from several factors, such as the irregular surfaces of the detected objects, uncertainties associated with the sensors, and the constant occurrences of overlapping and truncation among objects. Moreover, a phenomenon called mixed pixels occurs when discrete points appear between two objects that are horizontally alongside each

other [158]. It happens because the scanner received a part of the echo from one object and another part from the other object. Consequently, the data obtained from the LiDAR undergoes a smoothing and linearization process to provide a clearer comprehension of the objects present in the surroundings.

In order to apply depth measurements denoising, there are three suggested methods [210]: (*1*) spatial filters, (*2*) methods using depth and colour statistical information, and (*3*) segmentation-based methods. The most common spatial filters are Gaussian Low-Pass Filter [211], Median Filter [212], and Bilateral Filter [213]. On the other hand, the idea behind methods depending on depth and colour information is that discontinuities in colour typically correlate with discontinuities in range measurements. However, colour information is seldom available in LiDAR depth measurements. Finally, segmentation-based methods smooth regions in-depth images without impairing edges by segmenting the data and then performing local smoothing on each segment.

LiDAR measurements exhibit a Gaussian noise distribution with a variance of $\pm 3cm$. Consequently, employing a filter with a Gaussian impulse response function emerges as a suitable choice for noise reduction. In real-world scenarios, the surfaces of objects encompass both flat and rough areas as well as edges. Given the sparse nature of LiDAR measurements for distant objects, object edges manifest as disruptions in the LiDAR measurements, whereas flat surfaces exhibit gradual value changes. This underscores the necessity for an edge-preserving filter in the depth measurements reconstruction. To ensure real-time efficiency, a median filter was employed to carry out the filtering process while maintaining the integrity of edge features.

4.4.3 Grouping of LiDAR Measurements into Clusters with Unique IDs

Different studies have addressed the problem of object segmentation on LiDAR point clouds [60, 63, 64, 214, 215]. Nonetheless, their approaches either relied on 3D point clouds or operated under the assumption that the environment comprises distinct objects that do not overlap physically. In the proposed study, the challenge of multiple dynamic objects interacting and overlapping with one another is confronted.

Once the LiDAR measurements are refined through filtering, the subsequent step involves clustering, where similar neighbouring data readings are grouped together and assigned a

unique identifier accompanied by an average distance value. Two primary variables dictate this process:

- Minimum cluster size: To prevent the generation of numerous unnecessary mini clusters that might delineate subregions of objects, experimentations with various cluster sizes were made. Smaller cluster sizes led to an increased occurrence of false clusters.
- Threshold for difference: This parameter establishes the boundary between consecutive clusters, indicating the disparity that defines the separation between them.

4.5 Camera and LiDAR Fusion

4.5.1 Mapping between Image and LiDAR Coordinates

The results produced by the video-processing module encompass two essential components:

- Two-dimensional bounding boxes are superimposed upon the image pixels.
- Object classes attributed to the detected objects.

Establishing a correspondence between image pixels and real-world angular coordinates is imperative to facilitate the integration of sensor data between pixels (bounding boxes) and LiDAR measurements. Given the utilization of a 2D LiDAR, the focus remains on the horizontal plane (x-axis), as the LiDAR maintains a constant vertical value.

Leveraging the camera pinhole model illustrated in Figure 4.10, a functional mapping was formulated to translate image pixels into angular rotations. This function operates on the following inputs:

- pixel x-coordinate ($xpixel$)
- Frame width ($FrameWidth$)
- Horizontal field of view of the camera ($HFOV$)

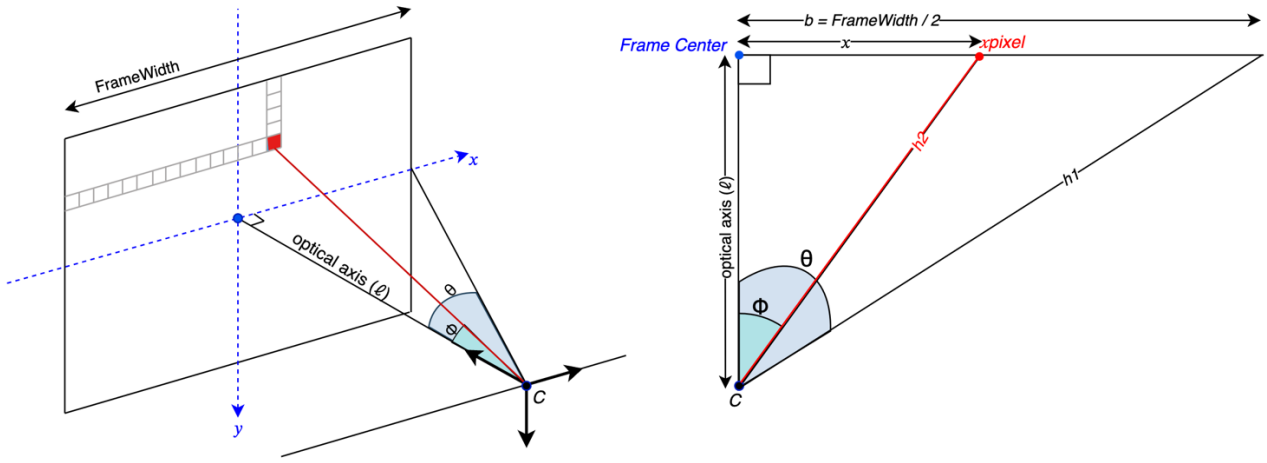

Figure 4.10: Conversion of pixel values into real-world angular coordinates

Considering the optical axis (ℓ), which is the direct line connecting the camera (C) and the centre of the frame, it's possible to establish two right-angled triangles:

1. The hypotenuse ($h1$) extends from the camera to the image's edge, forming an angle (θ) with the hypotenuse and (ℓ).
2. The hypotenuse ($h2$) runs from the camera to the position of ($xpixel$), resulting in an angle (ϕ) with the hypotenuse and (ℓ).

Within this configuration, the calculation of angle (ϕ) becomes necessary. The following trigonometric calculations are employed to calculate the angle (ϕ):

$$b = FrameWidth\ /2 \tag{4.2}$$

$$\theta\ =\ HFOV/2 \tag{4.3}$$

$$x = xpixel\text{-}FrameCentre \tag{4.4}$$

$$tan(\theta) = b/\ \ell \tag{4.5}$$

$$tan(\phi) = x/\ \ell \tag{4.6}$$

Making use of common (ℓ) in both Equations 4.5 and 4.6, both are solved for (ℓ), namely:

$$\ell = b/tan\ (\theta) = x/tan\ (\phi) \quad 4.7$$

$$\phi = tan^{-1}\ ((x\ tan(\theta))/b) \quad 4.8$$

This procedure is executed for both the left and right x-coordinates of each bounding box to transform the horizontal pixel values of the bounding boxes into angular coordinates in the real world.

4.5.2 Complementary Camera and LiDAR Fusion

Merging video and LiDAR data directly might seem straightforward, resulting in a horizontal line of pixels with associated distance measurements. However, the goal here is to link depth measurements with the 2D bounding boxes. Consequently, when the fusion is applied between bounding boxes and LiDAR measurements, the outcome comprises bounding boxes with linked distance measurements. Nonetheless, this task comes with a significant challenge: objects often overlap, causing the bounding boxes to intersect. As a result, the pixel range defined by a single bounding box could correspond to LiDAR measurements of multiple objects, as shown in Figure 4.11.

Furthermore, Figure 4.12 depicts a block diagram showcasing the fusion procedure between the video and LiDAR data processing modules. Both modules work separately and then complement each other; hence, for the system to operate reliably, both modules should be working and complementing each other as each module outputs different required data. The camera module outputs the object's class and its horizontal and vertical location, while the LiDAR provides the depth dimension.

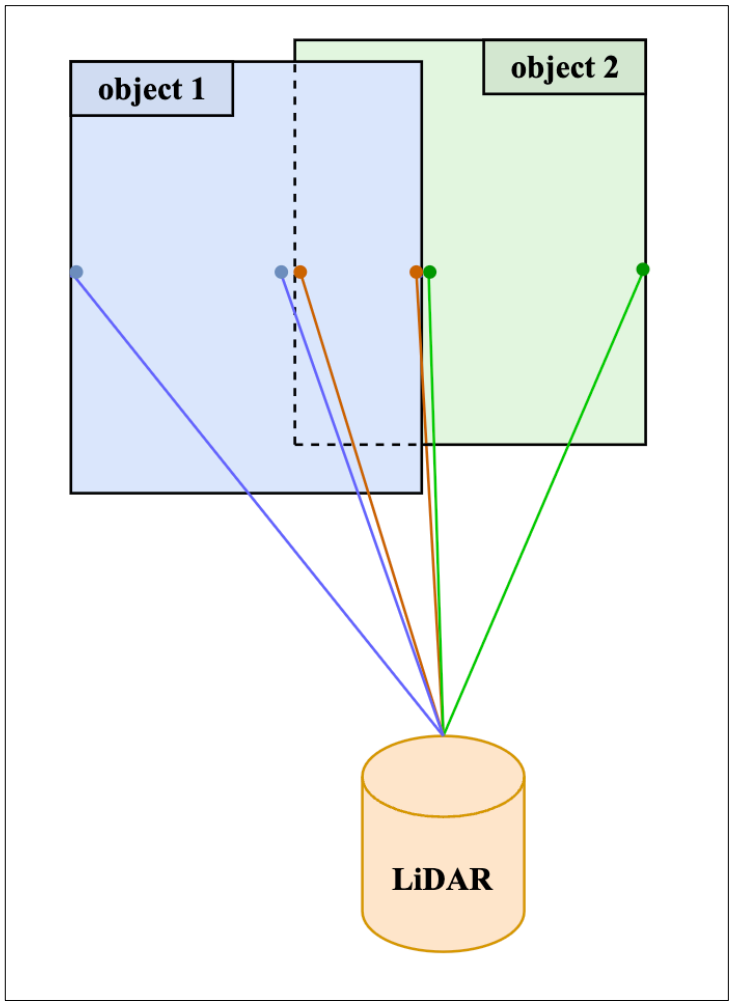

Figure 4.11 A scenario including two overlapping objects

During this stage, the process of sensor fusion takes place, involving the interaction between the bounding boxes generated by the video processing module and the clusters produced by the LiDAR data processing module. Several scenarios involving object overlapping need to be addressed:

- No overlapping.
- Object 'x' is entirely situated in front of object 'y' (object 'x' is smaller than object 'y') (as illustrated in Figure 4.13(a))
- Object 'x' is partially in front of object 'y' (as depicted in Figure 4.13(b)).
- Object 'x' is partially behind object 'y' (as shown in Figure 4.13(c)).

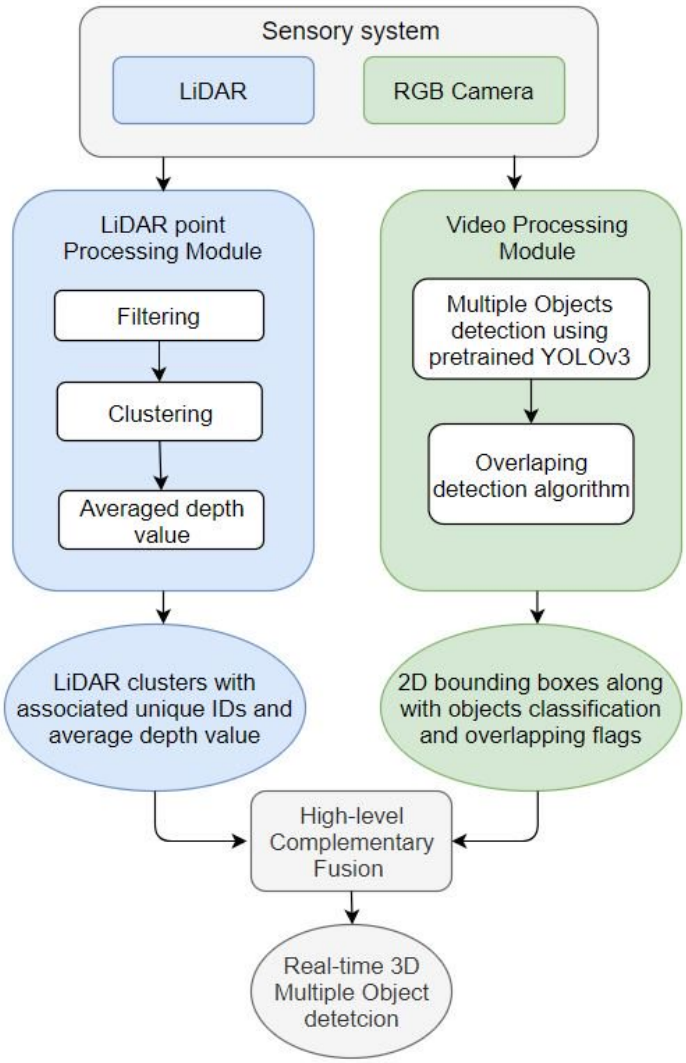

Figure 4.12: Block diagram for the object detection system of multiple overlapping objects using camera and LiDAR

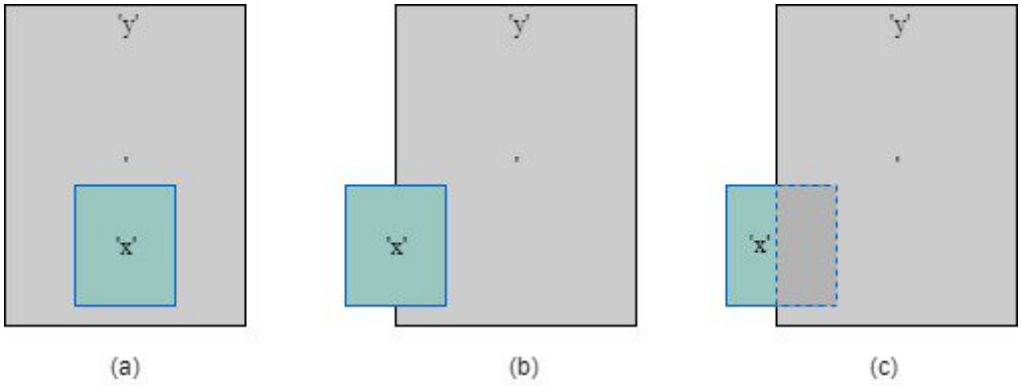

Figure 4.13: Different overlapping scenarios. (a) object 'x' is fully in front of object 'y'; (b) object 'x' is partially in front of object 'y'; (c) object 'x' is partially behind object 'y'

In this procedure, the algorithm examines the LiDAR clusters linked to the bounding boxes of every identified object. The process diagram for this step is illustrated in Figure 4.14. This task will enhance the real-time object detection performed by the video-processing module, as the bounding boxes typically exceed the actual perimeters of the detected objects.

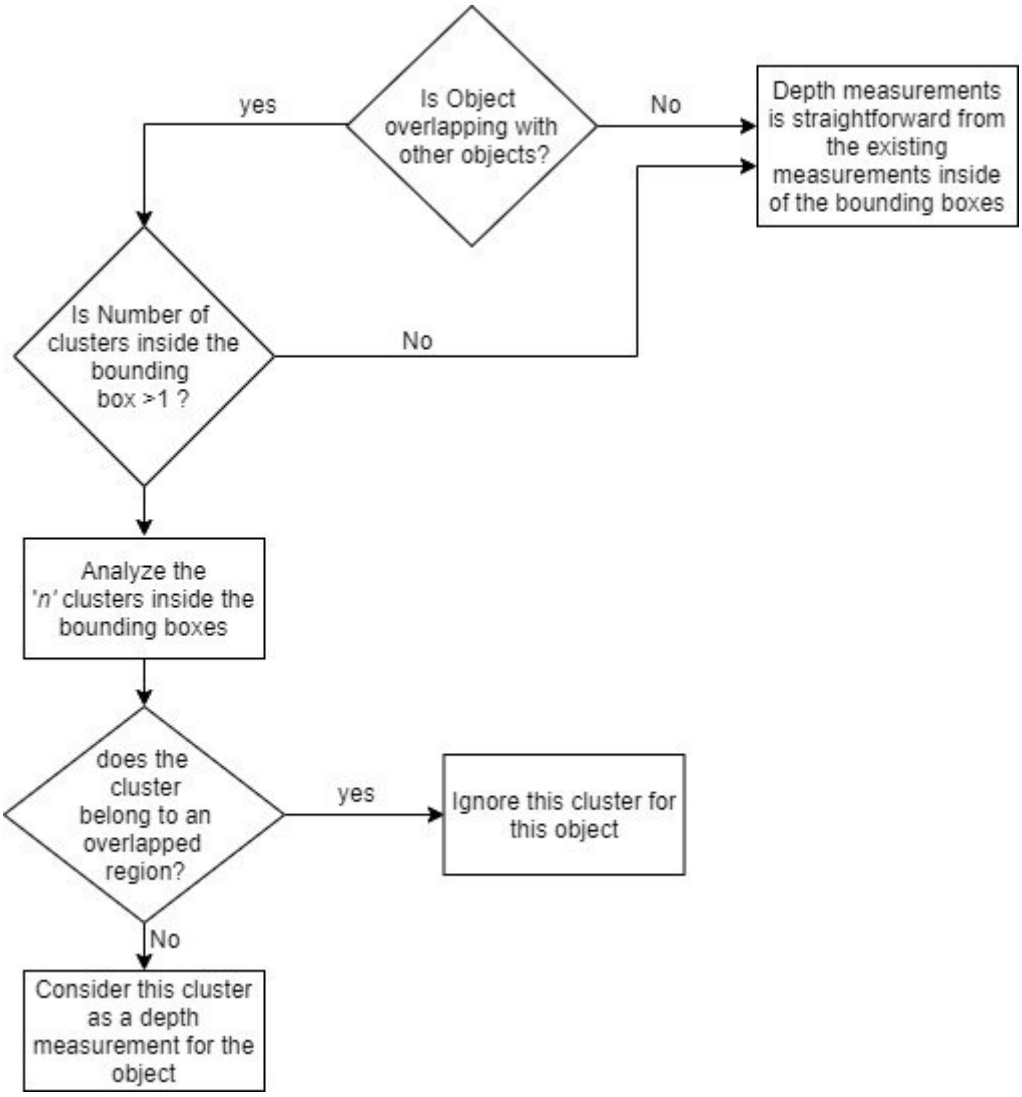

Figure 4.14: Flowchart for calculating the correct depth measurements for detected objects during overlapping objects

4.6 Summary

An affordable solution for achieving the necessary precision in obstacle detection for self-driving vehicles involves a monocular vision-based system. However, this approach only provides a two-dimensional positioning of objects. To overcome this limitation, the incorporation of a range-finder sensor becomes essential. Nevertheless, the adoption of 3D LiDARs presents a challenge due to their high costs, which currently pose an obstacle to the

extensive implementation of autonomous driving across both industrial and research domains.

In this chapter, a low-cost real-time 3D object detection for AVs has been developed using a monocular camera along with a 2D LiDAR. First, the state-of-the-art deep learning object detection method was adapted to detect multiple objects in real driving scenes; as shown in evaluation experiments, YOLOv3 has achieved the best performance (mAP) and running time (as shown in Section 2.4). Moreover, a mapping algorithm in order to convert image pixels to angular rotations was developed to be able to associate both bounding boxes' locations and LiDAR measurements. Secondly, overlapping bounding boxes of detected objects are associated with flags in order to aid with the subsequent process of associating LiDAR measurements.

Afterwards, LiDAR measurements are pre-processed by performing smoothing using a median filter and linearization to be able to divide LiDAR measurements into segments. Each segment is associated with a distance (depth) value, which is the closest depth to the LiDAR. Finally, these segments are mapped to the corresponding bounding boxes while utilizing the overlapping detection algorithm, as shown in Figure 4.14.

The presented study promotes the adoption of cost-effective 2D LiDARs within AVs, thereby enhancing the integration of autonomous driving technology into a broader range of vehicles. One constraint of the suggested methodology is its dependency on the performance of the video-processing module, like YOLOv3. Hence, further exploration is needed to improve this aspect, potentially through the application of de-raining techniques.

This work was published in Sensors Journal, as "Evaluation of 3D Vulnerable Objects' Detection Using a Multi-Sensors System for Autonomous Vehicles" in February 2022.

Chapter 5 Experimental Setup and Results

In this chapter, implementation details and experimental setup are illustrated. Different tasks require different evaluation protocols. Firstly, in order to evaluate the proposed deraining network, different metrics have been used to assess the quality of derained images. Also, both synthetic and real rainy datasets have been used. Secondly, to evaluate the objects detection approach, real road tests were made using the proposed sensory system, including a monocular camera and 2D LiDAR. Manual measurements and assessments validated this system due to the absence of datasets comprising 2D LiDAR measurements and images. Thirdly, joint deraining and detection experiments were performed in order to ensure that the proposed deraining network enhanced the following objects detection operation.

5.1 Evaluation Metrics

5.1.1 Evaluation of Deraining

Quantitative results are obtained after removing rain from rainy images. For synthetic rainy images, full-reference metrics are used as they compare rain-removed images and the original rain-free images. On the other hand, no-reference metrics are used on real rainy images, where the corresponding rain-free images are not present due to the dynamic nature of driving scenarios; no-reference metrics assess the quality of the output image itself.

i. Full-reference metrics

For evaluating different deraining models, two primary assessment metrics are employed based on reference rain-free images: Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM) [216]. Full-reference metrics compare between the de-rained images and their corresponding original rain-free images.

Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) represents the ratio between the maximum power of a signal and the power of the noise affecting the signal. To compute the PSNR between a test image g and a reference image f, both having dimensions M × N, the PSNR is determined using Equation 5.1 [216].

$$PSNR(f,g) = 10\log_{10}(255^2/MSE(f,g)) \quad 5.1$$

Where,

$$MSE(f,g) = \frac{1}{\mathrm{MN}}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f_{ij} - g_{ij}\right)^2 \quad 5.2$$

As the Mean Square Error (MSE) tends towards zero (as shown in Equation 5.2), the PSNR value tends towards infinity. Consequently, a higher PSNR value indicates better image quality, while a lower PSNR value suggests significant numerical differences between images.

Structural Similarity Index Measure

Another commonly used evaluation metric for measuring the similarity between two images is the SSIM (Structural Similarity Index Measure). It was developed by Wang et al. [217] and is closely related to the perceptual quality assessment of the Human Visual System (HVS). The HVS is naturally adapted to extract structural information from images due to the inherent structure of natural image signals. SSIM leverages this structural information present in images to approximate the distortion within an image. Unlike traditional methods that sum up errors, SSIM characterizes image distortion through three distinct factors: loss of correlation, luminance distortion, and contrast distortion. The SSIM is defined in Equation 5.3, where g is a test image, and f is a reference image.

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g) \quad 5.3$$

Where,

$$l(f,g) = \frac{2\mu_f\mu_g + C_1}{\mu_f^2 + \mu_g^2 + C_1} \quad 5.4$$

$$c(f,g) = \frac{2\sigma_f\sigma_g + C_2}{\sigma_f^2 + \sigma_g^2 + C_2} \quad 5.5$$

$$s(f,g) = \frac{\sigma_{fg} + C_3}{\sigma_f \sigma_g + C_3} \quad 5.6$$

- $l(f,g)$ corresponds to the luminance comparison, which measures the closeness of the mean luminance of two images (μ_f and μ_g). This factor is maximal and equal to 1 only if $\mu_f = \mu_g$ (Equation 5.4).
- $c(f,g)$, corresponds to the contrast comparison, which measures the closeness of the contrast of two images. Contrast is measured by the standard deviation σ_f and σ_g. It reaches its maximal and becomes equal to 1 only if $\sigma_f = \sigma_g$, (Equation 5.5).
- $s(f,g)$, corresponds to the structure comparison, which measures the correlation coefficient between the two images f and g. σ_{fg} is the covariance between f and g, (Equation 5.6).
- The positive constants $C_1, C_2, and\ C_3$ are added to avoid a null dominator.

The positive values of the SSIM ranges between '0' and '1'. A '0' value means that there is no correlation between images, on the other hand, '1' means that $f = g$. Therefore, to evaluate the output of the deraining network (while comparing it to ground truth clear images), a bigger SSIM value is indicates better deraining performance.

ii. No-reference metrics

When evaluating deraining algorithms on real-world rainy images, it is infeasible to collect pairs of corresponding clear and rainy images due to the dynamic nature of outdoor scenes. Therefore, no-reference metrics are used to evaluate deraining performance on real-world datasets. Therefore, two no-reference evaluation metrics are used: Naturalness Image Quality Evaluator (NIQE) [218] and Spatial-Spectral Entropy-based Quality (SSEQ) [219]

Naturalness Image Quality Evaluator (NIQE)

NIQE is considered one of the blind Images Quality Assessment (IQA) metrics. It makes use of measurable deviations from statistical regularities existing in images, without training on human-rated distorted images and without any exposure to distorted images [218]. A smaller score indicates better perceptual quality.

The NIQE is applied by computing the 36 identical Natural Scene Statistics (NSS) features from equal-sized patches from the image to analyse their quality, fitting them with the Multivariate Gaussian (MVG) model, and then comparing its MVG fit to the natural MVG

model. The quality of the image is expressed as the distance between an MVG fit of the NSS features extracted from the test image and an MVG model of the quality-aware features extracted from the corpus of natural images [218]. This is shown in Equation 5.7.

$$D(v_1, v_2, \sum_1, \sum_2) = \sqrt{(v_1 - v_2)^T \left(\frac{\sum_1 + \sum_2}{2}\right)^{-1} (v_1 - v_2)} \quad 5.7$$

Where, v_1, v_2, and $\sum_1$, $\sum_2$ are the mean vectors and covariance matrices of the natural MVG model and the rain-removed image's MVG model [218].

Spatial Spectral Entropy-based Quality (SSEQ)

SSEQ was proposed by Liu et al. [219]; it is an efficient general-purpose IQA model. It uses down-sampled responses as inputs, then extracts local entropy feature vector from the inputs (shown in Equation 5.8) and learns to predict image quality scores from these features. Image entropy indicates the amount of information contained within an image.

$$H = -\sum_{i=0}^{255} p_i \, log_2 \, p_i \quad 5.8$$

In Equation 5.8, p_i is the probability associated with the grey level i, obtained from the normalized histogram of an image.

5.1.2 Evaluation of Detection and Classification

The detection techniques produce bounding boxes to delineate the identified regions. In terms of classification, there exist four distinct categories:

- True Positive (TP): Indicates the number of correct predictions of the target class.
- True Negative (TN): Indicates the number of correct predictions of the other class.
- False Positive (FP): Indicates the number of incorrect predictions of the other class classified as the target class.
- False Negative (FN): Indicates the number of incorrect predictions of the target class classified as other class classified class.

Table 5.1 displays the confusion matrix for the above-mentioned classification categories.

Table 5.1: Confusion matrix for classification categories

		True Condition	
		Positive condition	Negative Condition
Predicted Condition	Positive Prediction	TP	FN
	Negative Prediction	FP	TN

These mentioned classification categories are generally adopted in imaging classification methods; Accuracy (Equation 5.9), Sensitivity (i.e., also known as Recall, Equation 5.10), and Precision (Equation 5.11). These measures are using the TP, TN, FP, and FN defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad 5.9$$

$$Sensitivity = \frac{TP}{TP + FN} \quad 5.10$$

$$Precision = \frac{TP}{TP + FP} \quad 5.11$$

Precision, also known as Positive Predictive Values (PPV), signifies the proportion of positive results that are correctly identified as true positives (TP). Additionally, Sensitivity also referred to as Recall, signifies the proportion of actual positive cases that are accurately predicted as positive. Lastly, the accuracy of a classification method reflects its overall performance by calculating the ratio of correct predictions (TP + TN) to all prediction outcomes (TP + TN + FP + FN)

i. Intersection over Union (IoU)

IoU serves as an assessment metric that compares the predicted bounding box (detection output area) with the area annotated in the ground truth. This metric is also known as the Jaccard index and quantifies the overlapping region between predicted and actual areas, as depicted in Figure 5.1. IoU is commonly employed for evaluating object detection techniques. Its mathematical representation is provided by Equation 5.12:

$$IoU = \frac{A_{gt} \cap A_p}{A_{gt} \cap A_p} \quad 5.12$$

Where, A_{gt} represents the area determined by the ground-truth annotations from experts, and A_p corresponds to the bounding box predicted by the detection method.

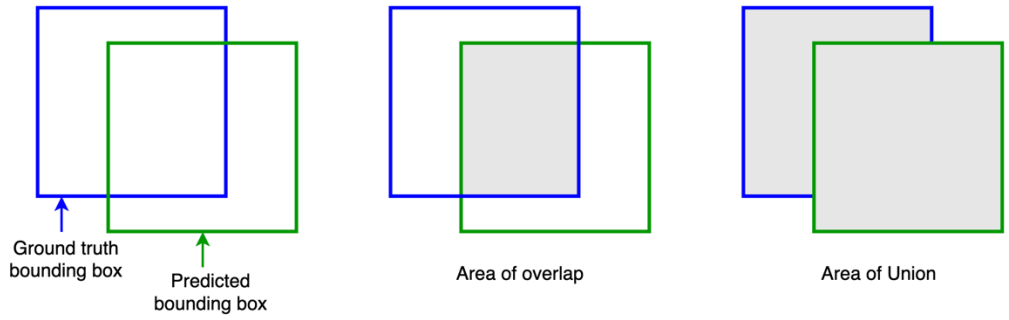

Figure 5.1: IoU calculation

ii. Mean of average Precision (mAP)

The mean of Average Precision (mAP) is employed to assess the detection localization performance of the proposed methods. mAP calculates the mean of Average Precision (AP) from the detection outputs. AP quantifies precision at various recall intervals, as defined in Equation 5.13.

$$AP = \frac{1}{11} \sum_{recall_i} Precision(Recall_i) \quad 5.13$$

5.2 Evaluation Datasets

Various datasets were used within this research to assess and refine the detection and deraining algorithms, enabling a comprehensive evaluation and comparison against existing methods in the field. The forthcoming subsections will elaborate on the specifics of data acquisition, data characteristics, and annotation procedures for each dataset.

5.2.1 Synthetic Deraining Evaluation Datasets

Falling rain is of a dynamic nature; it is uncontrollable and unpredicted. Therefore, it is infeasible to capture pairs of rainy and clear images simultaneously. Hence, rainy images are considered a no-reference category based on Wang et al. [217]. This encouraged the synthesis of datasets for deraining; based on this idea, researchers inject noise in clear images

using synthetic rain generation algorithms. Therefore, knowing where each rain streak is placed makes it easy to obtain a rain streak mask image that can provide a supervisory signal to ML methods.

There are mainly two approaches to synthesizing rainy images. The first approach is to use Adobe Photoshop and apply the non-linear "screen blend mode". The second approach is to superimpose rain and haze on images. These settings are limited in generating diverse types of rain.

The proposed deraining network is evaluated on multiple datasets across image restoration tasks. The datasets used are described below:

i. Test100

The name of this dataset ends with '100' as it consists of 100 images; it was introduced by Zhang et al. [220] due to the lack of availability of large-sized datasets for single-image deraining. They randomly chose 50 images from the last 500 images in the UCID dataset [221] and another 50 random images from the test set of the BSD-500 dataset [222]. After collecting these 100 images, Zhang et al. integrated rain streaks into them using Photoshop following the guidelines in [161]. They ensured adding rain effects of different intensities and orientations to emulate diverse rainy conditions. All images are resized to 256×256. A sample of the dataset is shown in Figure 5.2.

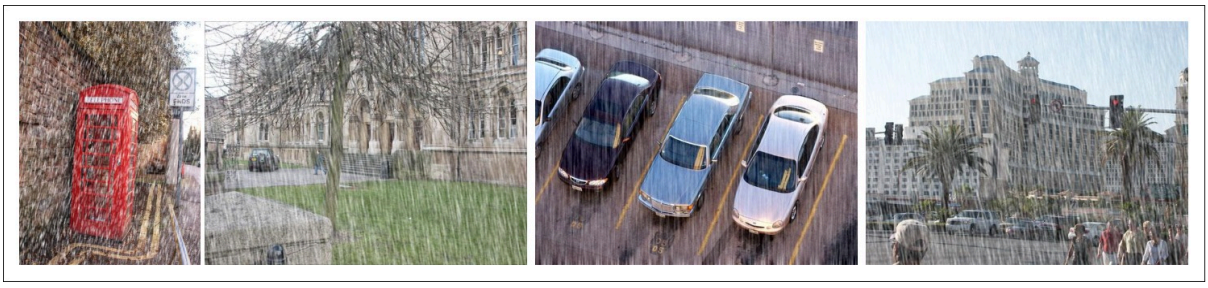

Figure 5.2: Samples of the test100 dataset [220]

ii. Rain100H

This dataset contains 1800 training pairs and another 100 pairs for testing (the reason for the '100' in its name). It was contributed by Yang et al. [139]. It has five heavy streak directions; that is where the letter 'H' at the end of its name comes from. While it is rare for a real rain image to contain rain streaks in many different directions, synthesizing this kind of images for training is observed to boost the capacity of the network. It is composed initially of samples from the BSD200 dataset [223]. A sample is shown in Figure 5.3.

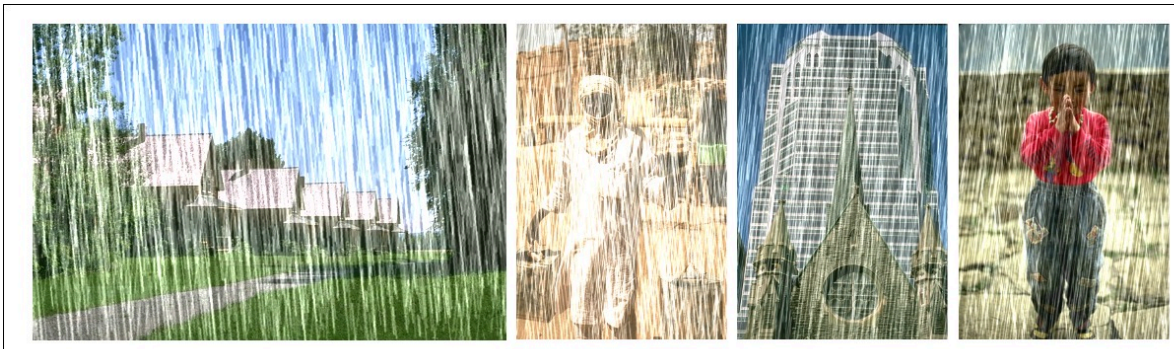

Figure 5.3: Samples of the Rain100H rainy dataset [139]

iii. Rain100L

Similar to the Rain100H dataset, this dataset was provided by Yang et al. [139]. The Rain100L dataset only consists of 200 training pairs and 100 testing pairs (the reason for the '100' in its name). It has synthetic rainy images featuring a single light type of rain streaks; that is where the letter 'L' at the end of its name comes from. It serves as a specialized resource to assess the capacity of deep neural deraining models in grasping the patterns of rain streaks. The images were extracted from BSD200 [223]. A sample is shown in Figure 5.4.

Figure 5.4: Samples of the Rain100L rainy dataset [139]

iv. Test2800

It was created by Fu et al. in [9]. It consists of 14,00 rainy/clean image pairs, a total of 2800 images, and that is the reason why it is called 'Test2800'. Thousand clean images were collected from the UCID dataset [221], BSD dataset [223] and Google image search to synthesize rainy images. Each clean image was used to generate 14 rainy images with different streak orientations and magnitudes. A sample is shown in Figure 5.5.

Figure 5.5: Samples of the Test2800 dataset [9]

v. Test1200

The Test1200 dataset is a large-scale dataset that was contributed to the work of Zhang et al. [185]. It contains 12,000 paired images of rainy scenes with different rain-density levels/labels (i.e., heavy, medium, and light). A sample is shown in Figure 5.7.

vi. NYU Depth Dataset

It was presented by Silberman et al. [224]. It contains 1449 RGBD images recorded by both RGB and depth cameras from Microsoft Kinect, capturing 464 diverse indoor scenes with detailed annotations. It is used in the deraining phase to generate synthetic hazy images to train the rain accumulation removal network. This dataset comprises an extensive collection of indoor images affected by haze, along with their corresponding depth maps. Given that the transmission rate within a scene influences the proportion of light reaching the camera's sensor, this factor is contingent on the distance involved. As a result, the transmission $T(x)$ at a specific location x can be delineated as a function of the depth [225], as illustrated in Equation 5.14.

$$T(x) = e^{-\beta d(x)} \tag{5.14}$$

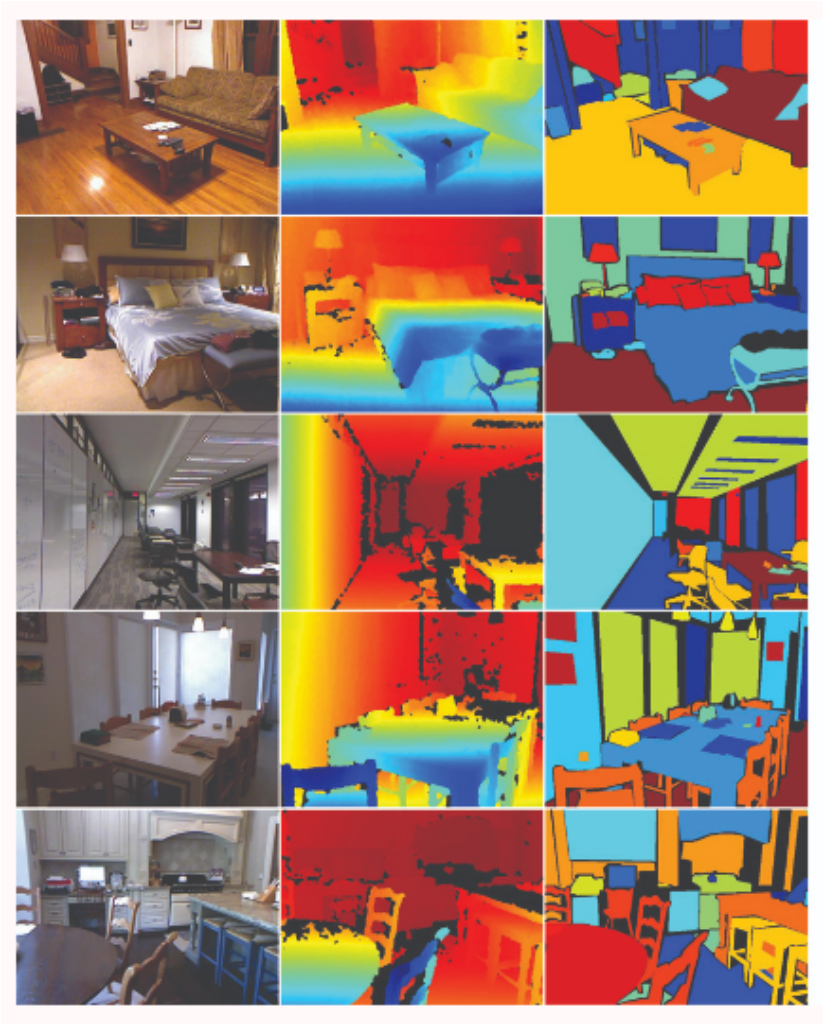

Figure 5.6: Samples of the NYU depth dataset

In Equation 5.14, β denotes the atmosphere's attenuation coefficient, and $d(x)$ represents the scene's depth at location x. A larger β value signifies greater fog density within the scene, with β= 0 indicating an absence of haze in the image. Utilizing clear, haze-free images alongside their corresponding depth data from the NYU dataset, Li et al. [202] generated a total of 16,038 synthetic hazy images. These images were created with varying degrees of fog densities by randomly assigning values for β within the range [0.5,1.5] according to Equation 5.14, and for atmospheric light A within the range [0.7,1.0]. All synthesized images possess a resolution of 480×640 pixels. An example is provided in Figure 5.6; the left column shows the RGB image, the middle column shows the raw depth image, and the right column shows class labels from the dataset [224].

Figure 5.7: Samples of the Test1200 dataset [185]

vii. NTIRE 2018 dataset

This dataset encompasses sets of matched haze-free images alongside their naturally occurring hazy counterparts. It comprises a total of 60 images for training, 10 for validation, and an additional 10 for testing. The collection consists of a diverse range of real-world scenes, both indoors and outdoors, captured under varying degrees of haze presence or absence. To create these corresponding pairs, two professional fog/haze machines were used to generate dense vapour, as documented in previous works [226-228]. Samples from this dataset were used to train the rain accumulation removal network.

5.2.2 Real World Rainy Images Dataset

As acquiring pairs of real driving scenes in different situations (rainy vs. clear) is infeasible, most algorithms depend on synthesized datasets to train and evaluate their algorithms. Although some works have been done to study the physical characteristics of rain, e.g., rain direction [178] and rain density [185], these synthesized datasets still lack the ability to fully model real-world rainy scenes.

viii. Rain in Driving (RID)

Li et al. [173] have collected 2,495 real rainy images from high-resolution driving videos captured by car-mounted cameras in rainy weather in different real traffic scenes during multiple drives. They labelled the bounding boxes for different road objects (e.g., pedestrians, cars, buses, bicycles, motorcycles, etc.) Most images are of 1920×990 resolution, with a few exceptions of 4023×3024 resolution [173]. Samples of the RID dataset are shown in Figure 5.8. Statistics of objects' classes presented in the RID dataset are presented in Table 5.2.

Figure 5.8: Samples of the RID dataset [173]

ix. Rain in Surveillance (RIS)

Li et al. [173] have also collected 2,048 real rainy images from surveillance video cameras with relatively lower resolution (compared to the RID dataset). They were gathered by a total of 154 networked cameras on rainy days; they also ensured the collection of diverse content. The rain effect in the RIS dataset is similar to rain and mist; in addition, the low-resolution cameras cause a foggy effect. Same as the RID dataset, they also annotated the main road objects (e.g., pedestrians, cars, buses, bicycles, motorcycles, etc.). Most of the cameras have a resolution of 640×368, with a few exceptions of 640×480. Samples of the RIS dataset are shown in Figure 5.9. Statistics of objects' classes presented in the RIS dataset are presented in Table 5.2.

Figure 5.9: Samples of the RIS dataset [173]

Table 5.2: Object statistics in RID and RIS datasets [173]

Categories	Car	Person	Bus	Bicycle	Motorcycle
RID Set	7332	1135	613	268	968
RIS Set	11415	2687	488	673	275

5.2.3 Object Detection Evaluation Datasets

In order to train and validate detection networks that will be used to detect objects surrounding AVs outdoors, different datasets with different features have been proposed in this field.

i. KITTI

KITTI was collected via an autonomous driving platform; this dataset encompasses a comprehensive range of tasks, spanning stereo, optical flow, visual odometry, and more. The section dedicated to object detection comprises 7,481 annotated training images and 7,518 testing images. These annotated images present both 2D and 3D bounding boxes of cars, pedestrians, and cyclists within urban driving scenarios.

The nearly equal distribution of images between the training and testing sets serves several aspects:

- Representativeness: it prevents the model from overfitting to specific features of the training set. A balanced distribution helps the model used to generalize unseen data well.
- Evaluation: having a comparable number of images in the testing set allows for a robust assessment of the model's performance.
- Fair assessment: A balanced split ensures that the evaluation is fair and unbiased while providing a more accurate assessment of the model's capabilities across different scenarios

The balance in the number of images is just one aspect of dataset design. The diversity and representativeness of the data, along with careful annotation, are also crucial factors for creating a dataset that effectively supports the development and evaluation of computer vision algorithms, especially in the context of autonomous driving.

The designated 3D metric is the Average Orientation Similarity (AOS), derived by multiplying the Average Precision (AP) of the 2D detector with the mean Cosine distance similarity pertaining to azimuth orientation [229, 230]. Furthermore, the dataset differentiates among three levels of difficulty:

- Easy: Bounding boxes with a minimum height of 40 pixels, encompassing fully visible objects (no occlusion), and truncation limited to a maximum of 15%.
- Moderate: Bounding boxes with a minimum height of 25 pixels, encompassing partially occluded objects, and truncation capped at a maximum of 30%.
- Hard: Bounding boxes with a minimum height of 25 pixels, encompassing objects that are challenging to discern, and truncation restricted to a maximum of 50%.

The KITTI dataset primarily consists of images and sensor data captured in dry weather conditions. The dataset focuses on urban scenarios, and rainy or adverse weather conditions are not explicitly included in the standard KITTI dataset.

Since the test ground-truth labels are not available, the KITTI training set has been equally divided into train and validation sets; also, it has been ensured that the training and validation sets do not come from the same video sequence. Samples from the KITTI dataset are shown in Figure 5.10.

ii. COCO350 and BDD350

Sponsored by Microsoft, the original COCO dataset encompasses over 330,000 fully segmented images, featuring an average of 7 objects per image across a total of 91 categories. The images were sourced from intricate real-life scenarios. Evaluation of outcomes is performed through Average Precision (AP) calculation across various levels of Intersection over Union (IoU) and diverse object sizes [231].

On the other hand, BDD100k is a large-scale dataset comprising diverse scene types, including city streets, residential areas, and highways, and diverse weather conditions at different times of the day [232]. It has 100k videos and 10 tasks to evaluate the progress of image recognition algorithms on autonomous driving. It includes different geographic, environmental, and weather conditions. A sample is shown in Figure 5.11.

In order to conduct joint deraining and object detection assessment, samples curated by Jiang et al. in their work [180] were utilized. Their dataset comprises a collection of samples drawn from diverse driving scenes, incorporating a range of conditions (850 samples originating from COCO and BDD datasets, which led to the creation of novel synthetic rain datasets COCO350 and BDD350). These datasets encompass varying rain orientations, magnitudes, and even encompass intricate scenarios such as night scenes.

Figure 5.10 KITTI Dataset samples [230]

Figure 5.11: Samples of the BDD100K dataset [232]

iii. PASCAL VOC 2007/2012

It consists of 4,824 images comprising 20 categories, including vehicles, animals, bicycles, buses, cars, motorbikes, and people. The images in the PASCAL VOC dataset are diverse and come from various sources, representing a wide range of real-world scenarios; for example, some images were sourced from web data, and others are contributed images. The

images include scenes from both indoor and outdoor environments, with different lighting conditions, backgrounds, and object poses. The dataset is split into three subsets: 1,464 images for training, 1,449 images for validation, and the test set has 1,911 images. Results are evaluated by the Average Precision (AP) in each category and the mean Average Precision (mAP) across all the 20 categories. Additionally, it provides standard labelling and evaluation tools [233, 234].

iv. ImageNet

Structured in accordance with the WordNet hierarchy, this dataset characterizes each meaningful concept through multiple words or phrases. Comprising a collection of 14,197,122 images, ImageNet is arranged into 21,841 subclasses. These subclasses can be viewed as sub-trees stemming from 27 overarching high-level categories [235].

v. SceneFlow

Comprising over 39,000 sets of stereo image pairs, this synthetic dataset exhibits a resolution of 960x540 pixels. The images are classified into three distinct categories: FlyingThings3D, Driving, and Monka. Notably, the dataset offers comprehensive and densely detailed disparity maps as the reference ground truth [236].

vi. ModelNet

Comprising a pristine compilation of 3D Computer-Aided Design (CAD) models, this dataset is sourced from online search engines. The ModelNet dataset manifests in two distinct variations: one containing solely the top 10 object categories and the other encompassing the complete set of 40 classes. In the context of ModelNet40, 12,311 3D mesh models spanning 40 categories are featured, with a division of 9843/2468 for training and testing. Additionally, the dataset offers diverse object alignment and facing-direction configurations. The evaluation metrics employed for ModelNet40 encompass mean per-class Accuracy (mA) and Overall Accuracy (OA) [237, 238].

vii. OutdoorScene

Comprising a total of 200 images, this dataset serves as a testing resource rather than a training one, with a principal focus on evaluating detections of extensively occluded and truncated objects. Within this dataset, there are 659 cars, out of which 235 cars are intentionally occluded and 135 are deliberately truncated [239].

viii. Sydney Urban Objects

Gathered through the utilization of a Velodyne HDL-64E LiDAR system in Sydney, this dataset encompasses 631 distinct scans featuring 26 diverse classes, including vehicles, pedestrians, signs, trees, and more. Its primary objective is to evaluate detection performance under demanding conditions involving varying viewpoints and levels of occlusion [240].

5.3 Experimental Results

The proposed platform is equipped with:

- Sensory system: A monocular camera, 2D LiDAR (Hokuyo UTM-30LX)
- Computational hardware: NVIDIA GeForce GTX 1050Ti GPU, which is regarded as having medium performance, and an Intel Core i7-8750H CPU.

Different experiments have been performed in order to evaluate both the deraining and objects detection approaches separately and then jointly.

1. Deraining assessment:
    a. Evaluation on synthetic rainy datasets (Test100, Rain100H, Rain100L, Test2800, Test1200) using full-reference metrics (PSNR and SSIM) and comparing the performance with other baseline deraining approaches.
    b. Evaluation on real rainy datasets (RID and RIS) using no-reference metrics (NIQE and SSEQ) and comparing the performance with other baseline deraining approaches.
    c. Time performance evaluation and comparing the running time with other baseline deraining approaches.
2. Objects detection
    a. Evaluation of the performance of objects detection from images using the KITTI dataset and YOLOv3 objects detector
    b. Evaluation of the performance of the LiDAR in calculating the depth of surrounding objects
    c. Evaluation of the performance of integrating both the camera and LiDAR in detecting and localizing surrounding objects in real driving scenes
3. Joint deraining and objects detection
    a. Evaluation of the performance of objects detection after performing deraining and comparing the performance with other baseline deraining approaches

5.3.1 Ablation Studies

In this section, different factors affecting the choice of the architecture of the proposed deraining network are investigated, specifically, the number of progressive stages employed in the rain streaks removal stage and also the effect of applying progressive operations.

i. Number of progressive stages in the rain streak removal phase

The diagrams presented in Figure 5.12 illustrate the impact of varying the number of PRN stages (T) on the evaluation metrics of PSNR and SSIM. These metrics are employed to assess the deraining performance concerning image quality on the Rain100H dataset. It can be observed that an increase in the number of stages within the network corresponds to higher values for both PSNR and SSIM. However, it can be noted that beyond $T > 8$, there is no further enhancement in both PSNR and SSIM.

Furthermore, as shown in Figure 5.13, the average execution times during testing on the Rain100H dataset for models with stages T = 2, 3, 4, 5, 6, 7, 8, 9, and 10 are provided. Balancing the trade-off between efficiency and deraining performance, a decision was made to set the value of T to 8 for subsequent experiments.

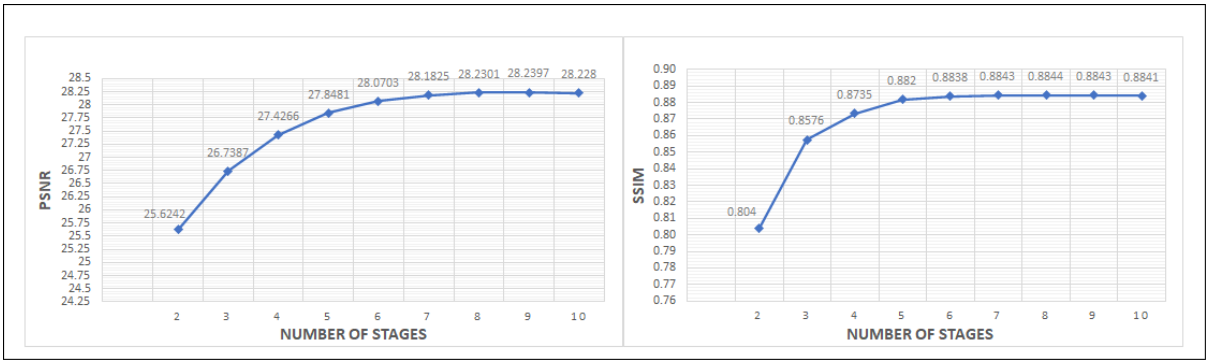

Figure 5.12: Effect of increasing number of progressive stages in rain streak removal network on PSNR and SSIM evaluation metrics.

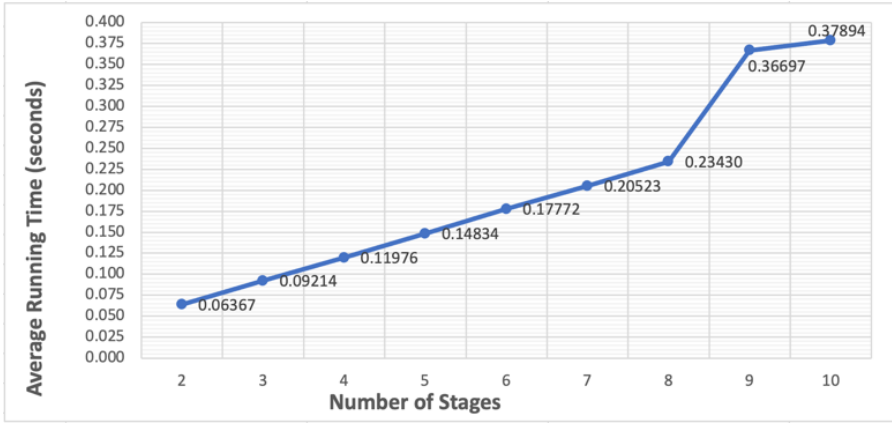

Figure 5.13: Effect of increasing number of progressive stages in rain streak removal network on running time (in seconds).

i. Effect of testing different stages progressively

Two experiments were conducted to determine the optimal arrangement of the network's components. Initially, consideration was given to implementing rain accumulation removal as an initial pre-processing step. However, as demonstrated in Figure 5.14, this approach adversely impacted the overall deraining process. This is attributed to the fact that the rain accumulation removal phase led to undesirable alterations in the quality of rain streaks, resulting in an increase in their contrast and visibility across both real-world and synthetic datasets. Subsequently, the second proposed approach involved incorporating rain accumulation removal progressively, in conjunction with rain streak removal, as depicted in Figure 5.15. Additionally, the corresponding PSNR and SSIM values for both suggestions are outlined in Table 5.3.

Table 5.3: Effect of applying rain accumulation removal as a pre-processing vs. progressive operation

Approach	**PSNR**	**SSIM**
Rain accumulation removal as pre-processing (Figure 5.14)	15.683	0.683
Rain accumulation removal performed progressively with rain streak removal (Figure 5.15)	33.01	0.901

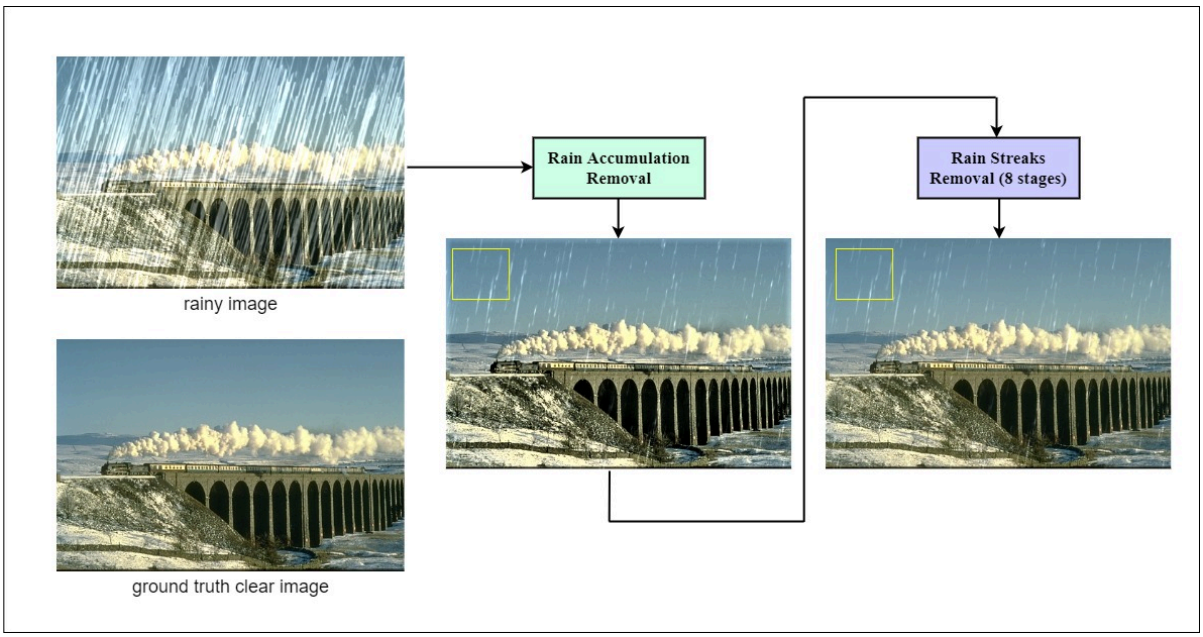

Figure 5.14: The effect of applying rain accumulation removal as a pre-processing step, and then rain streak removal

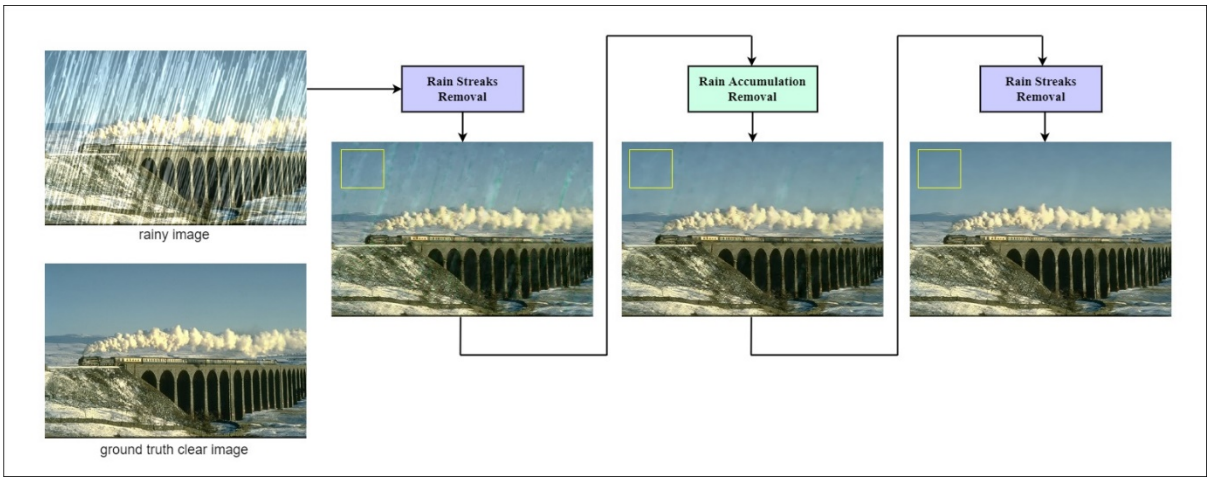

Figure 5.15: The effect of performing rain streak removal and rain accumulation removal progressively

5.3.2 Deraining Experimental Results

This section elaborates the settings on which the proposed network has been trained and tested. Also, comprehensive experimental results are presented and discussed in terms of quantitative and qualitative evaluations. Four main experiments were conducted in this section.

In the first experiment, five synthesized rainy datasets were used to evaluate the performance of the proposed network compared to other deraining networks (HINet [190], MPRNet [189], PreNet [141], PRN [141], and MSPFN [180]). The used synthesized datasets are Rain100H [178], Rain100L [178], Test100 [220], Test1200 [185], and Test2800 [9].

The second experiment was conducted to evaluate the performance of deraining networks on real datasets, RID and RIS [173]. However, due to the lack of presence of rainy/clear images as a reference, different evaluation metrics were used, namely, NIQE and SSEQ.

The third experiment was intended to ensure a fast performance of the deraining process.

i. Implementation Details

The training of both rain streak removal and rain accumulation removal was carried out independently; however, their testing was performed progressively. The training set for the rain streak removal network comprised 1800 images from the Rain100H dataset. On the other hand, training the rain accumulation removal network involved a synthetic dataset of hazy images derived from the NYU dataset [224], along with augmented real hazy data sourced from the NTIRE 2018 dataset. For training purposes, both networks employed the ADAM algorithm optimizer with an initial learning rate of 0.001.

ii. Evaluation on synthetic datasets

The performance of the proposed network is assessed across five distinct synthetic datasets, namely, Test100 [220], Rain100H [178], Rain100L [178], Test2800 [9], and Test1200 [185]. The comparative PSNR and SSIM values are organized and presented in Table 5.4. Evidently, the proposed model achieves a noteworthy enhancement over state-of-the-art (SOTA) deraining algorithms.

The qualitative outcomes, presented in Figure 5.16, highlight the results obtained on various datasets, as elaborated in Table 5.4. The proposed model notably excels in restoration performance, particularly evident in datasets such as Test100, Rain100H, and Test1200, which encompass a wide range of rainy conditions including both rain streaks and rain streak accumulations. Furthermore, the proposed network demonstrates enhanced performance levels on Rain100L and Test2800 datasets. In comparison, other networks tend to either overly smooth images, leading to blurriness, or leave certain rain streaks visible within the images.

Table 5.4: Average PSNR and SSIM comparison on the synthetic datasets Test100, Rain100H, Rain100L Test2800, and Test1200. Red, blue, and green colours are used to indicate 1st, 2nd, and 3rd rank, respectively.

Method	Test100		Rain100H		Rain100L		Test2800		Test1200		Average	
	PSNR	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
HINET	30.29	0.906	30.65	0.894	37.28	0.970	33.91	0.941	33.05	0.919	33.036	0.926
MPRNET	30.27	0.897	30.41	0.890	36.40	0.965	33.64	0.938	32.91	0.916	32.726	0.9212
MSPFN	27.50	0.876	28.66	0.860	32.40	0.933	32.82	0.930	32.39	0.916	30.754	0.903
PreNet	24.81	0.851	26.77	0.858	32.44	0.950	31.75	0.916	31.36	0.911	29.426	0.8972
PRN	23.512	0.761	28.07	0.884	36.98	0.9772	23.7882	0.8121	19.733	0.7095	26.417	0.829
proposed	32.43	0.93	33.01	0.901	37.21	0.969	33.89	0.932	34.23	0.953	**34.154**	**0.937**

Figure 5.16: Qualitative results on synthesized datasets

iii. Evaluation on Real Rain Datasets

Further assessments were undertaken on real-world rainy datasets, specifically RID and RIS [173], to comprehensively evaluate the generalization capabilities of the proposed model. Given the unavailability of ground truth images devoid of rain, the evaluation of deraining performance necessitates the application of no-reference metrics such as NIQE and SSEQ. The quantified outcomes are provided in Table 5.5 to offer quantitative insights into the achieved results.

Table 5.5: Comparison results of average NIQE/ SSEQ on real-world datasets (RID, and RIS). The smaller scores indicate better perceptual quality.

Dataset	Evaluation Metrics	HINet	MPRNet	MSPFN	PreNet	Proposed
RID	NIQE ↓	4.985	4.856	6.518	7.007	4.562
	SSEQ ↓	81.1619	50.648	40.47	43.04	48.213
RIS	NIQE ↓	6.887	6.045	6.135	6.722	6.148
	SSEQ ↓	52.983	51.689	43.47	48.22	52.681

iv. Time Performance Evaluation

Given the tight real-time demands of tasks in AVs, particularly concerning object detection, it is essential that any supplementary processing does not introduce substantial computational intricacy that could undermine real-time functionality. Therefore, the deraining process should take this into consideration. Moreover, the dynamic and mobile state of both AVs and surrounding objects presents another challenge. Given that computational efficiency stands as a critical consideration for autonomous driving applications, demanding real-time operation, a balance between processing demands and performance is crucial. Table 5.6 provides a breakdown of the execution times for different methods. This evaluation offers insights into the computational efficiency of each method, a vital criterion for applications in the AV domain. The proposed model achieves competitive performance compared with other models with inference time of 0.44× HINet, 0.31× MPRNet, 0.6× MSPFN, and 0.54× PreNet, on image size 500×500, and 0.41× HINet, 0.3× MSPFN, and 0.53× PreNet, on images of size 1024×1024. This speedup is due to the employment of progressive lightweight stages with fewer parameters compared to other baseline networks. The proposed network has 98.7K parameters; on the other hand, MSPFN has 13.22M parameters, while MPRNet has 3.57M parameters.

Table 5.6: Comparison of running time (seconds) on NVIDIA GTX 1050Ti GPU

Image size	HINet	MPRNet	MSPFN	PreNet	proposed
500x500	0.574	0.795	0.415	0.464	**0.251**
1024x1024	2.0655	2.782	1.5376	1.578	**0.843**

Authors of these deraining algorithms have evaluated the inference time in their research on different GPUs, therefore, their running times had to be re-evaluated on the same image sizes on a unified computer to produced valid comparisons. As stated in their published research:

- HINet: Tesla V100
    - Rain13K, 256×256, 0.027(s)
- MPRNet: Tesla V100/ NVIDIA Titan Xp
    - Rain13K, 256×256, 0.0374(s)
- MSPFN: NVIDIA Titan Xp
    - Test100, 512×384, 0.308(s)
    - COCO350, 60×480, 0.58(s)
    - BDD350, 1280× 720, 1.24(s)
- PreNet: NVIDIA GTX 1080Ti
    - 500× 500, 0.088(s)
    - 1024×2024, 0.551(s)
- PRN (6 stages): NVIDIA GTX 1080Ti
    - 500× 500, 0.088(s)
    - 1024× 1024, 0.296(s)

5.3.3 Objects Detection Experimental Results

Complementary fusion has been adopted to integrate both data captured by camera and LiDAR in order to achieve 3D object detection in real time. First, 2D object detection, classification, and horizontal and vertical localization have been performed using YOLOv3. Also, an algorithm has been developed in order to detect overlapping objects and flag them. Second, 2D LiDAR measurements are pre-processed and complemented the bounding boxes that are the output of the object detection framework in order to associate a depth measurement to detected objects in the bounding boxes.

i. Experimental setup

For real-time evaluation experiments, the monocular camera, the 2D LiDAR, and the computational hardware have been mounted such that they have a common vertical axis. The LiDAR has been placed ~559mm up from the ground in order to be above the vehicles' ground clearance. In order to perform tests in real driving scenarios, the sensory system has been placed on a trolley in order to navigate with it easily in the streets.

ii. Objects detection from images

YOLOv3 has been selected as the real-time object detector [229, 230]. The results obtained by YOLOv3 when tested on the raw KITTI and PASCAL VOC datasets are displayed in Table 5.7. Moreover, YOLOv3 achieved AP of 93.41%, 71.84%, and 83.6% for car, person, and cyclist detection, respectively. There are object detectors that perform better on object detection datasets (for example, Faster R-CNN and SSD). Still, they cannot be employed in real-time autonomous driving scenarios because of their slow execution speed. Additional comparisons between YOLOv3 and other deep learning object detection algorithms on various datasets are offered in [206].

Table 5.7: Mean average precision (mAP) of testing YOLOv3 on the KITTI and PASCAL VOC datasets

Detection Algorithm	mAP
KITTI	82.95%
PASCAL VOC	79.26%

iii. LiDAR measurements processing

Due to the diverse shapes of surrounding objects, non-linearity and different reflectivity, measurements captured by a LiDAR are not straight lines; on the contrary, they have many outliers. In order to smooth the readings while preserving the edges, median filtering is used as the initial stage in the processing of LiDAR measurements. Before filtering, a sample of LiDAR measurements is shown in Figure 5.17(a), and the filtered readings are shown in Figure 5.17(b).

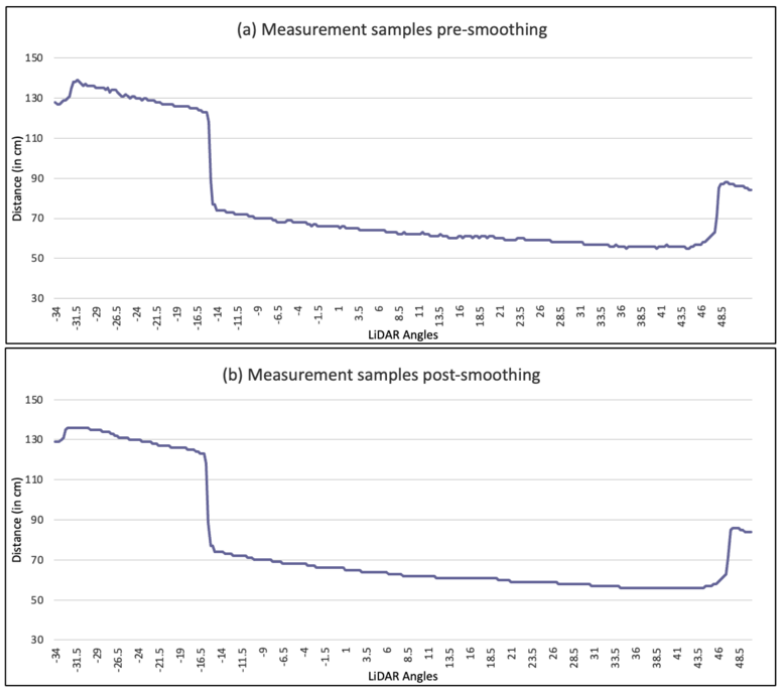

Figure 5.17: A sample of LiDAR measurements sample (a) pre-smoothing (b) post-smoothing

The next stage includes segmenting the LiDAR readings and giving each fragment a distinct ID and depth value. An example of LiDAR measurements taken when two cars were present is shown in Figure 5.18 (one car is partly in front of the other).

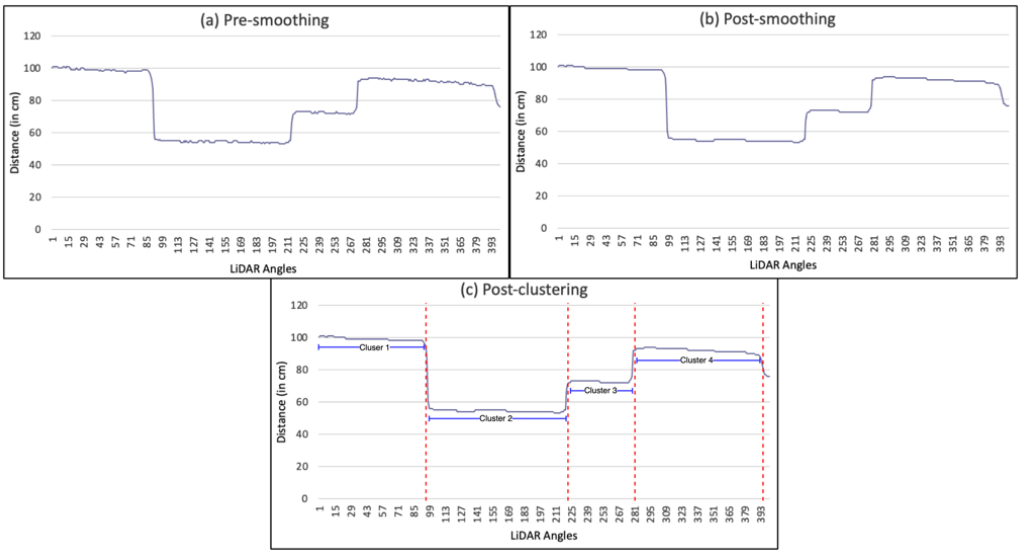

Figure 5.18: A sample of LiDAR measurements when scanning two overlapping objects

LiDAR measurements show different clusters along the horizontal plane, each associated with an average depth measurement. However, these measurements are not enough to indicate the class of the detected objects. Therefore, complementary sensory fusion with data

of rich features is mandatory in order to be able to complement the depth measurements with objects' classes.

iv. Three-dimensional measurements after sensors fusion

Experiments were done in actual driving situations, and the performance was manually assessed and validated because datasets with 2D LiDAR point clouds were unavailable. Manual markings were made on the testing ground with manually measured depth distances from the sensory system; these measurements are considered reference measurements in order to be compared to the distances measured by the proposed sensory system. Experiments were done in different surrounding scenarios:

- *Scenario 1*: Static cars
- *Scenario 2*: Static pedestrians
- *Scenario 3*: Four moving cars
- *Scenario 4*: Four moving pedestrians

Tests were made such that they included different aspects, for example, different objects' orientations, speed, and lighting. Each of the scenarios that included moving objects was tested over a time frame of 2 minutes. Table 5.8 shows samples of the measurements that were captured at different times and compared to the actual locations, which were manually marked per each scenario.

Table 5.8 Evaluation results of LiDAR measurements in different scenarios

Scenario	Sample	Measured distance (obj1, obj2)	Actual distance (obj1, obj2)	Error (obj1, obj2)	Average Error
Scenario 1	1	4.014m, 7.465m	4m, 7.5m (overlapping)	0.014m, -0.035m	0.04475m
	2	3.985m, 7.115m	4m, 7m	-0.015m, 0.115m	
Scenario 2	1	0.986m, 2.873m	1m, 3m	-0.014m, -0.127m	0.036075m
	2	1.998m, 2.0987m	2m, 2.1m (overlapping)	-0.002m, -0.0013	
Scenario 3	1	4.567m, 6.984m	4.5m,7m	0.067m, -0.016m	0.052625m
	2	5.909.m, 9.997m	6m, 10m (overlapping)	0.091m, -0.003m	
	3	7.532m, 11.985m	7.5m, 12m	0.032m, -0.015m	
	4	9.874m, 13.429m	10m, 13.5m (overlapping)	-0.126m, -0.071m	
Scenario 4	1	1.476m, 3.984m	1.5m, 4m	-0.024m, -0.016m	0.035625m
	2	1.511m, 1.879m	1.5m, 1.9m (overlapping)	0.011m, -0.021m	
	3	2.050m, 2.100m	2m, 2m	0.050m, 0.100m	
	4	5.521m, 2.042m	5.5m, 2m (overlapping)	0.021m, 0.042m	

The final stage involves using the LiDAR data to complement the 2D bounding boxes produced by the real-time visual object detector (YOLOv3) with a third dimension (depth) after filtering and grouping. The system was evaluated in real-time scenarios, and it was able to meet the real-time requirements by operating at 18 FPS on an average GPU, retaining dynamic object recognition, and enhancing the bounding boxes with a depth dimension. The model's speed could be improved by utilizing a more potent device. The attained execution time of the suggested system marks significant progress when contrasted with alternative methods (as shown in Table 2.8). Testing of the system took place amidst dynamic driving scenarios involving mobile vehicles and pedestrians, where objects exhibited overlap and mutual interaction.

During the performed experiments, YOLOv3 did not missed detecting any objects. In the performed experiments, manual validation was performed, both for bounding boxes and depth measurements. The surrounding scenarios in the test cases are considered to have lower difficulty levels compared to other more complicated driving scenarios present in detection-dedicated datasets.

Nonetheless, the system's constraint pertained to weather conditions since the video-processing component lacks the necessary resilience to effectively handle inclement weather situations like rain, snow, and fog. Therefore, this problem has been addressed by proposing a lightweight deraining network (Chapter 3). The following subsection will demonstrate the experimentation results when performing joint deraining and objects detection.

5.3.4 Joint Deraining and Detection Experimental Results

Diverse weather conditions, particularly rain, significantly deteriorate the accuracy of object detection. Raindrops obscure and distort the underlying visual attributes of nearby objects, attributes that detection methods rely on for object classification and localization. Figure 5.19 demonstrates the performance of object detection with and without deraining in advance.

Not all deraining algorithms contribute positively to the enhancement of object detection performance. As demonstrated by Hnewa and Radha in 2020 [5] and Li et al. in [173], certain deraining methods (for example: [9, 10]) might, in fact, deteriorate detection performance when compared to utilizing rainy images directly as input within corresponding detection frameworks. This degradation can be attributed to the fact that these tested deraining algorithms tend to introduce excessive smoothing into images. Consequently, important visual features of a scene are distorted, and the edges of objects become smoother, resulting in objects being missed during detection.

Hence, it is crucial to assess the effectiveness of the proposed deraining model by concurrently evaluating its impact on object detection performance on the de-rained results. An experiment was performed to apply object detection on de-rained images and evaluate the object detection performance. The influence that is imposed by the deraining models on the performance of object detection performed by AVs is illustrated.

To conduct this assessment, samples curated by Jiang et al. in their work [180] were utilized. Their dataset comprises a collection of samples drawn from diverse driving scenes,

incorporating a range of conditions (850 samples originating from COCO and BDD datasets, which led to the creation of novel synthetic rain datasets COCO350 and BDD350). These datasets encompass varying rain orientations and magnitudes, and even encompass intricate scenarios such as night scenes.

The evaluation process commenced by applying the proposed deraining algorithm alongside other baseline deraining techniques [141, 180, 189, 190] to restore the rain-free counterparts of these images. Subsequently, the YOLOv3 object-detection framework was employed to gauge the detection efficiency within the context of the derained images. The quantified outcomes, encompassing both the performance of the deraining process and the efficiency of object detection, are tabulated in Table 5.9. To provide a visual context, comparative examples are showcased in Figure 5.19; the first row denotes samples of the deraining results on the COCO350 dataset, and the second row denotes samples of the deraining results on the BDD350 dataset. The third and fourth rows show the results of object detection performed on the first and second rows sequentially. YOLOv3 has been used for object detection.

Table 5.9: Comparison results of joint image deraining, and object detection on samples of COCO and BDD datasets.

Methods	**Rainy Input**	**HINET**	**MPRNET**	**MSPFN**	**PreNet**	**Proposed**
Deraining; Dataset COCO350/BDD350						
PSNR ↑	14.79/14.13	18.0680/15.61	18.01/16.65	18.23/17.85	17.53/16.9	18.879/17.98
SSIM ↑	0.648/0.470	0.7859/0.5675	0.7864/0.7792	0.782/0.761	0.765/0.652	0.801/0.823
Object Detection; Algorithm: YOLOv3; Dataset: COCO350/BDD350; Threshold:0.6						
Precision (%) ↑	23.03/36.86	32.87/41.52	31.45/40.05	32.56/41.04	31.31/38.66	33.54/41.97
Recall (%) ↑	29.6/42.8	39.54/50.21	38.21/49.32	39.31/50.40	37.92/48.59	40.12/50.67
IoU (%) ↑	55.5/59.85	60.57/62.53	62.54/62.21	61.69/62.42	60.75/61.08	62.34/61.23

As revealed in Table 5.9, the precision of the produced de-rained images by the proposed model exhibits a substantial enhancement of 68% compared to the original rainy inputs. Furthermore, when contrasted with other state-of-the-art deraining models, the derained images generated through the proposed approach exhibit more coherent and distinct features which efficiently boosts the detection performance.

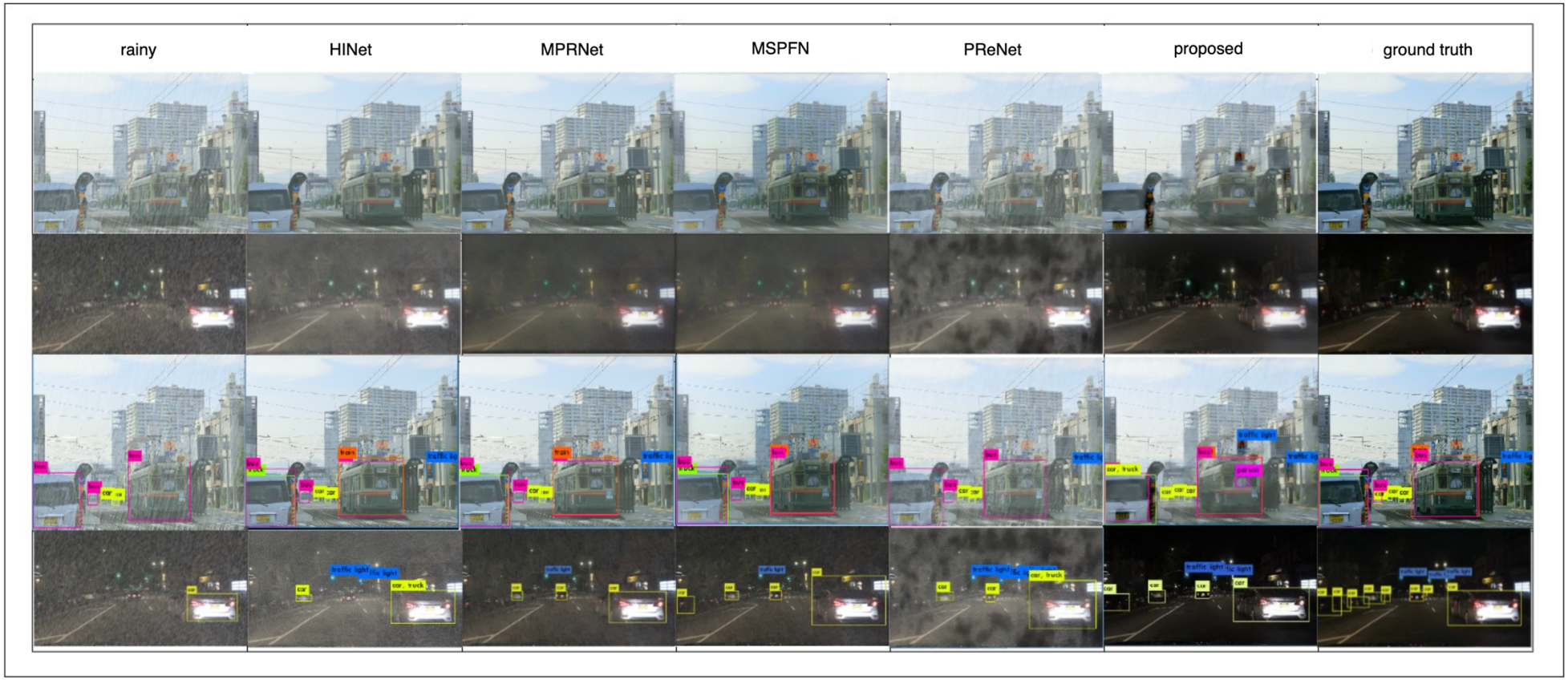

Figure 5.19: Examples of joint deraining and object detection

5.4 Summary

In this chapter, an exhaustive examination of the developed approaches is presented. The chapter encompasses a comprehensive discussion of the utilized evaluation metrics that are used to measure the effectiveness and efficiency of the proposed approaches. For the deraining process, full-reference metrics (*PSNR* and *SSIM*) are used to evaluate the proposed deraining network over synthetic datasets. In contrast, no-reference metrics (*NIQE* and *SSEQ*) are used when evaluating over real rainy datasets, as these datasets lack corresponding clear pairs. For the assessment of the object detection framework, the standard evaluation metrics are adopted: *Precision*, *Recall*, *IoU* and *mAP*.

Moreover, a thorough exploration of the used datasets is provided. Diverse datasets are used to guarantee the performance of the deraining network in different scenarios and include variable raining degradation effects. Five synthetic datasets are used (*Test100*, *Rain100H*, *Rain100L*, *Test2800*, *Test1200*), while two real rainy datasets are also used (*RIS* and *RID*). The diversity of the used datasets shows the effect of the proposed deraining network on different rain degradation effects compared to other baseline deraining approaches. On the other hand, different datasets are also explored, which are used for objects detection evaluation. Additionally, a sample of *COCO* and *BDD* datasets were modified with synthetic rain in order to perform joint deraining and objects detection using the proposed approaches.

Multiple experiments have been performed to assess different aspects of the proposed approaches, and results have been compared to existing baseline approaches. The proposed deraining network showed enhanced results especially on datasets having heavy and diverse rain degradations. Moreover, it is 2.23× faster than the average running speed of baseline deraining networks. Also, the proposed sensory system achieved reliable classification and localization of surrounding objects in real driving scenes. Moreover, both deraining and objects detection are jointly tested, and it was shown that performing deraining ahead of objects detection caused 1.45× enhancement in the objects detection precision.

Chapter 6 Conclusions and Future Works

6.1 Conclusion

Autonomous driving technologies are being researched and developed as next-generation transportation due to their multiple benefits such as reducing the number of traffic accidents and offering passengers extra free time during their journeys. While the deployment of full AVs (Level 5) is still expected in the next few years, the current available AVs are usually classified between SAE autonomy Level 2 and Level 3. The majority of commercially available AVs and research into them mainly depend on employing expensive sensors. However, this impedes the development of further research on the operations of AVs. Moreover, another important challenge in introducing AVs into the market is the ability to operate under various driving conditions by implementing a robust perception system that considers adverse weather conditions i.e., rain.

This thesis explores objects detection and 3D localization of surrounding objects in driving scenes while focusing on maintaining a reliable detection performance in rainy weather conditions. One of the key observations which can be drawn from the reported work is that deep learning has a profound impact on object detection and autonomous driving, bringing significant advancements and improvements to these fields. Deep learning models have enabled real-time object detection in diverse and complex environments. This capability is crucial for autonomous vehicles, where quick and accurate decision-making is essential for safety.

AVs operate under dynamic and diverse circumstances surrounded by multiple dynamic objects. Hence, AV should be robust enough to detect all dynamic surrounding objects in real time. Four main challenges that were addressed in this thesis are:

- High sensory systems' cost
- The complex interaction between objects where occlusion and truncation occur
- The dynamic changes in the perspective and scale of bounding boxes
- The effect adverse weather condition imposes on perceived images and their effect on the object detection task.

6.1.1 Data Deraining

Rain can be characterized by the presence of numerous drops exhibiting diverse sizes, intricate forms, and varying velocities. Rain contributes to two distinct types of visibility reduction. Rain streaks tend to create specular highlights, obstruct, and distort background scene elements. On the other hand, the accumulation of distant rain streaks produces atmospheric veiling effects, scattering light and obscuring the line of sight, similar to the effect of fog. Because the characteristics of rain streaks and the accumulation of rain are distinct, the distortions they introduce to images differ. Therefore, each type of rain degradation is addressed as a separate problem.

Numerous approaches have been proposed to address rain detection and removal. This challenge can be approached either from a video-based perspective or as an image-based one. However, a notable proportion of these approaches failed to consider the specific requirements of AVs, exhibiting the following shortcomings: high computation time, solely focusing on the problem of rain detection, assuming static scenarios, inability to perform under real-time constraints, and not considering the effect on objects detection performance.

As noted from existing deraining approaches, not all deraining algorithms contribute positively to the enhancement of object detection performance. Certain deraining methods might, in fact, deteriorate detection performance when compared to utilizing rainy images directly as input within corresponding detection frameworks.

6.1.2 Sensory Systems

Within the intricacies of autonomous driving, dynamic road elements such as pedestrians, cyclists, and different vehicles impose a significant challenge due to their unpredictable behaviour. In this context, the operational dependability of object detection, a critical aspect for AVs, is especially put to the test. The prevailing sensory systems employed in both commercial AVs and research initiatives rely on the use of expensive sensors which acquire huge amounts of data. This adds a requirement of utilizing extensive computational hardware and still adds to the running time of the whole system, which should be minimized in order to allow for additional operations applied by the system.

6.2 Contribution

To summarize, the contribution of this project includes developing a novel lightweight deraining network that performs single-image deraining and mitigates the degradations

caused by rain streaks and rain streaks accumulation. Moreover, a real-time objects detection platform has been proposed by applying sensor fusion between a monocular camera and a 2D LiDAR. A novel algorithm has been implemented in order to differentiate between interacting surrounding objects and be able to associate separate objects with specific 3D locations.

6.2.1 Data Deraining

The main contributions towards developing a deraining network are:

- A new baseline light-weight network has been proposed, which restores images distorted by different rain degradation effects.
- The network utilizes two subnetworks: the first is concerned with rain streak removal based on Progressive Residual Networks (PRNs), and the second is concerned with rain accumulation removal (similar to defogging).
- The proposed network was evaluated on synthetic datasets (Test100, Rain100H, Rain100L, Test1200, and Test2800) in addition to real-world rainy datasets (RID and RIS).
- The output of the proposed network was evaluated jointly with the object detection algorithm (YOLOv3) in order to show the difference in object detection performance on rainy and de-rained images.
- Baseline deraining algorithms have been implemented and compared with the proposed model by using different metrics. For example, PSNR and SSIM, when comparing their performance on synthetic datasets, NIQE and SSEQ on real-world rainy datasets. Moreover, they were all evaluated on a computer equipped with the same GPU (NVIDIA GTX 1050Ti) in order to measure their running times, as real-time performance is a crucial constraint for AVs performance.

The proposed model shows significant enhancement in the deraining performance, notably evident in datasets like Test100, Rain100H, and Test1200. These datasets cover diverse raining conditions, including both rain streaks and accumulations. Additionally, the proposed network exhibits improved performance on Rain100L and Test2800 datasets. In contrast to other networks, which either excessively smoothed the images resulting in blurriness or allow certain rain streaks to remain visible in the images.

Moreover, the proposed model achieves competitive running time compared with other models with inference time of 0.44× HINet, 0.31× MPRNet, 0.6× MSPFN, and 0.54× PreNet, on image size 500×500, and 0.41× HINet, 0.3× MSPFN, and 0.53× PreNet, on images of size 1024×1024. One of the reasons for this speedup is due to the employment of progressive lightweight stages with fewer parameters compared to other baseline networks.

6.2.2 3D Objects Detection

A low-cost single-beam LiDAR and a monocular camera have been used to achieve multiple 3D dynamic object detection in real driving scenarios, all the while maintaining real-time performance. This research acts as a foundation for the employment of 2D LiDARs on AVs as a lower-cost substitute for 3D LiDARs. Also, the challenge of the presence of multiple overlapping objects in the same scene was addressed. The main contribution of this work lies in several aspects:

- A fusion algorithm between LiDAR measurements and the detected objects in visual images captured by the monocular camera. Noting that, LiDAR works by measuring the distances in an angular rotational pattern; hence, the measurements acquired are radial. Therefore, in order to normalize the LiDAR measurements, these radial measurements must be converted into perpendicular measurements in order to map between Image and LiDAR coordinates.
- Clustering/grouping of LiDAR measurements and calculating an average depth value for each cluster.
- Detection of overlapping objects in images captured by the camera.
- Maintaining a real-time performance.
- Due to the lack of 2D LiDAR point clouds, all the testing was performed in real-time driving scenarios, and the performance was manually measured and validated.

The mAP obtained by YOLOv3 when tested on the raw KITTI and PASCAL VOC datasets are 82.95% and 79.26%, respectively. Furthermore, examinations of LiDAR measurements indicated an average error of ±0.04 meters in real driving scenarios. Additionally, in joint tests involving both deraining and object detection, it was demonstrated that conducting deraining prior to object detection led to a 1.45 times enhancement in object detection precision.

6.3 Future Work

The proposed methods for object detection of multiple dynamic road objects for AVs in challenging weather conditions have been evaluated on different available datasets. The evaluation results showed outstanding performance compared to the state-of-the-art results. The potential future directions in technical aspects are summarized below.

- **Mitigating other types of weather conditions**

In the current work, the model focused on rain streak and accumulation removal as it is the most common weather condition happening in the UK, with an annual average rainfall percentage of around 33.7%. Further research direction includes the investigation of enhancing the model to mitigate more diverse weather conditions such as snow, sun glare, and lightning. Moreover, the proposed network performs removal of both rain streaks and rain streaks accumulation even if only one of these degradations exists. Therefore, another direction to the proposed research is to add an initial step of detecting the type of degradation that exists in the captured images.

- **Experimenting with different variants of sensors**

Regarding future endeavours concerning the challenge of detecting multiple objects based on the proposed research, different approaches could be made; for example, multiple cameras could be used in order to extend the horizontal field of view without inducing significant image distortion. Moreover, dynamic calibration could be introduced to ensure accurate alignment and synchronization between the 2D LiDAR and monocular camera, especially as the vehicle operates in different environments and conditions.

References

[1] SAE On-Road Automated Driving Committee and others, "SAE J3016. taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," .

[2] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli and A. Prioletti, "PROUD—Public road urban driverless-car test," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, *(6),* pp. 3508-3519, 2015.

[3] K. Jo, J. Kim, D. Kim, C. Jang and M. Sunwoo, "Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Trans. Ind. Electron.,* vol. 62, *(8),* pp. 5119-5132, 2015.

[4] C. Ilas, "Electronic sensing technologies for autonomous ground vehicles: A review," in *2013 8th International Symposium on Advanced Topics in Electrical Engineering (Atee),* 2013, pp. 1-6.

[5] M. Hnewa and H. Radha, "Object detection under rainy conditions for autonomous vehicles," *arXiv E-Prints,* pp. arXiv: 2006.16471, 2020.

[6] H. Wang, Y. Wu, M. Li, Q. Zhao and D. Meng, "A survey on rain removal from video and single image," *arXiv Preprint arXiv:1909.08326,* 2019.

[7] F. Rosique, P. J. Navarro, C. Fernández and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors,* vol. 19, *(3),* pp. 648, 2019.

[8] H. Zhu, K. Yuen, L. Mihaylova and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems,* vol. 18, *(10),* pp. 2584-2601, 2017.

[9] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding and J. Paisley, "Removing rain from single images via a deep detail network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 3855-3863.

[10] R. Qian, R. T. Tan, W. Yang, J. Su and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 2482-2491.

[11] Department for Transport, "The pathway to driverless cars: Summary report and action plan," 2015.

[12] R. Sherony and C. Zhang, "Pedestrian and bicyclist crash scenarios in the US," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems,* 2015, pp. 1533-1538.

[13] J. Van Brummelen, M. O'Brien, D. Gruyer and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies,* vol. 89, pp. 384-406, 2018.

[14] SAE On-Road Automated Vehicle Standards Committee and others, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE International: Warrendale, PA, USA,* 2018.

[15] L. Jones, "Driverless cars: when and where?" vol. 12, *(2),* pp. 36-40, 2017.

[16] E. D. Dickmanns, B. Mysliwetz and T. Christians, "An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles," *IEEE Trans. Syst. Man Cybern.,* vol. 20, *(6),* pp. 1273-1284, 1990.

[17] D. Pfeiffer and U. Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in *2010 IEEE Intelligent Vehicles Symposium,* 2010, pp. 217-224.

[18] E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek and J. Schiehlen, "The seeing passenger car'VaMoRs-P'," in *Proceedings of the Intelligent Vehicles' 94 Symposium,* 1994, pp. 68-73.

[19] C. Rouff and M. Hinchey, *Experience from the DARPA Urban Challenge.* Springer Publishing Company, Incorporated, 2011.

[20] M. Walton, "Robots fail to complete grand challenge," *CNN News, Mar,* 2004.

[21] F. Kunz, D. Nuss, J. Wiest, H. Deusch, S. Reuter, F. Gritschneder, A. Scheel, M. Stübler, M. Bach and P. Hatzelmann, "Autonomous driving at ulm university: A modular, robust, and sensor-independent fusion approach," in *2015 IEEE Intelligent Vehicles Symposium (IV),* 2015, pp. 666-673.

[22] D. L. Peters, "Students Working Towards Robotic Chauffeurs: The SAE/GM Autodrive Challenge," *Mechanical Engineering Magazine Select Articles,* vol. 140, *(03),* pp. S6-S11, 2018.

[23] K. Bimbraw, "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology," in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO),* 2015, pp. 191-198.

[24] T. Luettel, M. Himmelsbach and H. Wuensche, "Autonomous ground vehicles—Concepts and a path to the future," *Proc IEEE,* vol. 100, *(Special Centennial Issue),* pp. 1831-1839, 2012.

[25] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny and G. Hoffmann, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics,* vol. 23, *(9),* pp. 661-692, 2006.

[26] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali and C. Geyer, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics,* vol. 25, *(8),* pp. 425-466, 2008.

[27] P. Chatterjee, "Self-driving car pushes sensor technology," *EDN Magazine, on-Line,* 2012.

[28] I. Barabás, A. Todoruţ, N. Cordoş and A. Molea, "Current challenges in autonomous driving," in *IOP Conference Series: Materials Science and Engineering,* 2017, pp. 012096.

[29] M. Martínez-Díaz and F. Soriguera, "Autonomous vehicles: theoretical and practical challenges," *Transportation Research Procedia,* vol. 33, pp. 275-282, 2018.

[30] M. Steger, M. Karner, J. Hillebrand, W. Rom and K. Römer, "A security metric for structured security analysis of cyber-physical systems supporting SAE J3061," in *2016 2nd*

International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data), 2016, pp. 1-6.

[31] M. Maurer, J. C. Gerdes, B. Lenz and H. Winner, "Autonomous driving," *Berlin, Germany: Springer Berlin Heidelberg,* vol. 10, pp. 978-973, 2016.

[32] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser and S. Mammar, "Highly automated driving on highways based on legal safety," *IEEE Transactions on Intelligent Transportation Systems,* vol. 14, *(1),* pp. 333-347, 2012.

[33] A. Mukhtar, L. Xia and T. B. Tang, "Vehicle detection techniques for collision avoidance systems: A review," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, *(5),* pp. 2318-2338, 2015.

[34] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi and M. Porta, "Artificial vision in road vehicles," *Proc IEEE,* vol. 90, *(7),* pp. 1258-1271, 2002.

[35] C. R. Wang and J. J. Lien, "Automatic vehicle detection using local features--a statistical approach," *IEEE Transactions on Intelligent Transportation Systems,* vol. 9, *(1),* pp. 83-96, 2008.

[36] E. U. Haq, S. J. H. Pirzada, J. Piao, T. Yu and H. Shin, "Image processing and vision techniques for smart vehicles," in *2012 IEEE International Symposium on Circuits and Systems,* 2012, pp. 1211-1214.

[37] M. Bertozzi, A. Broggi and S. Castelluccio, "A real-time oriented system for vehicle detection," *J. Syst. Archit.,* vol. 43, *(1-5),* pp. 317-325, 1997.

[38] C. Tzomakas and W. von Seelen, "Vehicle detection in traffic scenes using shadows," in *Ir-Ini, Institut Fur Nueroinformatik, Ruhr-Universitat,* 1998, .

[39] M. B. Van Leeuwen and F. C. Groen, "Vehicle detection with a mobile camera: spotting midrange, distant, and passing cars," *IEEE Robotics & Automation Magazine,* vol. 12, *(1),* pp. 37-43, 2005.

[40] W. For, K. Leman, H. Eng, B. Chew and K. Wan, "A multi-camera collaboration framework for real-time vehicle detection and license plate recognition on highways," in *2008 IEEE Intelligent Vehicles Symposium,* 2008, pp. 192-197.

[41] S. Ieng, J. Vrignon, D. Gruyer and D. Aubert, "A new multi-lanes detection using multi-camera for robust vehicle location," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* 2005, pp. 700-705.

[42] H. T. Niknejad, S. Mita, D. McAllester and T. Naito, "Vision-based vehicle detection for nighttime with discriminately trained mixture of weighted deformable part models," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC),* 2011, pp. 1560-1565.

[43] J. Kim, S. Hong, J. Baek, E. Kim and H. Lee, "Autonomous vehicle detection system using visible and infrared camera," in *2012 12th International Conference on Control, Automation and Systems,* 2012, pp. 630-634.

[44] F. García, A. Prioletti, P. Cerri, A. Broggi, A. de la Escalera and J. M. Armingol, "Visual feature tracking based on phd filter for vehicle detection," in *17th International Conference on Information Fusion (FUSION),* 2014, pp. 1-6.

[45] G. Saldaña González, J. Cerezo Sánchez, M. M. Bustillo Díaz and A. Ata Pérez, "Vision System for the Navigation of a Mobile Robot," *Computación Y Sistemas,* vol. 22, *(1),* pp. 301-308, 2018.

[46] B. Ling, D. R. Gibson and D. Middleton, "Motorcycle detection and counting using stereo camera, IR camera, and microphone array," in *Video Surveillance and Transportation Imaging Applications,* 2013, pp. 86630P.

[47] R. Sen, P. Siriah and B. Raman, "Roadsoundsense: Acoustic sensing based road congestion monitoring in developing regions," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,* 2011, pp. 125-133.

[48] M. Mizumachi, A. Kaminuma, N. Ono and S. Ando, "Robust sensing of approaching vehicles relying on acoustic cues," *Sensors,* vol. 14, *(6),* pp. 9546-9561, 2014.

[49] K. Yoon, Y. Song and M. Jeon, "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views," *IET Image Processing,* vol. 12, *(7),* pp. 1175-1184, 2018.

[50] A. Mousavian, D. Anguelov, J. Flynn and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 7074-7082.

[51] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016, pp. 2147-2156.

[52] Y. L. Chen, M. R. Jahanshahi, P. Manjunatha, W. Gan, M. Abdelbarr, S. F. Masri, B. Becerik-Gerber and J. P. Caffrey, "Inexpensive multimodal sensor fusion system for autonomous data acquisition of road surface conditions," *IEEE Sensors Journal,* vol. 16, *(21),* pp. 7731-7743, 2016.

[53] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler and R. Urtasun, "3d object proposals using stereo imagery for accurate object class detection," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 40, *(5),* pp. 1259-1272, 2017.

[54] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016, pp. 808-816.

[55] J. Lahoud and B. Ghanem, "2d-driven 3d object detection in rgb-d images," in *Proceedings of the IEEE International Conference on Computer Vision,* 2017, pp. 4622-4630.

[56] Y. Wang, W. Chao, D. Garg, B. Hariharan, M. Campbell and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2019, pp. 8445-8453.

[57] Y. You, Y. Wang, W. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell and K. Q. Weinberger, "Pseudo-LiDAR : Accurate Depth for 3D Object Detection in Autonomous Driving," *arXiv Preprint arXiv:1906.06310,* 2019.

[58] Z. Wang, Y. Wu and Q. Niu, "Multi-sensor fusion in automated driving: A survey," *IEEE Access,* vol. 8, pp. 2847-2868, 2019.

[59] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto and U. J. Nunes, "Multimodal vehicle detection: fusing 3D-LIDAR and color camera data," *Pattern Recog. Lett.,* vol. 115, pp. 20-29, 2018.

[60] X. Zhang, W. Xu, C. Dong and J. M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," in *2017 IEEE Intelligent Vehicles Symposium (IV),* 2017, pp. 54-59.

[61] T. Taipalus and J. Ahtiainen, "Human detection and tracking with knee-high mobile 2D LIDAR," in *2011 IEEE International Conference on Robotics and Biomimetics,* 2011, pp. 1672-1677.

[62] X. Shao, H. Zhao, K. Nakamura, K. Katabira, R. Shibasaki and Y. Nakagawa, "Detection and tracking of multiple pedestrians by using laser range scanners," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2007, pp. 2174-2179.

[63] Z. Rozsa and T. Sziranyi, "Obstacle prediction for automated guided vehicles based on point clouds measured by a tilted LIDAR sensor," *IEEE Transactions on Intelligent Transportation Systems,* vol. 19, *(8),* pp. 2708-2720, 2018.

[64] F. García, F. Jiménez, J. E. Naranjo, J. G. Zato, F. Aparicio, J. M. Armingol and A. de la Escalera, "Environment perception based on LIDAR sensors for real road applications," *Robotica,* vol. 30, *(2),* pp. 185-193, 2012.

[65] D. Dai, Z. Chen, P. Bao and J. Wang, "A Review of 3D Object Detection for Autonomous Driving of Electric Vehicles," *World Electric Vehicle Journal,* vol. 12, *(3),* pp. 139, 2021.

[66] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang and H. Li, "PV-RCNN : Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection," *arXiv Preprint arXiv:2102.00463,* 2021.

[67] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 4490-4499.

[68] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 652-660.

[69] C. R. Qi, W. Liu, C. Wu, H. Su and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 918-927.

[70] Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 2019, pp. 1951-1960.

[71] C. Hung, A. T. Lin, B. C. Peng, H. Wang, J. Hsu, Y. Lu, W. Hsu, J. C. Zhan, B. Juan and C. Lok, "9.1 toward automotive surround-view radars," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC),* 2019, pp. 162-164.

[72] G. Zhang, H. Li and F. Wenger, "Object Detection and 3D Estimation via an FMCW Radar Using a Fully Convolutional Network," *arXiv Preprint arXiv:1902.05394,* 2019.

[73] L. Arnone and P. Vicari, "Simultaneous odometry, mapping and object tracking with a compact automotive radar," in *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE),* 2019, pp. 1-6.

[74] M. Kam, X. Zhu and P. Kalata, "Sensor fusion for mobile robot navigation," *Proc IEEE,* vol. 85, *(1),* pp. 108-119, 1997.

[75] B. V. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proc IEEE,* vol. 85, *(1),* pp. 24-38, 1997.

[76] R. C. Luo, C. Yih and K. L. Su, "Multisensor fusion and integration: approaches, applications, and future research directions," *IEEE Sensors Journal,* vol. 2, *(2),* pp. 107-119, 2002.

[77] F. Castanedo, "A review of data fusion techniques," *ScientificWorldJournal,* vol. 2013, pp. 704504, Oct 27, 2013.

[78] X. Chen, H. Ma, J. Wan, B. Li and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 1907-1915.

[79] J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2018, pp. 1-8.

[80] X. Du, M. H. Ang, S. Karaman and D. Rus, "A general pipeline for 3d detection of vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA),* 2018, pp. 3194-3200.

[81] F. Garcia, D. Martin, A. De La Escalera and J. M. Armingol, "Sensor fusion methodology for vehicle detection," *IEEE Intelligent Transportation Systems Magazine,* vol. 9, *(1),* pp. 123-133, 2017.

[82] M. Liang, B. Yang, S. Wang and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proceedings of the European Conference on Computer Vision (ECCV),* 2018, pp. 641-656.

[83] P. Wei, L. Cagle, T. Reza, J. Ball and J. Gafford, "LiDAR and camera detection fusion in a real-time industrial multi-sensor collision avoidance system," *Electronics,* vol. 7, *(6),* pp. 84, 2018.

[84] A. Rövid and V. Remeli, "Towards raw sensor fusion in 3D object detection," in *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI),* 2019, pp. 293-298.

[85] J. Xue, D. Wang, S. Du, D. Cui, Y. Huang and N. Zheng, "A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars,"

Frontiers of Information Technology & Electronic Engineering, vol. 18, *(1),* pp. 122-138, 2017.

[86] J. Han, Y. Liao, J. Zhang, S. Wang and S. Li, "Target Fusion Detection of LiDAR and Camera Based on the Improved YOLO Algorithm," *Mathematics,* vol. 6, *(10),* pp. 213, 2018.

[87] F. García, J. García, A. Ponz, A. De La Escalera and J. M. Armingol, "Context aided pedestrian detection for danger estimation based on laser scanner and computer vision," *Expert Syst. Appl.,* vol. 41, *(15),* pp. 6646-6661, 2014.

[88] Y. Shi, S. Ji, X. Shao, P. Yang, W. Wu, Z. Shi and R. Shibasaki, "Fusion of a panoramic camera and 2D laser scanner data for constrained bundle adjustment in GPS-denied environments," *Image Vision Comput.,* vol. 40, pp. 28-37, 2015.

[89] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA),* 2015, pp. 2174-2181.

[90] A. Palffy, J. F. Kooij and D. M. Gavrila, "Occlusion aware sensor fusion for early crossing pedestrian detection," in *2019 IEEE Intelligent Vehicles Symposium (IV),* 2019, pp. 1768-1774.

[91] S. Chadwick, W. Maddern and P. Newman, "Distant vehicle detection using radar and vision," in *International Conference on Robotics and Automation (ICRA),* 2019, .

[92] H. Jha, V. Lodhi and D. Chakravarty, "Object detection and identification using vision and radar data fusion system for ground-based navigation," in *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN),* 2019, pp. 590-593.

[93] S. Steyer, C. Lenk, D. Kellner, G. Tanzmeister and D. Wollherr, "Grid-Based Object Tracking With Nonlinear Dynamic State and Shape Estimation," *IEEE Transactions on Intelligent Transportation Systems,* 2019.

[94] A. Ahrabian, M. Emambakhsh, M. Sheeny and A. Wallace, "Efficient multi-sensor extended target tracking using GM-PHD filter," in *2019 IEEE Intelligent Vehicles Symposium (IV),* 2019, pp. 1731-1738.

[95] L. Daniel, D. Phippen, E. Hoare, A. Stove, M. Cherniakov and M. Gashinova, "Low-THz radar, lidar and optical imaging through artificially generated fog," 2017.

[96] H. Cho, Y. Seo, B. V. Kumar and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA),* 2014, pp. 1836-1843.

[97] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, *(2),* pp. 91-110, 2004.

[98] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2005, .

[99] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings. International Conference on Image Processing,* 2002, pp. I.

[100] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2014, pp. 580-587.

[101] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision,* 2015, pp. 1440-1448.

[102] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems,* 2015, pp. 91-99.

[103] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision,* 2016, pp. 21-37.

[104] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016, pp. 779-788.

[105] Y. Li, R. Bu, M. Sun, W. Wu, X. Di and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems,* 2018, pp. 820-830.

[106] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2015, pp. 922-928.

[107] Y. Xiang, W. Choi, Y. Lin and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2015, pp. 1903-1911.

[108] D. Xu, D. Anguelov and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 244-253.

[109] 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 *2017 IEEE International Conference on Robotics and Automation (ICRA),* 2017, pp. 1355-1361.

[110] J. Chang and Y. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 5410-5418.

[111] L. Guan, Y. Chen, G. Wang and X. Lei, "Real-Time Vehicle Detection Framework Based on the Fusion of LiDAR and Camera," *Electronics,* vol. 9, *(3),* pp. 451, 2020.

[112] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2017, pp. 1513-1518.

[113] B. Li, T. Zhang and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in *Robotics: Science and Systems,* 2016, .

[114] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems,* 2019.

[115] Y. Xiang, W. Choi, Y. Lin and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV),* 2017, pp. 924-933.

[116] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal Process Image Commun,* vol. 47, pp. 482-489, 2016.

[117] Z. Zhao, H. Bian, D. Hu, W. Cheng and H. Glotin, "Pedestrian detection based on fast R-CNN and batch normalization," in *International Conference on Intelligent Computing,* 2017, pp. 735-746.

[118] A. Brunetti, D. Buongiorno, G. F. Trotta and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing,* vol. 300, pp. 17-33, 2018.

[119] R. Benenson, M. Omran, J. Hosang and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *European Conference on Computer Vision,* 2014, pp. 613-627.

[120] K. Wang and W. Zhou, "Pedestrian and cyclist detection based on deep neural network fast R-CNN," *International Journal of Advanced Robotic Systems,* vol. 16, *(2),* pp. 1729881419829651, 2019.

[121] K. Saleh, M. Hossny, A. Hossny and S. Nahavandi, "Cyclist detection in LIDAR scans using faster R-CNN and synthetic depth images," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC),* 2017, pp. 1-6.

[122] R. Huang, J. Pedoeem and C. Chen, "YOLO-LITE: A real-time object detection algorithm optimized for non-GPU computers," in *2018 IEEE International Conference on Big Data (Big Data),* 2018, pp. 2503-2510.

[123] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 7263-7271.

[124] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv Preprint arXiv:1804.02767,* 2018.

[125] M. Simon, S. Milz, K. Amende and H. Gross, "Complex-YOLO: An euler-region-proposal for real-time 3D object detection on point clouds," in *European Conference on Computer Vision,* 2018, pp. 197-209.

[126] Z. Liu, Z. Chen, Z. Li and W. Hu, "An Efficient Pedestrian Detection Method Based on YOLOv2," *Mathematical Problems in Engineering,* vol. 2018, 2018.

[127] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection." in *Robotics: Science and Systems,* 2015, pp. 10.15607.

[128] M. Liang, B. Yang, Y. Chen, R. Hu and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2019, pp. 7345-7353.

[129] K. Shin, Y. P. Kwon and M. Tomizuka, "Roarnet: A robust 3d object detection based on region approximation refinement," in *2019 IEEE Intelligent Vehicles Symposium (IV),* 2019, pp. 2510-2515.

[130] Z. Gong, H. Lin, D. Zhang, Z. Luo, J. Zelek, Y. Chen, A. Nurunnabi, C. Wang and J. Li, "A Frustum-based probabilistic framework for 3D object detection by fusion of LiDAR and camera data," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 159, pp. 90-100, 2020.

[131] J. Dou, J. Xue and J. Fang, "SEG-VoxelNet for 3D vehicle detection from RGB and LiDAR data," in *2019 International Conference on Robotics and Automation (ICRA),* 2019, pp. 4362-4368.

[132] V. A. Sindagi, Y. Zhou and O. Tuzel, "Mvx-net: Multimodal voxelnet for 3d object detection," in *2019 International Conference on Robotics and Automation (ICRA),* 2019, pp. 7276-7282.

[133] J. H. Yoo, Y. Kim, J. Kim and J. W. Choi, "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16,* 2020, pp. 720-736.

[134] L. Xie, C. Xiang, Z. Yu, G. Xu, Z. Yang, D. Cai and X. He, "PI-RCNN: An efficient multi-sensor 3D object detector with point-based attentive cont-conv fusion module," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2020, pp. 12460-12467.

[135] D. Dai, J. Wang, Z. Chen and H. Zhao, "Image guidance based 3D vehicle detection in traffic scene," *Neurocomputing,* vol. 428, pp. 1-11, 2021.

[136] T. Huang, Z. Liu, X. Chen and X. Bai, "Epnet: Enhancing point features with image semantics for 3d object detection," in *European Conference on Computer Vision,* 2020, pp. 35-52.

[137] K. Garg and S. K. Nayar, "Vision and rain," *International Journal of Computer Vision,* vol. 75, *(1),* pp. 3-27, 2007.

[138] K. Yoneda, N. Suganuma, R. Yanase and M. Aldibaja, "Automated driving recognition technologies for adverse weather conditions," *IATSS Research,* vol. 43, *(4),* pp. 253-262, 2019. Available: https://www.sciencedirect.com/science/article/pii/S0386111219301463. DOI: https://doi.org/10.1016/j.iatssr.2019.11.005.

[139] W. Yang, R. T. Tan, J. Feng, Z. Guo, S. Yan and J. Liu, "Joint rain detection and removal from a single image with contextualized deep networks," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 42, *(6),* pp. 1377-1393, 2019.

[140] R. Li, L. Cheong and R. T. Tan, "Single image deraining using scale-aware multi-stage recurrent network," *arXiv Preprint arXiv:1712.06830,* 2017.

[141] D. Ren, W. Zuo, Q. Hu, P. Zhu and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2019, pp. 3937-3946.

[142] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang and R. W. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2019, pp. 12270-12279.

[143] J. Xu, W. Zhao, P. Liu and X. Tang, "Removing rain and snow in a single image using guided filter," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE),* 2012, pp. 304-307.

[144] X. Zheng, Y. Liao, W. Guo, X. Fu and X. Ding, "Single-image-based rain and snow removal using multi-guided filter," in *International Conference on Neural Information Processing,* 2013, pp. 258-265.

[145] X. Ding, L. Chen, X. Zheng, Y. Huang and D. Zeng, "Single image rain and snow removal via guided L0 smoothing filter," *Multimedia Tools Appl,* vol. 75, *(5),* pp. 2697-2712, 2016.

[146] J. Kim, C. Lee, J. Sim and C. Kim, "Single-image deraining using an adaptive nonlocal means filter," in *2013 IEEE International Conference on Image Processing,* 2013, pp. 914-917.

[147] Y. Fu, L. Kang, C. Lin and C. Hsu, "Single-frame-based rain removal via image decomposition," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 2011, pp. 1453-1456.

[148] D. Chen, C. Chen and L. Kang, "Visual depth guided color image rain streaks removal using sparse coding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 24, *(8),* pp. 1430-1455, 2014.

[149] L. Kang, C. Lin, C. Lin and Y. Lin, "Self-learning-based rain streak removal for image/video," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS),* 2012, pp. 1871-1874.

[150] Y. Wang, S. Liu, C. Chen and B. Zeng, "A hierarchical approach for rain or snow removing in a single color image," *IEEE Trans. Image Process.,* vol. 26, *(8),* pp. 3936-3950, 2017.

[151] S. Sun, S. Fan and Y. F. Wang, "Exploiting image structural similarity for single image rain removal," in *2014 IEEE International Conference on Image Processing (ICIP),* 2014, pp. 4482-4486.

[152] S. Gu, D. Meng, W. Zuo and L. Zhang, "Joint convolutional analysis and synthesis sparse representation for single image layer separation," in *Proceedings of the IEEE International Conference on Computer Vision,* 2017, pp. 1708-1716.

[153] J. Starck, Y. Moudden, J. Bobin, M. Elad and D. L. Donoho, "Morphological component analysis," in *Wavelets XI,* 2005, pp. 209-223.

[154] X. Zhao, P. Liu, J. Liu and T. Xianglong, "The application of histogram on rain detection in video," in *11th Joint International Conference on Information Sciences,* 2008, pp. 382-387.

[155] M. Wang, J. Mai, R. Cai, Y. Liang and H. Wan, "Single image deraining using deep convolutional networks," *Multimedia Tools Appl,* vol. 77, *(19),* pp. 25905-25918, 2018.

[156] X. Wang, Z. Li, H. Shan, Z. Tian, Y. Ren and W. Zhou, "Fastderainnet: A deep learning algorithm for single image deraining," *IEEE Access,* vol. 8, pp. 127622-127630, 2020.

[157] M. Dekan, F. Duchoň, A. Babinec, P. Hubinský, M. Kajan and M. Szabova, "Versatile approach to probabilistic modeling of Hokuyo UTM-30LX," *IEEE Sensors Journal,* vol. 16, *(6),* pp. 1814-1828, 2015.

[158] X. Li, G. M. Xiong, Y. W. Hu, W. B. Li, Y. Jiang, J. W. Gong and H. Y. Chen, "Comparison of single-layer and multi-layer laser scanners for measuring characteristic," in *Applied Mechanics and Materials,* 2012, pp. 548-552.

[159] J. P. Espineira, J. Robinson, J. Groenewald, P. H. Chan and V. Donzella, "Realistic LiDAR with noise model for real-time testing of automated vehicles in a virtual environment," *IEEE Sensors Journal,* vol. 21, *(8),* pp. 9919-9926, 2021.

[160] C. Wang, M. Shen and C. Yao, "Rain streak removal by multi-frame-based anisotropic filtering," *Multimedia Tools Appl,* vol. 76, *(2),* pp. 2019-2038, 2017.

[161] X. Fu, J. Huang, X. Ding, Y. Liao and J. Paisley, "Clearing the skies: A deep network architecture for single-image rain removal," *IEEE Trans. Image Process.,* vol. 26, *(6),* pp. 2944-2956, 2017.

[162] M. Li, Q. Xie, Q. Zhao, W. Wei, S. Gu, J. Tao and D. Meng, "Video rain streak removal by multiscale convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 6644-6653.

[163] F. Nashashibi, R. de Charrette and A. Lia, "Detection of unfocused raindrops on a windscreen using low level image processing," in *2010 11th International Conference on Control Automation Robotics & Vision,* 2010, pp. 1410-1415.

[164] F. Al Machot, M. Ali, A. H. Mosa, C. Schwarzlmüller, M. Gutmann and K. Kyamakya, "Real-time raindrop detection based on cellular neural networks for ADAS," *Journal of Real-Time Image Processing,* vol. 16, *(4),* pp. 931-943, 2019.

[165] S. Krishnan and D. Venkataraman, "Restoration of video by removing rain," *International Journal of Computer Science, Engineering and Applications,* vol. 2, *(2),* pp. 19, 2012.

[166] S. You, R. T. Tan, R. Kawakami and K. Ikeuchi, "Adherent raindrop detection and removal in video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2013, pp. 1035-1042.

[167] W. Ren, J. Tian, Z. Han, A. Chan and Y. Tang, "Video desnowing and deraining based on matrix decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 4210-4219.

[168] H. Liao, D. Wang, C. Yang and J. Shine, "Video-based water drop detection and removal method for a moving vehicle," *Information Technology Journal,* vol. 12, *(4),* pp. 569-583, 2013.

[169] J. Chen and L. Chau, "A rain pixel recovery algorithm for videos with highly dynamic scenes," *IEEE Trans. Image Process.,* vol. 23, *(3),* pp. 1097-1104, 2013.

[170] W. Yang, J. Liu and J. Feng, "Frame-consistent recurrent video deraining with dual-level flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2019, pp. 1661-1670.

[171] C. Yeh, C. Lin, K. Muchtar and P. Liu, "Rain streak removal based on non-negative matrix factorization," *Multimedia Tools Appl,* vol. 77, *(15),* pp. 20001-20020, 2018.

[172] J. Chen, C. Tan, J. Hou, L. Chau and H. Li, "Robust video content alignment and compensation for rain removal in a cnn framework," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 6286-6295.

[173] S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. Cesar-Junior, J. Zhang, X. Guo and X. Cao, "Single image deraining: A comprehensive benchmark analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2019, pp. 3838-3847.

[174] K. Garg and S. K. Nayar, "When does a camera see rain?" in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1,* 2005, pp. 1067-1074.

[175] Y. Chen and C. Hsu, "A generalized low-rank appearance model for spatio-temporally correlated rain streaks," in *Proceedings of the IEEE International Conference on Computer Vision,* 2013, pp. 1968-1975.

[176] J. Bossu, N. Hautiere and J. Tarel, "Rain or snow detection in image sequences through use of a histogram of orientation of streaks," *International Journal of Computer Vision,* vol. 93, pp. 348-367, 2011.

[177] P. C. Barnum, S. Narasimhan and T. Kanade, "Analysis of rain and snow in frequency space," *International Journal of Computer Vision,* vol. 86, pp. 256-274, 2010.

[178] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo and S. Yan, "Deep joint rain detection and removal from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 1357-1366.

[179] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016, pp. 770-778.

[180] K. Jiang, Z. Wang, P. Yi, C. Chen, B. Huang, Y. Luo, J. Ma and J. Jiang, "Multi-scale progressive fusion network for single image deraining," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2020, pp. 8346-8355.

[181] A. Abdelhamed, R. Timofte and M. S. Brown, "Ntire 2019 challenge on real image denoising: Methods and results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops,* 2019, pp. 0.

[182] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo and C. Lin, "Deep learning on image denoising: An overview," *Neural Networks,* vol. 131, pp. 251-275, 2020.

[183] K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Trans. Image Process.,* vol. 26, *(7),* pp. 3142-3155, 2017.

[184] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv Preprint arXiv:1511.07122,* 2015.

[185] H. Zhang and V. M. Patel, "Density-aware single image de-raining using a multi-stream dense network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 695-704.

[186] X. Li, J. Wu, Z. Lin, H. Liu and H. Zha, "Recurrent squeeze-and-excitation context aggregation net for single image deraining," in *Proceedings of the European Conference on Computer Vision (ECCV),* 2018, pp. 254-269.

[187] X. Fu, B. Liang, Y. Huang, X. Ding and J. Paisley, "Lightweight pyramid networks for image deraining," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 31, *(6),* pp. 1794-1807, 2019.

[188] Z. Fan, H. Wu, X. Fu, Y. Hunag and X. Ding, "Residual-guide feature fusion network for single image deraining," *arXiv Preprint arXiv:1804.07493,* 2018.

[189] S. Waqas Zamir, A. Arora, S. Khan, M. Hayat, F. Shahbaz Khan, M. Yang and L. Shao, "Multi-Stage Progressive Image Restoration," *arXiv E-Prints,* pp. arXiv: 2102.02808, 2021.

[190] L. Chen, X. Lu, J. Zhang, X. Chu and C. Chen, "HINet: Half instance normalization network for image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2021, pp. 182-192.

[191] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention,* 2015, pp. 234-241.

[192] Met Office, "UK temperature, rainfall and sunshine time series," Available: https://www.metoffice.gov.uk/research/climate/maps-and-data/uk-temperature-rainfall-and-sunshine-time-series.

[193] A. K. Tripathi and S. Mukhopadhyay, "A probabilistic approach for detection and removal of rain from videos," *IETE Journal of Research,* vol. 57, *(1),* pp. 82-91, 2011.

[194] R. Qian, R. T. Tan, W. Yang, J. Su and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2018, pp. 2482-2491.

[195] W. -S. Lai, J. -B. Huang, Z. Hu, N. Ahuja and M. -H. Yang, "A comparative study for single image blind deblurring," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2016, pp. 1701-1709. DOI: 10.1109/CVPR.2016.188.

[196] C. Wang, Q. Liu, Y. Li and M. Gao, "LightCSPNet: A Lightweight Network for Image Classification and Objection Detection," *International Journal of Computational Intelligence Systems,* vol. 16, *(1),* pp. 46, 2023.

[197] M. S. Ebrahimi and H. K. Abadi, "Study of residual networks for image recognition," in *Intelligent Computing*Anonymous Springer, 2021, pp. 754-763.

[198] B. Cai, X. Xu, K. Jia, C. Qing and D. Tao, "Dehazenet: An end-to-end system for single image haze removal," *IEEE Trans. Image Process.,* vol. 25, *(11),* pp. 5187-5198, 2016.

[199] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml,* 2010, .

[200] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.,* vol. 13, *(4),* pp. 600-612, 2004.

[201] S. G. Narasimhan and S. K. Nayar, "Vision and the atmosphere," *International Journal of Computer Vision,* vol. 48, *(3),* pp. 233-254, 2002.

[202] Z. Li, J. Zhang, R. Zhong, B. Bhanu, Y. Chen, Q. Zhang and H. Tang, "Lightweight and Efficient Image Dehazing Network Guided by Transmission Estimation from Real-World Hazy Scenes," *Sensors,* vol. 21, *(3),* pp. 960, 2021.

[203] X. Mao, C. Shen and Y. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," *Advances in Neural Information Processing Systems,* vol. 29, 2016.

[204] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics,* 2010, pp. 249-256.

[205] D. Clevert, T. Unterthiner and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv Preprint arXiv:1511.07289,* 2015.

[206] M. Haris and A. Glowacz, "Road Object Detection: A Comparative Study of Deep Learning-Based Algorithms," *Electronics,* vol. 10, *(16),* pp. 1932, 2021.

[207] (). *Scanning Rangefinder Distance Data Output/UTM-30LX Product Details | HOKUYO AUTOMATIC CO., LTD.* Available: Scanning Rangefinder Distance Data Output/UTM-30LX Product Details | HOKUYO AUTOMATIC CO., LTD.

[208] J. Redmon, "Darknet," *Darknet: Open Source Neural Networks in C,* 2013.

[209] pjreddie, "darknet," 2018. Available: https://github.com/pjreddie/darknet.

[210] N. Charron, S. Phillips and S. L. Waslander, "De-noising of lidar point clouds corrupted by snowfall," in *2018 15th Conference on Computer and Robot Vision (CRV),* 2018, pp. 254-261.

[211] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. no. 98CH36271),* 1998, pp. 839-846.

[212] S. B. Gokturk, H. Yalcin and C. Bamji, "A time-of-flight depth sensor-system description, issues and solutions," in *2004 Conference on Computer Vision and Pattern Recognition Workshop,* 2004, pp. 35.

[213] Y. Li, J. Li, L. Wang, J. Zhang, D. Li and M. Zhang, "A weighted least squares algorithm for time-of-flight depth image denoising," *Optik,* vol. 125, *(13),* pp. 3283-3286, 2014.

[214] Z. Fang, S. Zhao, S. Wen and Y. Zhang, "A Real-Time 3D Perception and Reconstruction System Based on a 2D Laser Scanner," *Journal of Sensors,* vol. 2018, 2018.

[215] D. Choi, Y. Bok, J. Kim, I. Shim and I. Kweon, "Structure-From-Motion in 3D Space Using 2D Lidars," *Sensors,* vol. 17, *(2),* pp. 242, 2017.

[216] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition,* 2010, pp. 2366-2369.

[217] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.,* vol. 13, *(4),* pp. 600-612, 2004.

[218] A. Mittal, R. Soundararajan and A. C. Bovik, "Making a “completely blind” image quality analyzer," *IEEE Signal Process. Lett.,* vol. 20, *(3),* pp. 209-212, 2012.

[219] L. Liu, B. Liu, H. Huang and A. C. Bovik, "No-reference image quality assessment based on spatial and spectral entropies," *Signal Process Image Commun,* vol. 29, *(8),* pp. 856-863, 2014.

[220] H. Zhang, V. Sindagi and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *IEEE Transactions on Circuits and Systems for Video Technology,* 2019.

[221] G. Schaefer and M. Stich, "UCID: An uncompressed color image database," in *Storage and Retrieval Methods and Applications for Multimedia 2004,* 2003, pp. 472-480.

[222] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 33, *(5),* pp. 898-916, 2010.

[223] D. R. Martin, C. C. Fowlkes and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 26, *(5),* pp. 530-549, 2004. . DOI: 10.1109/TPAMI.2004.1273918.

[224] N. Silberman, D. Hoiem, P. Kohli and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision,* 2012, pp. 746-760.

[225] S. G. Narasimhan and S. K. Nayar, "Vision and the atmosphere," *International Journal of Computer Vision,* vol. 48, *(3),* pp. 233-254, 2002.

[226] C. O. Ancuti, C. Ancuti, M. Sbert and R. Timofte, "Dense-haze: A benchmark for image dehazing with dense-haze and haze-free images," in *2019 IEEE International Conference on Image Processing (ICIP),* 2019, pp. 1014-1018.

[227] C. O. Ancuti, C. Ancuti, R. Timofte and C. De Vleeschouwer, "O-haze: A dehazing benchmark with real hazy and haze-free outdoor images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops,* 2018, pp. 754-762.

[228] C. Ancuti, C. O. Ancuti, R. Timofte and C. D. Vleeschouwer, "I-HAZE: A dehazing benchmark with real hazy and haze-free indoor images," in *International Conference on Advanced Concepts for Intelligent Vision Systems,* 2018, pp. 620-631.

[229] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition,* 2012, pp. 3354-3361.

[230] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "The KITTI vision benchmark suite," *URL Http://Www.Cvlibs.Net/Datasets/Kitti,* 2015.

[231] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision,* 2014, pp. 740-755.

[232] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2020, pp. 2636-2645.

[233] M. Everingham, L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The PASCAL visual object classes challenge 2007 (VOC2007) results," 2007.

[234] M. Everingham and J. Winn, "The pascal visual object classes challenge 2012 (voc2012) development kit," *Pattern Analysis, Statistical Modelling and Computational Learning, Tech.Rep,* 2011.

[235] J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition,* 2009, pp. 248-255.

[236] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016, pp. 4040-4048.

[237] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2015, pp. 1912-1920.

[238] Anonymous "ModelNet," 2015. Available: https://modelnet.cs.princeton.edu/#.

[239] Y. Xiang and S. Savarese, "Object detection by 3d aspectlets and occlusion reasoning," in *Proceedings of the IEEE International Conference on Computer Vision Workshops,* 2013, pp. 530-537.

[240] The University of Sydney, "Sydney Urban Objects Dataset," 2013. Available: http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml.

Appendix A List of Publications

- Khatab, E., Onsy, A., Varley, M., & Abouelfarag, A. (2021). Vulnerable objects detection for autonomous driving: A review. *Integration*, *78*, 36-48.
    - https://doi.org/10.1016/j.vlsi.2021.01.002

- Khatab, E., Onsy, A., & Abouelfarag, A. (2022). Evaluation of 3D Vulnerable Objects' Detection Using a Multi-Sensors System for Autonomous Vehicles. *Sensors*, *22*(4), 1663.
    - https://doi.org/10.3390/s22041663
- Khatab, E., Onsy, A., Varley, M., & Abouelfarag, A. (2023). A Lightweight Network for Real-Time Rain Streaks and Rain Accumulation Removal from Single Images Captured by AVs. *Applied Sciences*, *13*(1), 219.
    - https://doi.org/10.3390/app13010219

Appendix B Hokuyo UTM-30LX Specifications

As stated by the manufacturer, these are the specifications of Hokuyo UTM-30LX 2D LiDAR:

Model No.	**Hokuyo UTM-30LX**
Range	30m
Angular Resolution	0.25°
scanning angle	270°
Light source	Laser semiconductor 870nm, Laser class 1
Supply voltage	12 VDC ± 10%
Supply Current	maximum of 1A, normal is 0.7A
Power consumption	Less than 8W
Accuracy	It can be affected by strong light such as direct sunlight: • Under 3000lx: White Kent sheet: ± 30mm (0.1m to 10m) • Under 100000lx: White Kent sheet: ± 50mm (0.1m to 10m)
Detection range and detection object:	• Guaranteed Range: 0.1-30m (White Kent sheet) • Maximum Range: 0.1-60m • Minimum Width detected at 10m: 130mm
Measurement Resolution and Repeated Accuracy	• 1mm • Under 3000lx: 10mm (White Kent sheet up to 10m) • Under 100000lx: 30mm (White Kent sheet up to 10m)
Scan time	25 msec/scan
Sound level	Less than 25dB
Connection	• Power and synchronous output: 2m flying lead wire • USB: 2m cable with type-A connector
Ambient (Temperature/Humidity)	-10 to 50°C, less than 85% relative humidity

Appendix C Vehicles' Ground Clearance

In the table below, a comparison between a big range of different vehicles' ground clearance is shown, in order to get an overview of the average range, maximum, and minimum values.

Manufacturer	Model	Ground Clearance (mm)
SUVs/ Cross Over		
Land Rover	Defender	323
	Discovery 4	301
	Range Rover	295
	Range Rover Evoque	215
Jeep	Grand Cherokee	287
	Wrangler	254
	Cherokee	221
	Renegade Trailhawk	220
	Compass	205
	Renegade	200
Toyota	Hilux	286
	FJ Cruiser	245
	Land Cruiser 79	235
	Land Cruiser 200	230
	Land Cruiser 76	230
	Fortuner	220
VW	Toureg	243
	Tiguan	175
Nissan	NP300	240
	Patrol	230
	X-Trail	209
	Qashqai	182
	Juke	180
	NP200	177
Volvo	XC90	238
	XC60	230

	V70 Cross Country	210
	V40 Cross Country	144
Ford	Ranger	237
	Everest	225
	Eco sport	200
	Kuga	197
Audi	Q7	235
	Q5	200
	Q3	170
Mazda	BT-50	232
	CX-5	215
Mitsubishi	Pajero	225
	Outlander Sport	216
	ASX	195
Subaru	Forester	220
	Outback	213
Isuzu	KB300	220
Chevrolet	Trailblazer	218
Mercedes	GL class	215
	GLE	215
	G-class	206
	GLA	183
	GLC	181
Porsche	Cayenne	215
	Macan	198
BMW	X6	210
	X5	209
	X3, X4	204
	X1	183
Mahindra	Scorpio Bakkie	210
	Bolero	200
	Genio	195
	Xylo	186

	Scorpio SUV	180
Renault	Koleos	206
	Duster	205
	Captur	200
	Sandero Stepway	193
Suzuki	Grand Vitara	200
	Jimmy	190
	Vitara	185
Citroen	C4 Aircross	200
Honda	HR-V	185
	CR-V	170
Dodge	Journey	185
Hyundai	Sante Fe	185
	Tucson	172
	IX35	170
Kia	Sorento	185
Fiat	500X	179
	Panda 4x4	150
Chevrolet	Captiva	178
Peugeot	2008	165
Mini	Countryman	160
Opel	Mokka	157
Buses		
In some areas buses are required to have a ground clearance of at least 100 mm. Too much ground clearance can cause the vehicle to have an excessively high center of gravity, which could cause the vehicle to be unstable or even flip		
Trucks		
238.76 – 559 mm		

Appendix D Code Samples for LiDAR Measurements

This appendix includes samples from the C++ code that was implemented to manipulate LiDAR measurements. Full work has been represented in Chapter 3**.**

```cpp
//.............................Constants.............................
#define HP_HFOV 75.37
#define Logitech_HFOV 61.0
#define HP_Video_Width 640.0
#define Logitech_Video_Width 640.0
#define LiDAR_resolution 0.25
#define PI 3.14159265
//.........Functions related to LIDAR operation.......................
namespace
{   //Prints LiDAR data
    void print_data(const Urg_driver& urg,
        const vector<long>& data, long time_stamp)
    {
        double radian;
        // Shows only the front step
        int front_index = urg.step2index(0);
        radian = urg.index2rad(front_index);
        cout << "First index data:   " << data[front_index] << " [mm], ("
            << time_stamp << " [msec])" << endl;
    }
    //Prints measurements in steps
    void print_step2data(const Urg_driver& urg, const vector<long>& data, long time_stamp, int step)
    {
        int index = urg.step2index(step);
        cout << "Data in Step (" << step << ") " << data[index] << " [mm], ("
            << time_stamp << " [msec])" << endl;
    }
    //Prints all data
    void print_all_data(const Urg_driver& urg, const vector<long>& data, long time_stamp)
    {
        size_t data_n = data.size();
        cout << "data size is: " << data_n << " time stamp: " << time_stamp << endl;
        for (size_t i = 0; i < data_n; ++i)
        {
            cout << "data#: " << i << ":  " << data[i] << "[mm]" << endl;
        }
    }
    //Converts angles to distances
    vector<float> angles_to_distances(const Urg_driver& urg, const vector<long>& data, long time_stamp, float start_angle, float end_angle)
    {
        int start_step = start_angle / LiDAR_resolution;
        int end_step = end_angle / LiDAR_resolution;
        vector<float> distances;
        int i_index;
        float distance;
        for (int i = start_step; i <= end_step; i++)
        {
            i_index = urg.step2index(i);
            distance = data[i_index] * 0.1;
            distances.push_back(distance);
        }
        return distances;
```

```
53      }
54
55      //Converts radii to linear measurements
56      vector<float> radii_to_linear(vector<float> radii, float start_angle,
          float end_angle)
57      {
58          vector<float> angles;
59          for (int i = start_angle; i <= end_angle; i = i + 0.25)
60          {
61              angles.push_back(i);
62          }
63          int sizes = angles.size();
64          vector<float> linear_distances;
65          float linear_distance;
66
67          for (int i = 0; i < sizes; i++)
68          {
69              linear_distance = sin(angles[i]) * radii[i];
70              linear_distances.push_back(linear_distance);
71          }
72          return linear_distances;
73      }
74      //Converts from degree to LiDAR measurement data
75      int degree_to_data(const Urg_driver& urg, const vector<long>& data, long
          time_stamp, double degree)
76      {
77          size_t data_n = data.size();
78          int input_index = urg.deg2index(degree);
79          int output_data = data[input_index];
80          return output_data;
81      }
82
83      //Check LiDAR status
84      int check_LiDAR(Urg_driver& urg, int argc, char *argv[])
85      {
86          qrk::Connection_information information(argc, argv);
87
88          if (!urg.open(information.device_or_ip_name(),
89              information.baudrate_or_port_number(),
90              information.connection_type()))
91          {
92              cout << "Urg_driver::open(): " << information.device_or_ip_name()
                    << ": " << urg.what() << endl;
93              return 1;
94          }
95          else
96          {
97              cout << "Urg_driver::open():" << information.device_or_ip_name()
                    << ":" << urg.what() << endl;
98          }
99          return 0;
100     }
101 }
102
```

Appendix E Code Samples for LiDAR and Camera Integration

This appendix includes samples from the C++ code that was implemented to manipulate the fusion between the detected bounding boxes using YOLO and the LiDAR measurements. Moreover, it handles the problem of the presence of multiple objects that could be overlapping. Full work has been represented in Chapter 4.

D:\OneDrive\PhD Studies\Thesis\Fusion_Code_Samples.cpp 1

```cpp
//.............Sensors Constants.....................................................
#define HP_HFOV 75.37
#define Logitech_HFOV 61.0
#define HP_Video_Width 640.0
#define Logitech_Video_Width 640.0
#define LiDAR_resolution 0.25
#define PI 3.14159265


//.............defining a new struct for overlapping boundig boxes...................
struct bbox_oL_t {
    unsigned int x, y, w, h;       // (x,y) - top-left corner, (w, h) - width & height of bounded box
    float prob;                    // confidence - probability that the object was found correctly
    unsigned int obj_id;           // class of object - from range [0, classes-1]
    unsigned int track_id;         // tracking id for video (0 - untracked, 1 - inf - tracked object)
    unsigned int frames_counter;   // counter of frames on which the object was detected
    std::vector<int> overlapped_pixels; //Pixels that have overlapped with other objects
    std::vector<int> overlapped_objects; //objects' number that overlapped with it
    std::vector<float> depths;       //depth points measured by LiDAR from point x to x+w
    std::vector<float> depth_wo_ol;   //depth points not overlapped (actual depth of this specific object)
};

//If we assumes HP wide vision HD camera HFOV is 88 deg
//But if 88 deg is the DFOV, and aspect ratio is (4:3 = 640:480), the HFOV is 75.37 deg


//Conerting pixels coordinates to angles to b mapped with LiDAR measurements
float Pixel_to_Angle(int H_pixel, int cam_num)
{
    float theta_degree, angle_radian, tan_theta_radian, angle_degree;
    int abs_dist_from_center, dist_from_center;
    if (cam_num == 0)
    {
        theta_degree = 0.5 * HP_HFOV;
        tan_theta_radian = tan(theta_degree * PI / 180.0);
        abs_dist_from_center = std::abs(H_pixel - (HP_Video_Width / 2.0)); // to eliminate direction
        dist_from_center = H_pixel - (HP_Video_Width / 2.0); //negative is before center towards left border
        angle_radian = atanf((2.0 * dist_from_center*tan_theta_radian) / HP_Video_Width);
    }
    else if (cam_num == 1)
    {
```

```
42          theta_degree = 0.5 * Logitech_HFOV;
43          tan_theta_radian = tan(theta_degree * PI / 180.0);
44          abs_dist_from_center = std::abs(H_pixel - (Logitech_Video_Width /
              2.0)); //to eliminate direction
45          dist_from_center = H_pixel - (Logitech_Video_Width / 2.0); //negative
              is before center towards left border
46          angle_radian = atanf((2.0 * dist_from_center*tan_theta_radian) /
              Logitech_Video_Width);
47      }
48      angle_degree = angle_radian * 180 / PI;
49      return angle_degree;
50  }
51
52  //Converting pixels to mirrored angles
53  float Pixel_to_Flipped_Angle(int H_pixel, int cam_num)
54  {
55      float theta_degree, angle_radian, tan_theta_radian, angle_degree;
56      int abs_dist_from_center, dist_from_center;
57      if (cam_num == 0)
58      {
59          theta_degree = 0.5 * HP_HFOV; //Theta = HFOV/2
60          tan_theta_radian = tan(theta_degree * PI / 180.0); //tan(Theta)
61          abs_dist_from_center = std::abs(H_pixel - (HP_Video_Width / 2.0)); //
              to eliminate direction
62          dist_from_center = H_pixel - (HP_Video_Width / 2.0); //negative is
              before center towards left border, x=HPixel-FrameCenter
63          angle_radian = atanf((2.0 * dist_from_center*tan_theta_radian) /
              HP_Video_Width); //Alpha = tan(inv) ((2 * x * tan(Theta))/
              Framewidth)
64      }
65      else if (cam_num == 1)
66      {
67          theta_degree = 0.5 * Logitech_HFOV;
68          tan_theta_radian = tan(theta_degree * PI / 180.0);
69          abs_dist_from_center = std::abs(H_pixel - (Logitech_Video_Width /
              2.0)); //to eliminate direction
70          dist_from_center = H_pixel - (Logitech_Video_Width / 2.0); //negative
              is before center towards left border
71          angle_radian = atanf((2.0 * dist_from_center*tan_theta_radian) /
              Logitech_Video_Width);
72      }
73      angle_degree = angle_radian * 180 / PI;
74      return angle_degree * -1.0;
75  }
76
77  //returns the range of angles between two points corresponding to LiDAR
      certain range
78  vector<float> Angles_to_Vector(float angle_1, float angle_2)
79  {
80      vector<float> angles;
81
82      if (angle_1 > 0)
83      {
84          angle_1 = (round(angle_1 / LiDAR_resolution)) * LiDAR_resolution;
85      }
86      else if (angle_1 == 0)
```

```
87      {
88          angle_1 = angle_1;
89      }
90      else
91      {
92          angle_1 = round(abs(angle_1 / LiDAR_resolution)) * LiDAR_resolution;
93          angle_1 = angle_1 * -1.0;
94      }
95
96      if (angle_2 > 0)
97      {
98          angle_2 = round(angle_2 / LiDAR_resolution) * LiDAR_resolution;
99      }
100     else if (angle_2 == 0)
101     {
102         angle_2 = angle_2;
103     }
104     else
105     {
106         angle_2 = round(abs(angle_2 / LiDAR_resolution)) * LiDAR_resolution;
107         angle_2 = angle_2 * -1.0;
108     }
109     int num_of_angles = abs((angle_1 - angle_2) * 4) + 1;
110     angles.push_back(angle_1);
111     for (int i = 1; i < num_of_angles; i++)
112     {
113         angles.push_back(angles[i - 1] - LiDAR_resolution);
114     }
115     return angles;
116 }
117
118 //writing the measured distances in a certain range in a file
119 void Write_to_file_with_distances(std::ofstream& outdata, string obj_name, int
       track_id, int left_p, int right_p, float left_a, float right_a,
      vector<float> distances)
120 {
121     if (!outdata)
122     {
123         cerr << "Error: file could not be opened" << endl;
124         exit(1);
125     }
126     outdata << obj_name << "," << track_id << ":" << endl;
127     outdata << "Left Pixel: " << "," << left_p << "," << "Right Pixel: " <<
        "," << right_p << endl;
128     outdata << "Left Angle: " << "," << left_a << "," << "Right Angle: " <<
        "," << right_a << endl;
129     outdata << "Distances: " << endl;
130     for (int i = 0; i < distances.size(); i++)
131     {
132         outdata << to_string(distances[i]) << ",";
133     }
134     outdata << endl;
135 }
136
137
138 //Converting the regular bounding boxes to the new struct bbox_OL
```

```
139 bbox_oL_t Convert_bboxes(bbox_t bbox)
140 {
141     //Convert from bbox_t to bbox_ol_t
142     bbox_oL_t bbox_ol;
143     bbox_ol.frames_counter = bbox.frames_counter;
144     bbox_ol.h = bbox.h;
145     bbox_ol.x = bbox.x;
146     bbox_ol.y = bbox.y;
147     bbox_ol.w = bbox.w;
148     bbox_ol.prob = bbox.prob;
149     bbox_ol.obj_id = bbox.obj_id;
150     bbox_ol.track_id = bbox.track_id;
151     return bbox_ol;
152 }
153
154 //Overlapping checking function
155 bbox_oL_t Overlapping_Check(std::vector<bbox_oL_t>& occupied_objects,
      bbox_oL_t new_object, int& overlap)
156 {
157     //Must be placed inside a loop, to loop over all detected objects
158     int left_pixel = new_object.x;
159     int right_pixel = left_pixel + new_object.w;
160     overlap = 0;
161     bbox_oL_t old_object;
162     if (occupied_objects.empty())
163     {
164         overlap = 0;
165         cout << "New Object:" << endl;
166     }
167     else
168     {
169         int vector_size = occupied_objects.size();
170         for (int i = 0; i < vector_size; i++)
171         {
172             old_object = occupied_objects[i];
173             if ((right_pixel > old_object.x) && left_pixel < (old_object.x +
                  old_object.w)) // partial OR full Overlap
174             {
175                 overlap = overlap + 1;
176                 cout << "Overlap with object # " << i << endl;
177                 new_object.overlapped_objects.push_back(i);
178                 occupied_objects[i].overlapped_objects.push_back(vector_size);
179                 //only add the start and end pixels overlapped
180                 if (right_pixel < (old_object.x + old_object.w))
181                 {
182                     new_object.overlapped_pixels.push_back(old_object.x);
183                     new_object.overlapped_pixels.push_back(right_pixel);
184                     occupied_objects[i].overlapped_pixels.push_back
                        (old_object.x);
185                     occupied_objects[i].overlapped_pixels.push_back
                        (right_pixel);
186                 }
187                 else
188                 {
189                     new_object.overlapped_pixels.push_back(left_pixel);
190                     new_object.overlapped_pixels.push_back(old_object.x);
```

```
191                     occupied_objects[i].overlapped_pixels.push_back
                        (left_pixel);
192                     occupied_objects[i].overlapped_pixels.push_back
                        (old_object.x);
193                 }
194             }
195             else //No Overlap .. do nothing
196             {
197                 //nothing as over iterations, it gets updated
198             }
199         }
200     }
201     occupied_objects.push_back(new_object); //Whether there is overlap or not,
          add the object in the occupied_objects
202     return new_object;
203 }
204
205
206 //Getting the Bounding boxes associated with 3D measurement
207 void Draw_3D_Boxes(Urg_driver& urg, int cam_number, cv::Mat mat_img,
      std::vector<bbox_t> result_vec, std::vector<std::string> obj_names, int
      current_det_fps = -1, int current_cap_fps = -1)
208 {
209     cout << "..............................................................."
          << endl;
210     cout << "New Frame:" << endl;
211     //For the writing in File//
212     ofstream outdata;
213     outdata.open("Test_Distances.csv", fstream::app);
214     if (!outdata)
215     {
216         cerr << "Error: file could not be opened" << endl;
217         exit(1);
218     }
219
220     std::vector<bbox_oL_t> occupied_objects;
221     bbox_oL_t bbox_ol;
222
223
224     int const colors[6][3] = { { 1,0,1 },{ 0,0,1 },{ 0,1,1 },{ 0,1,0 },
        { 1,1,0 },{ 1,0,0 } };
225     int Frame_Width;
226     if (cam_number == 0) Frame_Width = HP_Video_Width;
227     else if (cam_number == 1) Frame_Width = Logitech_Video_Width;
228
229     int overlap = 0;
230     std::vector<int> occupied_frame_pixels;
231
232     for (auto &i : result_vec)
233     {
234         cv::Scalar color = obj_id_to_color(i.obj_id);
235         cv::rectangle(mat_img, cv::Rect(i.x, i.y, i.w, i.h), color, 2);
236         int left_pixel = i.x;
237         int right_p = i.x + i.w;
238         int right_pixel = std::max(right_p, 5);
239         int center_x = i.x + floor(i.w / 2);
```

```
240         int center_y = i.y + floor(i.h / 2);
241         float left_angle = Pixel_to_Flipped_Angle(left_pixel, cam_number); //
              smaller number
242         float right_angle = Pixel_to_Flipped_Angle(right_pixel,
              cam_number); //bigger number
243
244
245         bbox_ol = Convert_bboxes(i);
246         bbox_ol = Overlapping_Check(occupied_objects, bbox_ol, overlap);
247
248         vector<float> object_distances;
249
250         vector<long> data;
251         long time_stamp = 0;
252         if (!urg.get_distance(data, &time_stamp))
253         {
254             cout << "Urg_driver::get_distance(): " << urg.what() << endl;
255         }
256         object_distances = angles_to_distances(urg, data, time_stamp,
              left_angle, right_angle);
257         object_distances = radii_to_linear(object_distances, left_angle,
              right_angle);
258         bbox_ol.depths = object_distances;
259         {
260             cout << endl << obj_names[i.obj_id] << ":" << endl;
261             cout << "Left pixel:" << left_pixel << ", Right Pixel: " <<
                  right_pixel << endl;
262             cout << "Left Angle:" << left_angle << ", Right Angle: " <<
                  right_angle << endl;
263
264             cout << "distances:" << endl;
265             for (int counter = 0; counter < object_distances.size(); counter+
                  +)
266             {
267                 cout << object_distances[counter] << ",";
268             }
269         }
270
271         if (obj_names.size() > i.obj_id)
272         {
273             std::string obj_name = obj_names[i.obj_id];
274             if (i.track_id > 0)
275                 obj_name += " - " + std::to_string(i.track_id);
276             cv::Size const text_size = getTextSize(obj_name,
                  cv::FONT_HERSHEY_COMPLEX_SMALL, 1.2, 2, 0);
277             int max_width = (text_size.width > i.w + 2) ? text_size.width :
                  (i.w + 2);
278             max_width = std::max(max_width, (int)i.w + 2);
279             //max_width = std::max(max_width, 283);
280             cv::rectangle(mat_img, cv::Point2f(std::max((int)i.x - 1, 0),
                  std::max((int)i.y - 35, 0)),
281                 cv::Point2f(std::min((int)i.x + max_width, mat_img.cols - 1),
                      std::min((int)i.y, mat_img.rows - 1)),
282                 color, CV_FILLED, 8, 0);
283             //putText(image, text string to be written, bottom-left corner,
                  font, font scale, color, thickness of text)
```

D:\OneDrive\PhD Studies\Thesis\Fusion_Code_Samples.cpp 7

```
284             putText(mat_img, obj_name, cv::Point2f(i.x, i.y - 16),
                  cv::FONT_HERSHEY_COMPLEX_SMALL, 1.2, cv::Scalar(0, 0, 0), 1.5);
285         }
286
287         Write_to_file_with_distances(outdata, obj_names[i.obj_id], i.track_id,
               left_pixel, right_pixel, left_angle, right_angle,
              object_distances);
288
289     }
290
291     if (current_det_fps >= 0 && current_cap_fps >= 0)
292     {
293         std::string fps_str = "FPS detection: " + std::to_string
              (current_det_fps) + "   FPS capture: " + std::to_string
              (current_cap_fps);
294         putText(mat_img, fps_str, cv::Point2f(10, 20),
              cv::FONT_HERSHEY_COMPLEX_SMALL, 1.2, cv::Scalar(50, 255, 0), 1.5);
295     }
296 }
297
298
```

Appendix F Auxiliary Material

Deraining Codes in the Literature

Algorithms	Paper link	Projects Link
HINET	Paper link	Code link
MPRNET	Paper link	Code link
MSPFN	Paper link	Code link
PreNet	Paper link	Code link

Object Detection Code

Algorithm	Code link	Pretrained model link
YOLOv3	Code link	Model
Faster R-CNN	Code link	Model
SSD-512	Code link	Model

Synthesized Datasets

Dataset	Download link
Test100	Google Drive link
Rain100H	Google Drive link
Rain100L	Google Drive link
Test2800	Google Drive link
Test1200	Google Drive link

Real-World Rainy Datasets

Dataset	Download link
RID	Google Drive link
RIS	Google Drive link