School of Computation, Information and Technology - Informatics
Technical University of Munich

Master's Thesis in Informatics

Real-Time and Multi-Modal 3D Object Detection on the Autonomous Driving Test Stretch Using Camera and LiDAR Sensors

Echtzeit- und Multimodale 3D-Objekterkennung auf der Teststrecke für Autonomes Fahren unter Verwendung von Kamera- und LiDAR-Sensoren

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Walter Zimmer, M.Sc.

Author Stefan Petrovski

Date February 15, 2023

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, February 15, 2023

(Stefan Petrovski)

Abstract

Autonomous vehicles require reliable and accurate perception of their surroundings to navigate safely and efficiently. While LiDAR and camera sensors are commonly used for object detection, they have different strengths and limitations. LiDAR sensors can provide high-resolution 3D point clouds, but are susceptible to noise and occlusion. Camera sensors can provide rich visual information, but are limited by the range, resolution and depth loss of their 2D images. In this thesis, we propose a real-time 3D multi-modal object detection system that combines the complementary strengths of multiple LiDAR and camera sensors to improve the accuracy and robustness of object detection. To accomplish that, we take a late-level fusion approach to fuse detections from two LiDAR sensors and two cameras, and lastly from both modalities. We arrive at three possible techniques, most notable of which being to match detections through data association and further extract the best performing feature from each modality, thus gaining the optimal combination of camera's and LiDAR's respective strengths. Through this approach we gain sufficient data redundancy, important for the system's reliability. We evaluate our model on the A9 infrastructure dataset and achieve a result of 69.45 `mAP` on south1 camera FoV test set.

Zusammenfassung

Autonome Fahrzeuge benötigen eine zuverlässige und genaue Wahrnehmung ihrer Umgebung, um sicher und effizient zu navigieren. Während LiDAR- und Kamerasensoren häufig zur Objekterkennung verwendet werden, haben sie unterschiedliche Stärken und Einschränkungen. LiDAR-Sensoren können hochauflösende 3D-Punktwolken liefern, sind jedoch anfällig für Rauschen und Verdeckungen. Kamerasensoren können reichhaltige visuelle Informationen liefern, sind jedoch durch die Reichweite, Auflösung und den Tiefenverlust ihrer 2D-Bilder begrenzt. In dieser Arbeit schlagen wir ein multimodales 3D-Echtzeit-Objekterkennungssystem vor, das die komplementären Stärken von mehreren LiDAR- und Kamerasensoren kombiniert, um die Genauigkeit und Robustheit der Objekterkennung zu verbessern. Um dies zu erreichen, verfolgen wir einen Late-Level-Fusion-Ansatz, um die Erkennungen von zwei LiDAR-Sensoren und zwei Kameras und schließlich von beiden Modalitäten zu fusionieren. Wir kommen zu drei möglichen Techniken, von denen die bemerkenswerteste darin besteht, Erkennungen durch Datenzuordnung abzugleichen und die leistungsstärkste Eigenschaft jedes Modalitäts weiter zu extrahieren, um so die optimale Kombination der jeweiligen Stärken von Kamera und LiDAR zu erhalten. Durch diesen Ansatz erreichen wir eine ausreichende Datenredundanz, die für die Zuverlässigkeit des Systems wichtig ist. Wir evaluieren unser Modell auf dem Infrastruktur-Datensatz A9 und erreichen ein Ergebnis von 69.45 `mAP` auf dem Kamera-FoV-Testset South1.

Contents

Chapter 1

Introduction

1.1 Autonomous vehicles

Autonomous vehicles (AVs) are revolutionizing the transportation industry by offering a safer and more convenient mobility solution for different applications - personal transportation, transportation of goods, and even public transportation [46]. They can reduce the number of accidents caused by human error, distractions or impaired driving. They can detect and respond to hazards on the road more quickly and accurately than humans, and can make decisions based on real-time data from sensors.

But both development and deployment of AVs require a significant amount of research and development, and is a complex task that involves many different technical issues. In order for the autonomous vehicles to operate effectively and, most importantly, safely, they need to have a reliable perception of their surroundings.

1.2 Perception and Object Detection

Perception, in the context of autonomous vehicles, refers to the process of understanding and interpreting the environment through an abundance of sensor data, extracted from cameras, LiDARs, radars, and various ultrasonic sensors [47] [30]. The goal of perception is to create a robust and accurate representation of the vehicle's environment, including the location and properties of surrounding objects - vehicles, pedestrians, and obstacles, as well as the road and traffic infrastructure.

Having a reliable perception of the environment is crucial for the autonomous vehicle to make safe and efficient decisions. Without accurate perception, it would not be able to navigate, avoid collisions, or follow traffic rules. Furthermore, perception is also important for the interaction with other road users, pedestrians included, and to coordinate with traffic infrastructure - traffic lights, road markings and signs.

One of the key challenges in the development of AVs is the reliable and precise perception of their environment and surroundings, which is a necessary step for safe and efficient navigation. And object detection is one of the most crucial aspects of the perception process, as it involves identifying and classifying objects from the environment.

There are various sensors that can be used for object detection, including radar, lidar and camera. Radar sensors can detect objects at extended ranges and even through occlusions, but unfortunately they provide limited resolution and are prone to display false positives (FP). Lidar sensors, on the other hand, can provide high-resolution 3D point clouds of the environment, but they are highly susceptible to noise and occlusion. Lastly, camera sensors

can provide rich visual information, but are limited by the range and resolution of their 2D images, in addition to the loss of depth information that the former 2 sensors can provide.

Real-time 3D multi-modal object detection is still a challenging and important task in the field of computer vision and robotics. Detecting and classifying objects in 3D space accurately and efficiently is crucial for a wide range of applications, not only limited to autonomous vehicles.

In order to ensure the safe and efficient operation of AVs across all the traffic participants, we need the ability to detect, classify and even track objects in the environment with a high enough accuracy and later on process these detections in real-time.

1.3 Motivation

One of the key components of autonomous driving systems is 3D object detection - the ability to perceive and detect objects (traffic participants) in real-time. This enables vehicles to make immediate informed decisions based on their surroundings. Object detection is a fundamental requirement for safe and efficient autonomous driving. 3D object detection is particularly important as it enables the system to estimate the position, dimensions, and orientation of objects in 3D space.

Existing approaches to 3D object detection typically rely on a combination of sensors, including radars, cameras and lidars. As cameras and LiDARs provide a high spatial resolution, accompanied with more accurate object classification and edge estimation, they (and their fusion) are chosen for the scope of this work. Although radars work at even longer ranges than the other two sensors, they have a really low resolution and do not provide any color information.

Moreover, the fusion of multi-modal (camera and LiDAR) sensor data can help further enhance the behavior of autonomous driving systems, by providing comprehensive, more accurate and reliable data. Apart from the obvious redundancy in data, accomplished by this approach, that improves the system's reliability, we are expecting even greater performance in different weather and lighting conditions, especially important during night scenes. Unfortunately, fusion in real-time poses significant computational and communication challenges that can affect the overall scalability of the system. Therefore a computationally inexpensive approach is required.

Most autonomous driving systems and datasets nowadays are vehicle-based focused - the sensors are placed on the vehicle itself [16] [4]. This approach is advantageous regarding some aspects, including immediate environment observation, but also carries a lot of limitations. The field-of-view of these sensors is limited - objects can be obstructed or missed or they can simply be out of the sensing range. Objects can also be occluded by other objects, creating blind spots, making it particularly difficult for vehicle-placed sensors to detect them. Other factors to consider are the limited accuracy and calibration - vehicle-based sensors often do not have the same level of accuracy as stationary sensors and need to calibrated more regularly, which can be a time-consuming process that takes the vehicle out of commission.

Roadside sensors, on the other hand, can provide several advantages for object detection in the context of autonomous driving systems [69] [3] [71] [11]. First, they can cover a much larger area than vehicle-based sensors, which are limited to the immediate vicinity of the vehicle. They can be further placed at strategic locations to cover blind spots at intersections and other areas where the on-board sensors may not be that effective. This can improve situational awareness and enable the detection of objects that may be outside the vehicle's field-of-view and also cover blind spots. Second, roadside sensors can be positioned in a more stable and controlled environment, reducing the impact of external factors such as weather,

road conditions, and other vehicles. This can lead to more consistent and accurate data.

Finally, roadside sensors can provide additional information, such as the speed and direction of the objects, which can help in object tracking and prediction. Accompanied by being even more cost-effective than their vehicle-based counterparts, infrastructure-side sensors are extremely important for the improvement of the safety and performance of autonomous driving systems, helping them understand and better respond to their environment.

1.4 Outline and Contribution

In this work, we will explore the state-of-the-art techniques for real-time 3D multi-modal object detection. We will begin by reviewing the relevant literature, including both traditional and deep learning-based approaches. We will then present our own contribution to the field - a late-fusion real-time 3D multi-modal roadside object detection system that combines the complementary strengths of the most widely used sensors on the market of AVs - LiDAR and camera sensors - to improve the accuracy and robustness of object detection. Our system uses a modified variation of the Hungarian algorithm [27] for data association in its core - the *modified Junker-Volgenant* algorithm [12] - to fuse the detections from the two modalities, and is designed to run in real-time on Providentia++ infrastructure live system that uses roadside sensors. We evaluate the performance of our approach on an infrastructure based autonomous driving dataset of real-world driving scenes, the A9 infrastructure dataset [11] (Figure 1.1 and Chapter 4.1), and discuss the challenges and limitations of the proposed system.

Figure 1.1: The Providentia++ project and test bed used for acquiring the datasets. [49]

Chapter 2

Background

For the scope of this master thesis, we will be investigating the different sensors that play a part in an autonomous driving object detection system, while focusing on the sensors that are of particular interest to our approach - Camera and LiDAR. Afterwards, we will examine the different types of fusion techniques, most widely used in practice today. At the end, we will mention important datasets in the autonomous driving domain, as well as elaborate on their respective types, in terms of sensors - camera, LiDAR, radar, etc., as well as perspective - vehicle, infrastructure (roadside) and cooperative (vehicle + infrastructure).

2.1 Sensors

2.1.1 Radar

Radio Detection and Ranging (Radar) sensors are widely used in autonomous driving systems for object detection and perception [35] [74]. These sensors emit radio waves and measure the reflection (return) of these waves from the objects in the environment. The time difference between the emission and the reception of those reflected waves is used to estimate the distance of the objects, while the phase shift of the reflected waves is used to estimate the velocity of the objects.

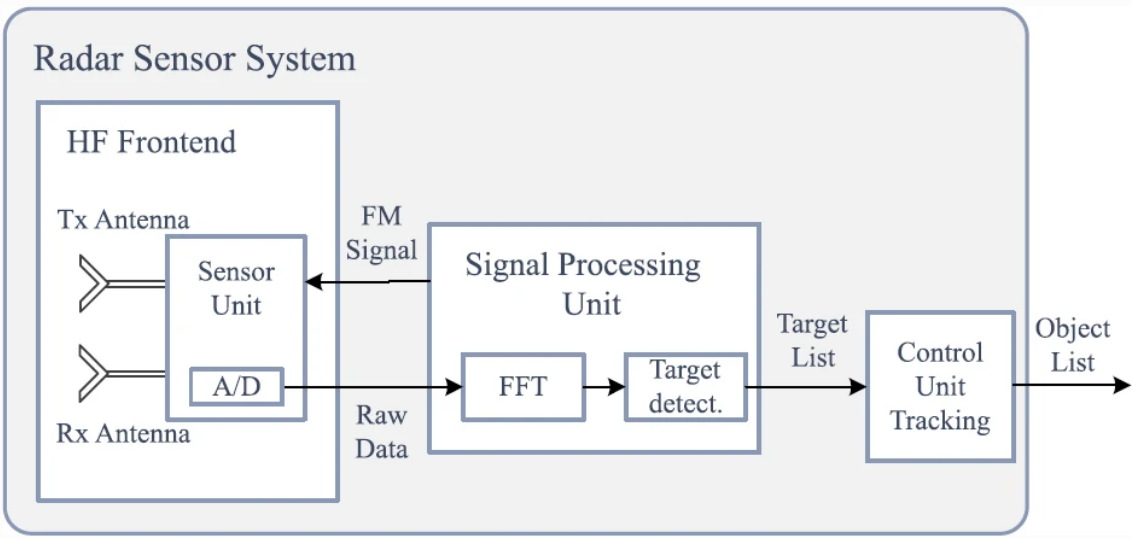

Figure 2.1: Radar sensor system. [35]

Radar sensors main advantage over other sensors used in AVs is the detection of objects at long ranges and in low-lighting conditions, such as fog, rain, or during the night. This makes them particularly useful for detecting distant vehicles, pedestrians, and other objects regardless of occlusions. Additionally, radar sensors are relatively unaffected by the effects of glare and reflection, which can prove to be major issues for both camera and LiDAR sensors.

Radar sensors are used to detect and track objects in the environment, estimate velocities and positions, even in some cases classification of traffic participants - vehicles, pedestrians, bicycles, etc. These capabilities play an important role for enabling autonomous vehicles to safely navigate through their environment and avoid collisions with other objects.

Radar sensors can also be used to complement other sensors, such as cameras and LiDAR, to provide a more complete picture of the vehicle's surroundings. For example, radar data can be used to provide additional information of an object - velocity or distance from sensor and in more rare cases to confirm or reject detections made by other sensors.

2.1.2 Ultrasonic Sensors

Ultrasonic sensors (sonar sensors) are commonly used in autonomous driving systems for object detection and perception. They work by emitting high-frequency sound waves and measuring the time taken for said waves to bounce back, allowing for the calculation of the distance to objects [74] [41] [19] [29]. Ultrasonic sensors have a relatively short range, typically less than 10 meters, but are able to detect objects in close proximity to the vehicle.

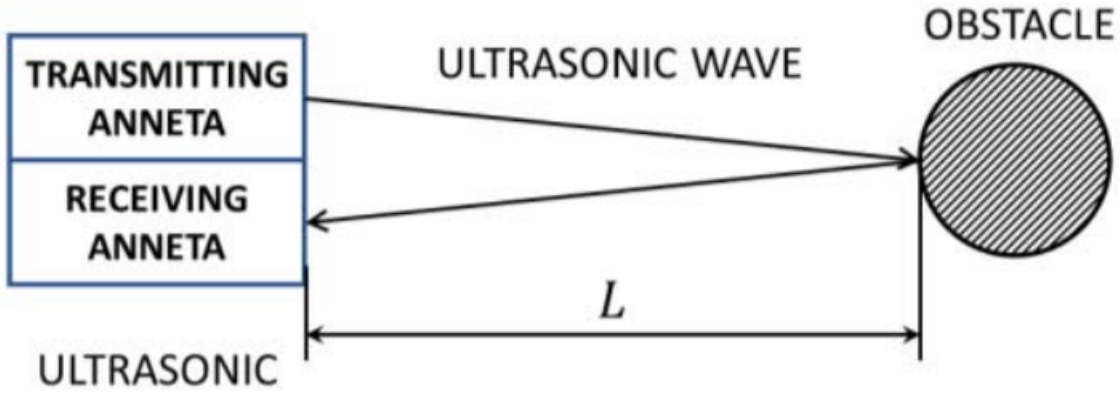

Figure 2.2: Ultrasonic radar system. [74]

One of the advantages of ultrasonic sensors is their low cost and ease of integration into a vehicle. They are also relatively robust to environmental conditions, such as dust and rain, which can affect the performance of other sensors. Ultrasonic sensors are commonly used in parking assist systems to detect obstacles from the environment, their distance [29], and enable automated parking maneuvers and even road speed limit recognition [74].

However, ultrasonic sensors have some limitations. They are not able to detect objects at long distances, and their accuracy can be affected by environmental factors such as wind, which can cause the sound waves to be scattered. Additionally, ultrasonic sensors are not able to perceive objects with low reflectivity, such as foam batting or other that absorb sound waves, which can result in false negatives [25].

In recent years, research has been focused on improving the performance of ultrasonic sensors for autonomous driving by developing advanced signal processing techniques, such as beamforming and Doppler processing, to increase the range and accuracy of the sensor. Additionally, the use of multiple ultrasonic sensors in a sensor array configuration has been

proposed to improve the field of view and robustness to environmental conditions.

In conclusion, ultrasonic sensors are a cost-effective and robust solution for object detection and perception in autonomous driving systems, with the main advantage of working well in close proximity to the vehicle. However, they have some limitations such as range and environmental factors. Research is ongoing to improve the ultrasonic sensor technology to overcome these limitations in the future.

2.1.3 LiDAR

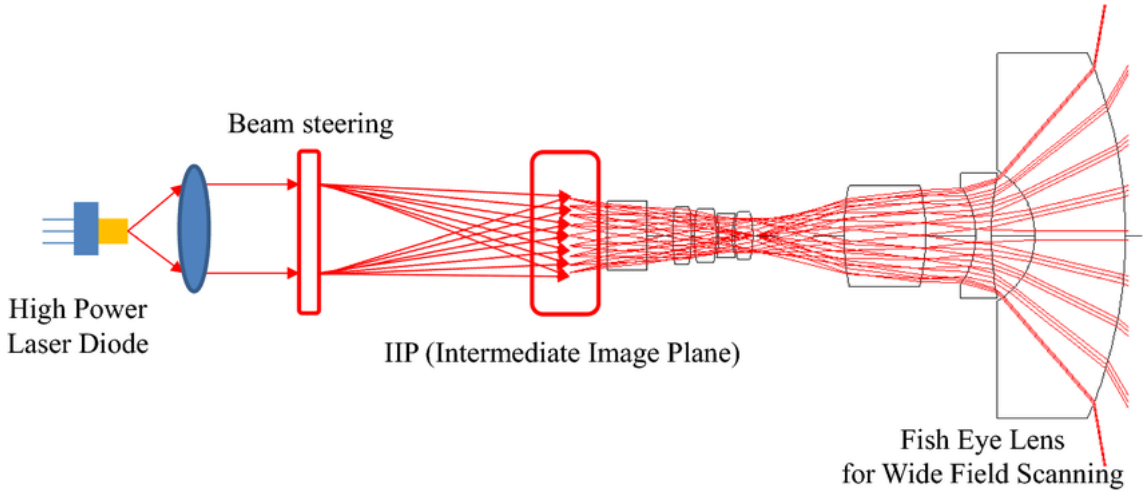

Figure 2.3: LiDAR sensing system schematic. [7]

LiDAR, or Light Detection and Ranging, is a sensor that uses laser beams to measure distance between a sensor and objects in its environment [37]. In autonomous driving systems, LiDAR is commonly used for object detection and perception.

One of the key advantages of LiDAR is its ability to provide high-precision depth estimation, or range measurement. Unlike cameras, which rely on visual cues to infer depth, LiDAR uses the time-of-flight of laser pulses to directly measure the distance to objects [36] [17]. This allows for accurate 3D reconstruction of the environment, which is crucial for detecting and tracking objects such as cars, pedestrians, and other traffic participants.

LiDARs are also capable of providing a high density of point measurements, which allows for a more detailed representation of the environment. This high resolution point cloud data can be used to extract features such as edges, corners, and surfaces, which can be used to detect and track objects. Additionally, LiDARs are able to operate in a wide range of lighting conditions, including complete darkness, making it an ideal sensor for autonomous driving.

However, LiDARs also have some limitations. For example, LiDARs are, in gerenal, more expensive than other sensors, used for perception tasks, and have a limited field-of-view (FoV) [33]. Additionally, LiDARs are sensitive to reflections and can be affected by atmospheric scattering [37], which can cause errors in range measurements. The quality of the detection under rain, snow and sometimes even fog, especially when the conditions are extraordinary in amount, becomes severely degraded, especially regarding range due to the absorption and scattering events induced by water droplets. Despite these limitations, LiDARs are widely used in autonomous driving systems due to their ability to provide high-precision depth estimation and 3D reconstruction.

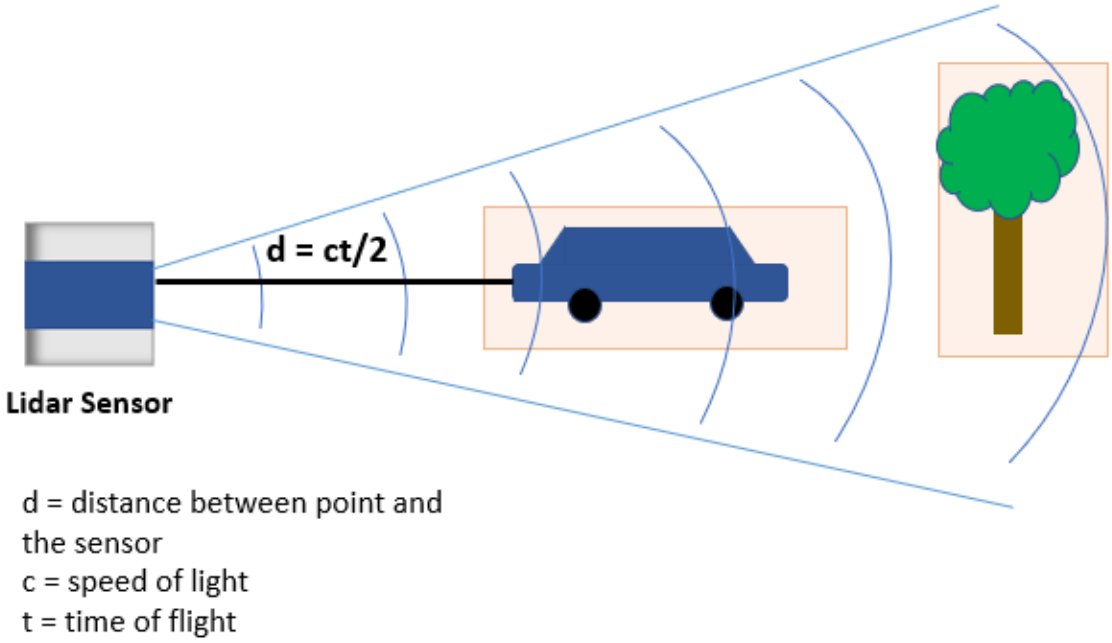

Figure 2.4: LiDAR distance measurement system. [36]

2.1.4 Camera

Cameras are widely used in autonomous driving systems for object detection and perception due to their ability to provide high-resolution images and their relatively low cost compared to other sensors such as LiDAR [67].

One of the main advantages of using cameras for object detection is their ability to capture rich visual information, including color, texture, and shape [13]. This information is useful for identifying and classifying objects in the scene, such as vehicles, pedestrians, and traffic signs. Additionally, cameras can provide a wide field of view, which can be beneficial for detecting objects at a distance and in peripheral areas of the scene.

A variety of computer vision techniques have been developed to perform object detection and perception using cameras in autonomous driving systems and many popular approaches are based on convolutional neural networks (CNNs), which have been shown to be highly effective at recognizing objects in images. CNNs can be trained on large datasets of labeled images, such as the KITTI dataset [16], to learn to detect objects in real-world scenes. Other techniques include using stereo cameras to estimate depth, and using multiple cameras to provide a more complete view of the scene [13].

However, camera-based object detection and perception also has some limitations. One major drawback is that cameras rely on visible light, which means that their performance can be affected by lighting conditions such as glare, shadows, and reflections [14] [26]. Cameras also have limited range and can be affected by weather conditions such as rain or fog. Additionally, cameras have limited ability to detect objects that are close to the vehicle or that have low reflectivity.

To overcome these limitations, cameras are often used in conjunction with other sensors such as LiDAR.

Table 2.1: Comparison of different sensor types [38].

Metric	Radar	Camera	LiDAR	Camera + LiDAR
Range	✓✓✓	✓✓	✓✓	✓✓
Distance Estimation	✓✓✓	✓✓	✓✓✓	✓✓✓
Object Localization	✓✓	✓	✓✓✓	✓✓✓
Object Classification	✓	✓✓✓	✓✓	✓✓✓
Edge Estimation	✓	✓✓✓	✓✓✓	✓✓✓
Lane Tracking	✓	✓✓✓	✓	✓✓✓
Night Tracking	✓✓✓	✓✓	✓✓✓	✓✓✓
Hazardous weather	✓✓✓	✓	✓✓	✓✓
Color Information	✓	✓✓✓	✓	✓✓✓
Spatial Resolution	✓	✓✓✓	✓✓	✓✓✓

2.1.5 Multi-modal

Multi-modal perception for autonomous vehicles (AVs) refers to the use of multiple sensor modalities to perceive and understand the environment around [15]. It greatly improves the robustness and accuracy of the perception system, as each different sensor modality has its own strengths and weaknesses. For example, LiDARs provide high-precision depth estimation, while cameras offer high spatial resolution and color information. Radars provide robustness against lighting conditions and can detect objects at vast distances. Ultrasonic sensors are used for close-range perception and can be used for detecting obstacles in a vehicle's immediate vicinity.

Multi-modal perception systems can use data fusion techniques to combine information from different sensors to improve the overall perception performance [15]. There are two main types of data fusion: early-level fusion and late-level fusion. Early-level fusion combines all of the raw sensor data before it is sent for processing, while late-level fusion combines the already pre-processed sensor data. Both approaches have their own advantages and disadvantages, such as the difference in computational cost, the complexity of data association required as well as the overall robustness of the system.

Examples of multi-modal perception systems for AVs include combining LiDAR and radar data for object detection and tracking, or combining camera and ultrasonic sensor data for parking assistance.

2.2 Types of Fusion

2.2.1 Early-Level Fusion

Early-level detection fusion for autonomous driving refers to the process of combining information from multiple sensors at an early stage of the perception pipeline, typically before the detection of objects [15]. This approach allows for the integration of sensor data with complementary characteristics, such as LiDAR and radar, to improve the robustness and accuracy of object detection. The main benefit of early-level fusion is that it can leverage the strengths of each sensor to improve the overall performance of the perception system.

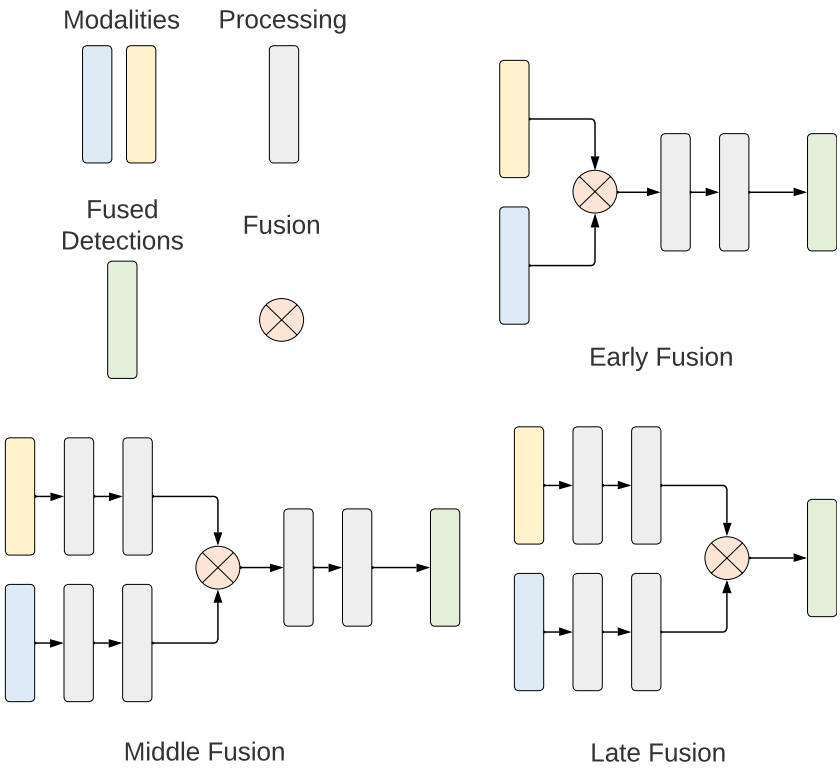

Figure 2.5: Early, middle and late fusion approaches. [15]

2.2.2 Middle-Level Fusion

Middle-level fusion combines the feature representations from different sensors at intermediate processing layers of the object detection framework [15]. This allows for a more sophisticated and flexible fusion of the information from different sensors. By combining the features at an intermediate layer, a network can learn to be able to weight the information from each sensor based on its relative importance for the task of object detection.

2.2.3 Late-Level Fusion

Late-level detection fusion for autonomous driving, on the other hand, refers to the process of combining information from multiple sensors at a later stage of the perception pipeline, typically after the detection of objects [15]. This approach allows for the integration of sensor data with similar characteristics, such as multiple cameras, to improve the accuracy and consistency of object detection. The main benefit of late-level fusion is that it can leverage the redundancy of the sensor data to improve the robustness of the perception system [71]. However, it can also increase the complexity of the system and require significant computational resources.

2.3 Datasets

Recent breakthroughs [21] in the deep learning domain have enabled machines to perform a wide range of tasks that were previously only possible for humans. This is a crucial step towards the development of self-driving vehicles that can independently sense and autonomously navigate through challenging road conditions and hazards. Furthermore, it creates opportunities for future smart city solutions and promotes the transition to safe and efficient transportation. Unfortunately, the development of these AI based mobility solutions is heavily dependent on the availability of big data. The training of advanced AI models necessitates a vast quantity of labeled data, as the models' generalization ability increases with the diversity found in the training data [11].

There is a variety of different autonomous driving datasets readily available. They can be broadly categorized into two parts - the sensors they use - camera, LiDAR, radar and their perspective - vehicle, infrastructure, cooperative (vehicle + infrastructure).

2.3.1 Sensor Type

On the basis of sensors, datasets can be broadly categorized into several categories:

LiDAR point clouds

These datasets typically consist of LiDAR scans collected from a vehicle, and contain information about the location and intensity of reflected laser beams. Examples of LiDAR datasets include the KITTI dataset [16] and the Waymo Open Dataset [61] [39].

Camera images

These datasets typically consist of images and corresponding labels collected from cameras mounted on a vehicle. The labels often include information about the location and properties of objects in the image such as pedestrians, vehicles, and road signs. Examples of camera datasets include the Cityscapes dataset [10] [9] and the KITTI dataset [16].

Radar data

These datasets typically consist of radar scans collected from a vehicle, and include information about the location and velocity of objects in the environment. Examples of radar datasets include the nuScenes radar dataset [4].

Multi-modal datasets

These datasets include more than one modality such as LiDAR, radar, cameras, GPS and IMU. These datasets are more challenging to collect and label, they cover a wide range of scenarios, and they can be used to train and test the performance of multi-modal perception and prediction systems. Examples of multi-modal datasets include the Waymo Open Dataset [39] and the ApolloScape dataset [20].

2.3.2 Perspective

Based on the relative perspective, three main approaches have emerged in the field of autonomous driving datasets: vehicle-based, infrastructure-based and cooperative-based datasets. In the next sections we will be focusing on the first two.

Vehicle-based datasets are collected from sensors mounted on the vehicle (roof, front and rear bumpers), such as cameras and LiDAR sensors. These datasets typically provide detailed information about the vehicle's immediate environment, including its surroundings, but may have limited coverage, severe occlusion due to other traffic participants and lack information about the broader context of the scene.

The data captured by vehicle-based sensors is highly relevant for tasks such as object detection and tracking, lane keeping, and traffic sign recognition.

Infrastructure-based datasets, on the other hand, are collected from sensors mounted on roadside infrastructure. These datasets usually provide a broader view of the scene and may include information about the road network, traffic signs, and other static objects. However, they may suffer from limited information about the vehicle's immediate environment. The use of infrastructure-based sensors also allows for the collection of data in a variety of locations and over extended periods of time.

The main advantage of infrastructure-based datasets is that they provide a comprehensive view of the entire environment and thus can cover the blind spots for two extra advantages over car views: a long-range global perspective to extend the vehicles field-of-view spatially and temporally, as well as, provide a global trajectory prediction (for safety purposes) [69].

Vehicle based

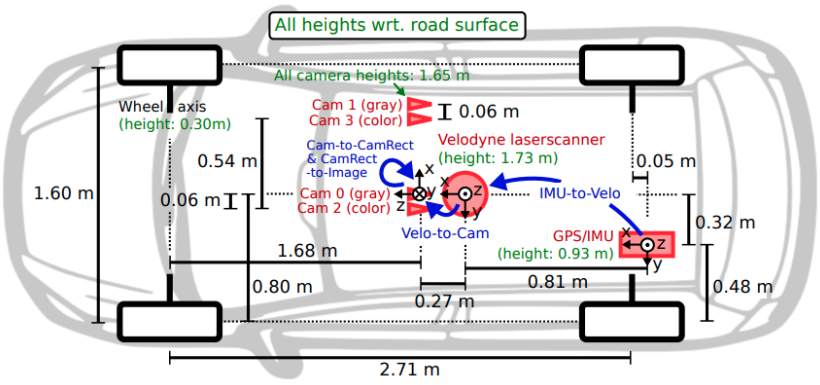

Figure 2.6: Sensor setup on the KITTI recording platform (vehicle). [16]

The KITTI dataset [16], also known as Karlsruhe Institute of Technology and Toyota Technological Institute (of Chicago) dataset, is a widely used vehicle based benchmark dataset for evaluating the performance of various computer vision and machine learning models, in the context of autonomous driving. The dataset was created by the Karlsruhe Institute of Technology and Toyota Technological Institute, and it is publicly available for research purposes.

The KITTI dataset includes a variety of data modalities, such as stereo images, LiDARs, and high-precision GPS/IMU measurements. The dataset also includes a diverse set of scenarios, including urban, suburban, and rural areas, as well as different lighting conditions and weather. Additionally, there is a suite of benchmarks such as stereo, optical flow, visual odometry, and 3D object detection, which allows researchers to evaluate the performance of their models in a wide range of applications.

The dataset is collected from a car equipped with two color and grayscale stereo cameras, a Velodyne HDL-64E laser scanner and a high-precision GPS/IMU unit [16]. The dataset includes synchronized stereo images and LiDAR point clouds, along with ground truth information such as 3D bounding boxes for objects and 6-DoF poses of vehicles [16]. It is one of the most widely used datasets for autonomous driving research.

The nuScenes dataset, developed by the autonomous vehicle company nuTonomy, is a large-scale, multi-modal dataset for autonomous driving research [4]. It is designed to provide an exhaustive set of sensor data and annotations to support the development and evaluation of diverse models for various autonomous driving tasks, such as object detection and tracking, semantic and instance segmentation, as well as, scene understanding.

The dataset includes data from 1,000 scenes, each of which is 20 seconds in duration and covers a total area of approximately 20 km^2 [4]. Each scene includes data from various sensor modalities, such as LiDAR, radar, camera, and GPS/IMU measurements, which are synchronized at a 10 Hz rate [4]. Additionally, the dataset includes high-definition maps of the area, which provide detailed information about the road network, traffic signs, and other static objects.

The annotations in the nuScenes dataset include 3D bounding boxes for objects, 6-DoF poses for vehicles, and semantic and instance segmentation for scenes. These annotations provide a lot of information for various autonomous driving tasks and allow for the development and evaluation of a wide range of models.

The dataset covers a diverse set of scenarios, including urban, suburban, and highway driving, and various weather conditions, lighting, and road types [4]. Therefore, the nuScenes dataset has been widely used in various studies and evaluations.

Waymo Open Dataset is a large-scale dataset for autonomous driving released by Waymo, Alphabet's self-driving car division [39]. The dataset is designed to support research in areas such as perception and multi-modal sensor fusion.It consists of high-resolution sensor data collected from Waymo self-driving vehicles, including LiDAR point clouds, high-resolution camera images, and sensor calibration data. The dataset contains a little over 10 million 3D annotated frames, representing a diverse set of scenarios and environmental conditions [39].

Waymo Open Dataset includes annotations for a variety of object classes, including vehicles, pedestrians and bycicles [39]. The annotations include 3D bounding boxes, attributes, and instance-level segmentation masks. Additionally, the dataset includes detailed calibration data for the LiDAR and camera sensors, as well as information on the vehicle's motion and ego position.

The Waymo Open Dataset is designed to provide a challenging and diverse set of data for the research community to work on, and has already been used to advance the state of the art in autonomous driving perception and motion prediction. It provides a valuable resource for researchers and developers working in this field, and its release is expected to drive further advances and innovations in autonomous driving technology.

Infrastructure-based Datasets

Rope3D: The Roadside Perception dataset is a large-scale, high-diversity, publicly available dataset for autonomous driving research [69]. It was created by researchers at the China University of Mining and Technology. This dataset consists of 50k images and over 1.5 million 3D bounding box labeled objects. Its frames span across 26 scenes a variety of lighting conditions (night, dawn) and weather conditions (rain) [69]. This dataset's only downside is its single-modality - it only consists of monocular data (images) and 3D monocular labels. There are no LiDAR point clouds and LiDAR labels.

LUMPI (Leibniz University Multi-Perspective Intersection) dataset is a multi-modal and multi-view (infrastructure + vehicle) dataset for autonomous driving and object detection

[3]. The dataset was recorded at an intersection in Hanover, Germany. This intersection has a dense traffic, consisting of cars, vans, busses, trucks, motorcycles and even pedestrians and bicycles. LUMPI contains 200k images and 90k point clouds recorded from two camera and five LiDAR views, respectively [3].

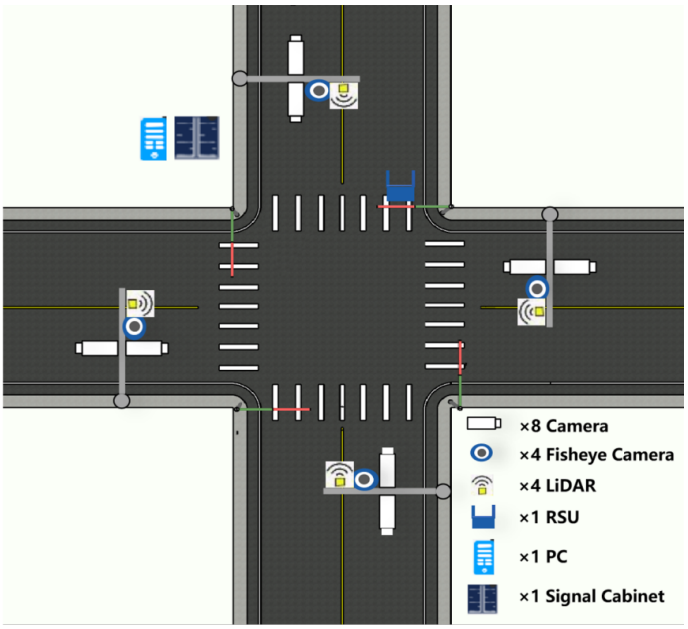

Figure 2.7: Sensor setup on one of the intersections used for the collection of the DAIR-V2X-I dataset [71]

DAIR-V2X (Dataset for Automated Intersection and Road-to-X) dataset is the first large-scale, multi-modal and multi-view dataset [71]. It was created by researchers at the Institute for AI Industry Research (AIR), Tsinghua University and the University of Chinese Academy of Science and is designed to support the development of advanced driver assistance systems (ADAS) and autonomous driving systems. It contains 71,254 LiDAR frames and 71,254 Camera frames captured in intersection scenes with a multi-sensor equipped vehicle that passes through multiple multi-sensor equipped intersections [71]. 40% of the frames are captured from infrastructure sensors and 60% of the frames are captured from vehicle sensors. The dataset covers 38 km^2 of driving regions with differing weather and lighting conditions[71].

DAIR-V2X-I is a part of the DAIR-V2X Dataset, that consists entirely of high-resolution roadside sensor data collected from various road scenarios, including intersections, roundabouts, and highways. The data includes LiDAR point clouds and high-resolution camera images - 10k in total per modality, with 493k 3D bounding boxes labeled and spanned throughout 10 different classes - vehicles, pedestrians, bicycles and others.

This dataset's focus on intersections and road-to-X scenarios makes it particularly well-suited for the development of advanced driver assistance systems and autonomous driving systems that need to handle complex road scenarios and make real-time decisions based on sensor data.

Chapter 3

Related Work

For the related work part we are firstly going to go through the related methods and models used for the back-end of our multi-modal 3D object detector - both monocular and LiDAR. Afterwards, we are going to discuss and compare the current and state-of-the-art fusion methods for multi-modal 3D object detection.

3.1 Monocular Based Methods

3.1.1 YOLOv7

YOLOv7 is one of the most efficient and accurate real-time object detectors, while only trained on the MS COCO dataset, without any other pre-trained weights [65].

Its architecture consists of a couple of reforms, including a new computational block backbone - E-ELAN (Extended Efficient Layer Aggregation Network), which uses different methods for continuous learning enhancement.

In comparison to a previous version from the YOLO family - YOLOv4, YOLOv7 achieves 1.5% higher `mAP`, while relying on 75% less parameters and being 36% less computationally expensive [65].

YOLOv7 has recently been outperformed by its successor - YOLOv8, developed and maintained by Ultralytics [70].

3.2 LiDAR Based Methods

3.2.1 PointPillars

PointPillars is a 3D object detection method that only relies on 2D convolutional layers (Figure 3.1) [28].

It predicts 3D bounding boxes using an encoder that learns features from vertical columns from the point cloud, also called pillars [28]. As the model operates on pillars, all of the operations can be formulated as 2D convolutions, which are really computationally inexpensive, especially on GPUs [28].

PointPillars achieved state-of-the-art performance, back in 2019, on the KITTI dataset, beating even multi-modal approaches that utilize both camera and LiDAR. In our work we are using PointPillars due to its fast inference time of approximately 42 FPS or 62 FPS with the use of TensorRT.

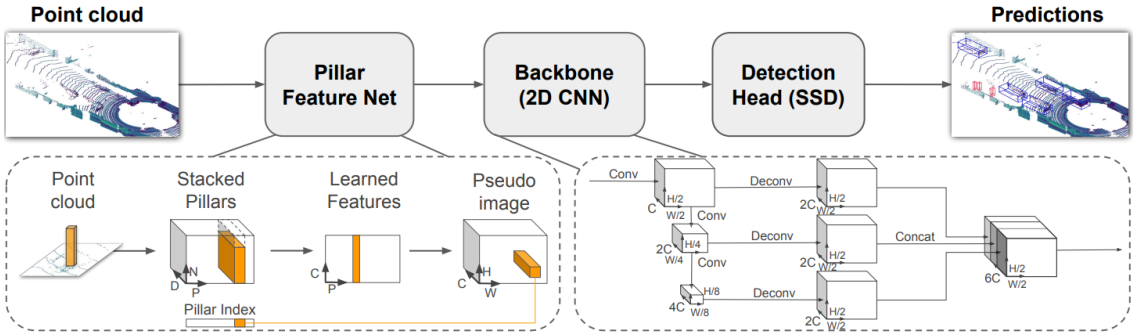

Figure 3.1: PointPillars network structure [28]

3.3 Multi-Modal Based Methods

3.3.1 DeepFusion

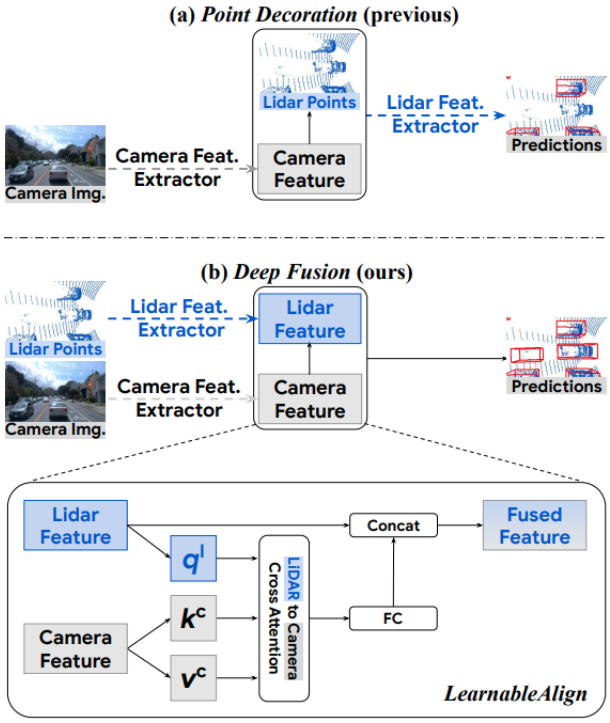

Figure 3.2: Comparison between Point Decoration [64] [66] (a) and DeepFusion (b) pipelines [31]

DeepFusion, developed by researchers from John Hopkins University and Google, is a family of generic multi-modal 3D object detection models (Figure 3.2) [31]. It fuses both modalities on a deep feature level, whereas previous methods such as PointPainting decorate LiDAR points with camera features at the early level.

To address the issue of correspondence between LiDAR and camera features, two simple, yet effective techniques are proposed: InverseAug, which inverses geometric-related data augmentations and uses sensor parameters to associate them, and LearnableAlign, which relies on cross-attention to learn correlations between the different modality features [31].

DeepFusion achieved state-of-the-art performance on the Waymo Open Dataset [39].

3.3.2 TransFusion

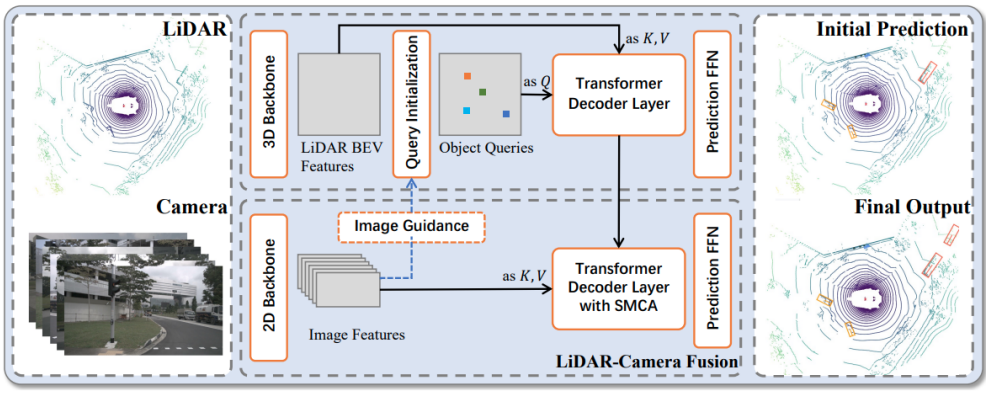

Figure 3.3: TransFusion multi-modal model pipeline [1]

TransFusion is another multi-modal, sensor fusion, object detection model that was considered state-of-the-art, scoring first on the nuScenes dataset (Figure 3.3) [1].

It is a powerful model for combining both modalities - LiDAR and camera data. It includes convolutional networks and a detection head based on a transformer decoder [1]. The decoder's first layer generates initial bounding boxes from the LiDAR point cloud. Then its second layer blends the received object with relevant image features, considering both location and context. The transformer's attention mechanism lets the model choose what information to use from the image, resulting in a reliable fusion strategy that performs well even under certain calibration errors [1].

3.3.3 BEVFusion

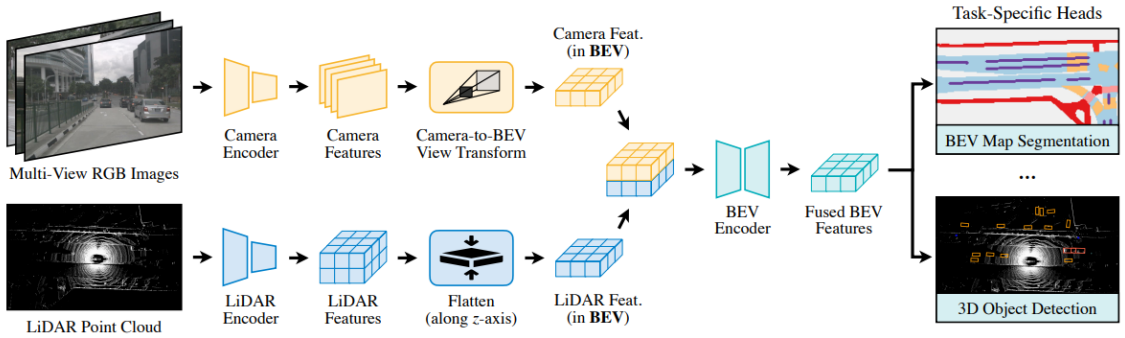

Figure 3.4: BEVFusion multi-task, multi-modal framework pipeline [34]

BEVFusion, developed by researchers from MIT, OmniML and Shanghai Jiao Tong University, is an efficient multi-task and multi-sensor fusion object detection model, which combines multi-modal features in the shared bird's-eye view (BEV) space, in order to maintain the initial semantic and geometric data (Figure 3.4) [34].

Initially the model applies specific encoders to extract features from LiDAR and camera data. In the next step, we apply an optimized blending of features in the BEV representation with the help of precomputation and interval reduction [34]. Then the so called BEV encoder, made up of convolutional layers, is applied to the multi-modal features, encoding them and finally, a 3D Object detection head is added.

BEVFusion, when released, lead the state-of-the-art on nuScenes and currently scores second on the leaderboard (3rd with an inclusion of an unknown model) [45].

3.3.4 DeepInteraction

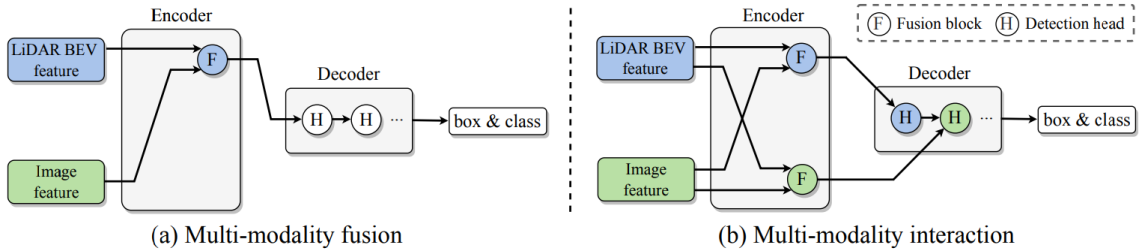

Figure 3.5: Comparison between existing multi-modality fusion (a) and multi-modality interaction based DeepInteraction (b) [68]

DeepInteraction is another state-of-the-art multi-modal framework that is currently occupying the first place on the nuScenes 3D object detection leaderboard (Figure 3.5) [68] [45]. It is defined as the first "modality interaction" strategy [68]. The model maintains both modality specific features throughout the whole process with a so-called "representational interaction" in the encoder and a "predictive interaction" in the decoder part (Figure ....) [68].

The strategy behind the approach is specifically constructed to resolve the limitations of the current one-sided fusion approaches, which do not fully capitalize on the image features.

3.3.5 RoarNet

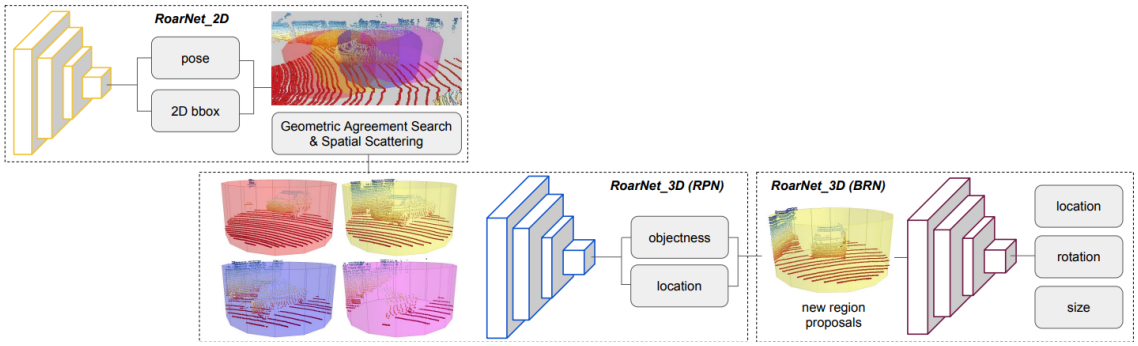

Figure 3.6: RoarNet pipeline [58]

RoarNet is an older late-fusion 3D object detection framework from 2019 (Figure 3.6), based on a two stage object detection framework with a backbone network PointNet [50] [58].

Using geometric agreement search, the model suggests region proposals (feasable candidates) in 3D space, from a monocular input image. Thereafter, it adopts a two-step object detection strategy in order to refine the search space in the 3D point clouds. Finally, the model calculates the IoU (Interection over Union) between the resultung 2D and 3D detections, in order to form the final 3D bounding box output.

RoarNet achieved in 2019 state-of-the-art performance on the KITTI dataset and showed robustness against synchronization errors.

3.4 Data Association Techniques

3.4.1 Non Maximum Suppression

Non Maximum Suppression (NMS) is a technique used in various computer vision tasks, including object detection in images and videos [43].

The idea behind NMS is to reduce the number of overlapping bounding boxes that correspond to a same exact object in an image by keeping only the bounding box with maximum probability score. It is applied after object detection models have generated a multitude of bounding boxes for the same object, with different scores, that show a probability that these boxes contain the object in question.

Non Maximum Suppression works by selecting the bounding box with the highest score, removing it from the list of candidate bounding boxes, and then "suppressing" all other bounding boxes that have a significant overlap with the selected box. This overlap between two bounding boxes is typically measured using the Intersection Over Union (IoU) metric [53] [44].

This process is repeated numerous times until all candidate bounding boxes have been handled. At the end, only bounding boxes with high scores and insignificant overlap with other bounding boxes are kept as final detections. This helps to reduce false positives, improve the overall accuracy of object detection algorithms and reduce the number of duplicated detections.

3.4.2 Nearest Neighbor and K-Means

Nearest Neighbor (NN) is a simple and widely used data association technique in computer vision [6]. The data association problem is reduced to finding a nearest neighbor of each detection from one modality to a detection from another modality. NN is typically defined based on a distance metric that captures the similarity or closeness between detections, such as the Euclidean distance.

K-Means is a popular modification of Nearest Neighbor [6]. It divides detections into k different clusters, where k is a predefined number of clusters. The detections within the same cluster are assumed to belong to the same object.

To perform K-Means clustering, the algorithm first randomly selects k initial cluster centers, which can be chosen from a set of detections from one modality. The algorithm then assigns each detection to the nearest cluster center, based on a distance metric such as Euclidean distance.

After all detections have been assigned to clusters, cluster centers are updated to a mean of the assigned detections. The algorithm then repeats assignment and update steps until convergence, which is typically achieved when cluster centers no longer change significantly.

3.4.3 Kalman Filter

Kalman filter is a recursive algorithm that models the motion of an object as a linear dynamical system [24]. This filter also incorporates measurements of the object's state from noisy sensors.

Kalman filter consists of two main steps: prediction and update [24]. In the prediction step, Kalman filter uses a state model to predict position and velocity of an object in both modalities at the next time step. In the update step, this filter combines measurements from both modalities using a measurement model to estimate a true state of an object.

3.5 Limitations

In the case of Non Maximum Suppression, at the end we retain only one bounding box per object - result of the "suppression" process [43]. Unfortunately, this results in a loss of information - we lose the second bounding box, provided from the other modality (sensor). Additionally, we do not build upon positives that each modality contributes. We only take the box with the highest score, given from the initial detector.

Kalman filters are usually used for data association through frames - to track objects in a sequence [24]. It can consider multiple variables of an object, including velocity and acceleration. As such it is described as computationally complex [24] [51] - especially for large systems with many state variables and objects. In real-world scenarios, this leads to poor real-time performance.

Regarding Nearest Neighbor and K-means, we are met with poor performance when we have false positives - an issue that arises often, when using LiDAR detectors [6]. In addition, K-Means does not always find the optimal solution for its cluster centers and needs to know a priori the number of clusters [6]. Even in the case when we assign the number of clusters to be equal to the lowest number of detections from either modality in a specific frame, we can again expect poor performance due to a possibility of false positives and thus creating more clusters than necessary.

Chapter 4

Dataset

4.1 A9 Dataset

Figure 4.1: The *Providentia++* test bed. [11]

In this thesis, we will be focusing on a roadside (infrastructure-side) dataset - the A9 dataset [11] from the TUM RCIL Lab.

The A9-Dataset is constructed using sensor infrastructure from the 3-kilometer-long *Providentia++* test field located near Munich, Germany. The dataset comprises of high-resolution, multi-modal sensor and object data, encompassing a wide range of traffic scenarios.

The first set of data (R0) includes camera and LiDAR frames collected from two overhead gantry bridges on the A9 autobahn, along with the corresponding object labels represented by 3D bounding boxes. R0 contains recordings of dense traffic on the autobahn and includes in total 1,098 labeled frames and 14,459 labeled 3D objects with an average of 13.17 labels per frame [11].

The entire test field, including R0, encompasses diverse types of roads such as the A9 autobahn, rural and urban scenarios, as well as, intersections. The dataset is geared towards enhancing the robustness of vehicle detection models and improving their perception capabilities [11].

The A9 dataset is one of the first publicly available datasets that contain diverse road scenarios in diverse lighting and weather conditions and was captured by stationary multi-modal sensors on gantry bridges [11].

4.1.1 A9 Intersection R1

While the first R0 release contains recordings of dense traffic on the autobahn (High-Speed dataset), the R1 part of the dataset contains another section of the test bed - s110 (Figure 4.1), a fairly busy intersection. R1 includes camera and LiDAR frames collected from one overhead gantry bridge on the intersection, in comparison to the two bridges on the A9 autobahn for the R0 dataset, along with the corresponding object labels represented by 3D bounding boxes. It contains images and point clouds to the amount of 9,600 consisting of 57,743 labeled 3D objects.

R1 Scenes

The R1 release of the dataset comprises of a variety of scenes and environmental conditions. It includes day and night scenes, as well as sunny (with reflectivity) and rainy weather, including a scene with both night and rain. The dataset is split in test, train and validation with an 80, 10 and 10 ratio. This is done through the process of stratified sampling [59] - we sample each scene (subgroup) independently. Apart from sampled frames (with images and point clouds), the test set additionally contains a 100 frame test sequence.

The dataset is recorded with the help of two camera sensors (Basler) and two LiDAR sensors (Ouster), with the cameras having only a small angle of field-of-view (FoV) intersection between them. The two LiDAR sensors are positioned respectively near to the respective cameras, but due to their wider FoV they are able to produce detection results in both camera images.

In terms of object labels, the R1 release provides 10 different classes for evaluation. Their distribution among the whole dataset and among the test set is illustrated in detail in Tables 4.1 and 4.2.

Table 4.1: Train/Validation/Test set. Average width, length and height, rounded to the 4th element after the decimal point, as well as number of instances. Categories *EMERGENCY VEHICLE* and *OTHER* have the lowest number of instances throughout the dataset.

Category	Width	Length	Height	Instances
CAR	1.90	4.25	1.60	33212
TRUCK	2.91	3.04	3.44	4862
TRAILER	3.14	9.93	3.66	5440
VAN	2.56	6.40	2.51	6830
MOTORCYCLE	0.86	1.73	1.63	1190
BUS	3.14	12.41	3.42	1696
PEDESTRIAN	0.75	0.82	1.79	3591
BICYCLE	0.86	1.42	1.73	755
EMERGENCY VEHICLE	2.37	6.17	2.19	83
OTHER	1.93	5.28	1.90	84

Figures 4.2 and 4.3 provide a better picture of the underlying class distributions. Although we are using stratified sampling on the basis of scenes only, we can still see that the class

Table 4.2: Test set. Average width, length and height, rounded to the 4th element after the decimal point, as well as number of instances. Categories *EMERGENCY VEHICLE* and *OTHER* have the lowest number of instances throughout the test set.

Category	Width	Length	Height	Instances
CAR	1.92	4.28	1.63	3211
TRUCK	3.02	3.14	3.60	682
TRAILER	3.26	10.81	3.88	741
VAN	2.62	6.17	2.62	1119
MOTORCYCLE	0.76	1.17	1.67	181
BUS	3.46	12.40	3.61	422
PEDESTRIAN	0.65	0.81	1.74	509
BICYCLE	0.81	1.48	1.75	114
EMERGENCY VEHICLE	2.30	6.29	2.25	2
OTHER	1.93	5.28	1.91	4

distributions are relatively saved. In fact, apart of classes *CAR*, *TRAILER* and *EMERGENCY VEHICLE*, we retain or even exceed the percentage of objects in comparison to the full set. This can be considered as a positive (apart from class *EMERGENCY VEHICLE*), as both *CAR* and *TRAILER* have respectively the first and third most examples. Thus, we end with a distribution that includes roughly enough examples of all possible classes.

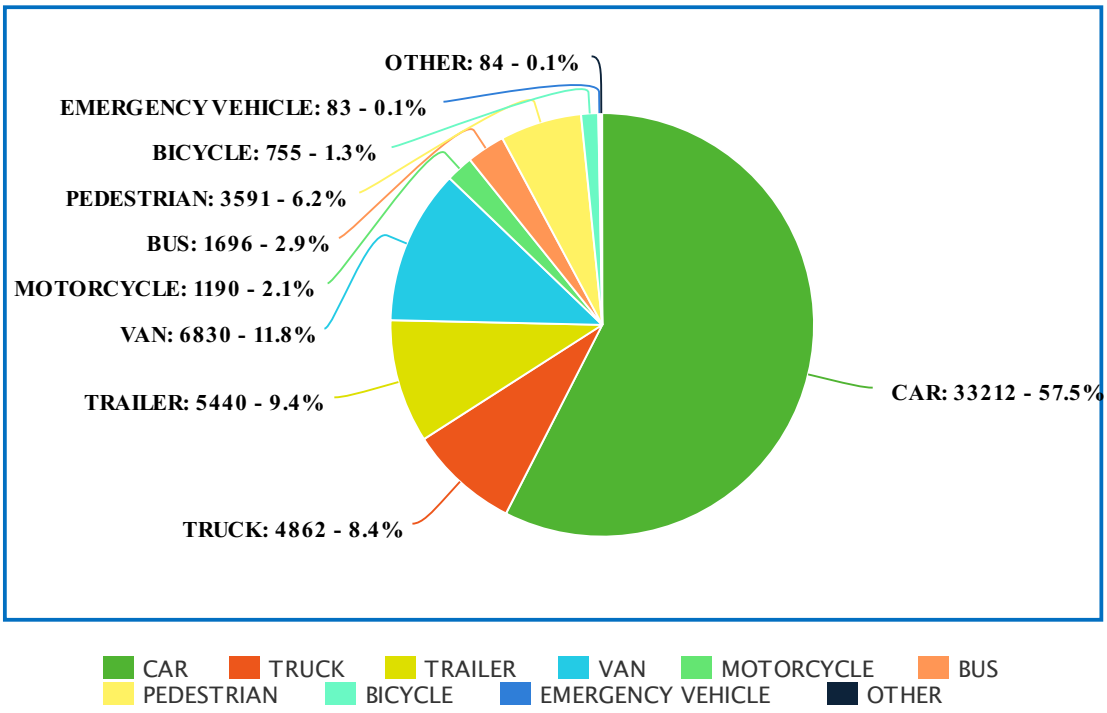

Figure 4.2: Train/Validation/Test set. Class distribution. (Created with Meta-chart [40])

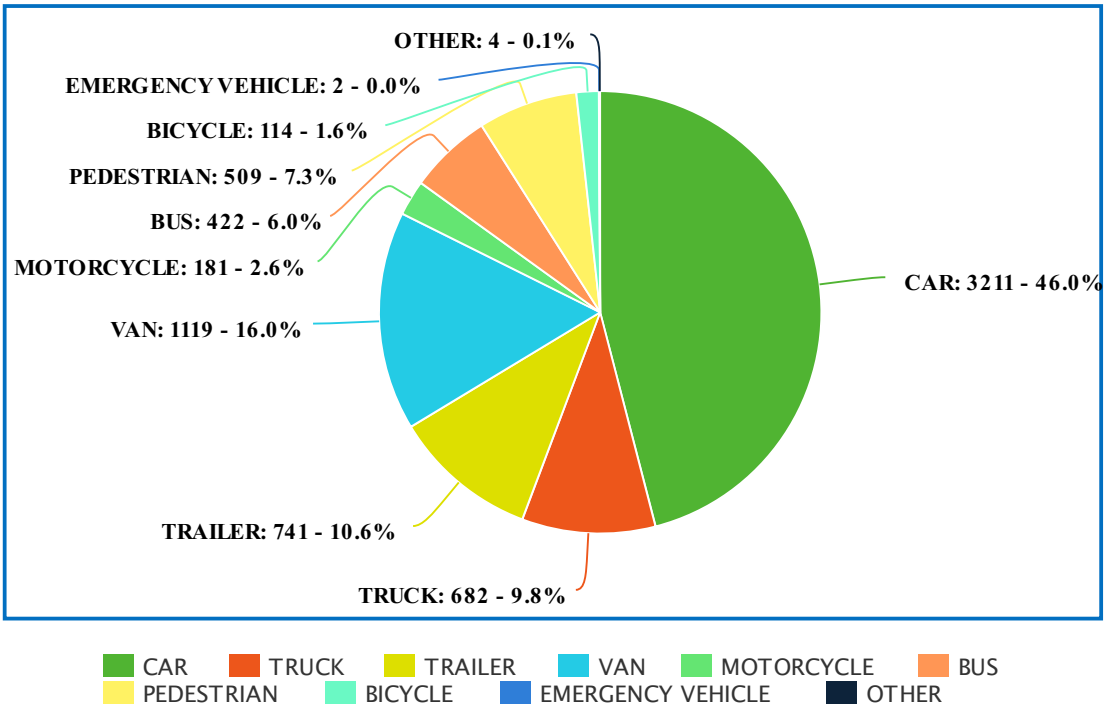

Figure 4.3: Test set. Class distribution. (Created with Meta-chart [40])

Chapter 5

Implementation

5.1 Background

5.1.1 Sensor Calibration

In the multi-modal 3D object detection different infrastructure sensor detections are fused and processed together, meaning that the calibration between LiDARs and cameras is one of the initial building blocks of the system.

To calibrate the sensors, an automatic targetless LiDAR-Camera calibration model based on the paper published by Liu et al. in Hong Kong University [8] was developed. In addition, to improve the robustness of the model under various external conditions, such as different scene complexities, lighting conditions and sensor conditions, various automatic pre-processing submodules were developed.

5.1.2 Monocular 3D Detector

The monocular 3D Detector consists of 2 parts: 2D object detection using YOLOv7 mask segmentation [65], pretrained on the MS COCO dataset [32] and a monocular detection pipeline utilizing L-Shape-Fitting [73] and HD maps for heading lookup.

The monocular 3D detector is based on an augmented L-Shape-Fitting algorithm as proposed initially by [73]. The augmentation of this algorithm with object tracking, to score yaw hypotheses based on historical plausibility, is novel. Furthermore, we propose the integration of the High-Definition (HD) map to limit yaw hypotheses with regard to matching lanes - features inspired by TrafficNet [52] and UrbanNet [5] architectures.

A YOLOv7 Instance Segmentation model [65] on RGB camera frames is used. The RGB frames are firstly down-scaled to 1280x720 pixels, to speedup the instance segmentation runtime. The instance masks are further processed to extract the bottom image contour from the respective masks. The 2D bottom contour coordinates for each mask are then projected from screen-space to 3D intersection space via the process of ray-casting.

Using an HD map for lane geometry estimation of the intersection, each lane's road surface is converted into a heading lookup framework, covering the field-of-view of the cameras (south1 and south2).

5.1.3 LiDAR 3D Unsupervised Detector

The unsupervised 3D LiDAR object detector is based on clustering and utilizes a four step process for background filtering:

First step: the detector crops a region of interest (ROI).

Second step: the detector finds points belonging to the ground by considering the Euclidean distance to a predefined plane model together with a threshold of 0.2 m.

Third step: the detector filters background artifacts within the region of interest based on the coarse-fine triangle algorithm [72].

Forth step: radius outlier removal, which refines the extraction of the foreground point cloud.

The remaining point cloud shows only traffic objects. DBSCAN is used to divide the point clusters [55]. Around each point cluster, the detector fits an oriented 3D bounding box using principal component analysis (PCA). Finally, the detector classifies the localized objects by means of object dimensions and point density.

5.1.4 LiDAR 3D Supervised Detector

For the data-driven approach, PointPillars [28] is used. It runs at a fast inference rate of 38 FPS. In comparison to the unsupervised approach, we can input the registered point cloud (262k points before filtering) directly into the model, consisting of three modules.

In the first step, the *PillarFeatureNet* converts the point cloud into a sparse pseudo-image.

After obtaining the pseudo-image, the 2D backbone produces features at a small spatial resolution.

In the last step, an anchor-based detection head tries to match the bounding boxes to the ground truth. The PointPillars implementation of OpenPCDet [62] was used and adapted to the A9 infrastructure dataset.

For training, the point cloud range was limited from -64 m to 64 m in $x-y$ direction and from -8 m to 0 m in z direction and the voxel size is set to $[0.16, 0.16, 0.8]$. The model was trained on 10 classes for 160 epochs.

5.2 Late Fusion

5.2.1 First iteration

In the case of fusing camera and LiDAR detections, one common approach is to use the Intersection over Union (IoU) as a metric for multi modal bounding boxes association. The IoU is a metric that measures the overlap between two bounding boxes, and it is calculated as the ratio of the area of intersection of the boxes to the area of union of the boxes [53] [44]. In the context of object detection, multiple detection boxes from different modalities may correspond to the same object. By computing the IoU between each pair of boxes, the boxes that overlap significantly (above a certain IoU threshold) can be considered as detections of the same object.

To fuse the detections, one common approach is to merge the overlapping boxes into a single bounding box that encompasses the area of overlap. This process can improve the

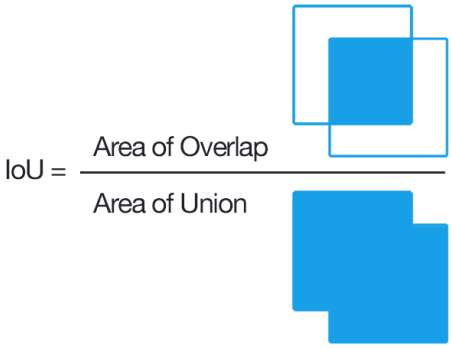

Figure 5.1: 2D Intersection over Union (IoU). [22] [53]

accuracy and robustness of the detection by combining information from multiple modalities. The choice of IoU threshold is a trade-off between precision and recall, with a higher threshold leading to more precise detections, but potentially missing some true detections, and a lower threshold leading to more complete detections, but with a higher risk of false detections.

In our case, the variation is that we are working with 3D bounding boxes and the intersection between those boxes. Moreover, we do not need the box that we receive from the intersection, as the boxes may appear in the end smaller than the actual object. To circumvent that, we can use the intersection of the 3D Bounding Boxes as just a measure of association - which bounding boxes correspond to each other from both modalities, based on the IoU between them. Afterwards, we take the mean average of the corner points of the matched labels and thus fuse them.

Unfortunately, the results are not optimal. Camera and LiDAR labels often do not have the same orientation - they are positioned in the reverse direction. In addition, sometimes bounding boxes of two separate vehicles get fused together, when stationed on neighboring lanes. This happens due to the closeness of the bounding boxes - apart from the intersection between the LiDAR and monocular label, there is an intersection between one of those boxes and the respective box from the neighboring vehicle. These problems are better illustrated in Figure 5.2.

5.2.2 Data association

In the field of autonomous driving, accurately detecting and tracking objects in the environment is essential for the safety of navigation. One approach to achieving this is by fusing information from multiple modalities, such as cameras, LiDARs, and radars, at a later stage of the perception pipeline, known as late-fusion detection. The main idea behind a late-fusion detection approach is to improve the accuracy and consistency of object detection by leveraging the redundancy in the sensor data. Each modality generates a set of detections independently. These detections are then associated with each other and fused - to combine the strengths of each modality. One of the advantages of this approach is that it can effectively handle situations where one modality may perform better than the others, such as LiDAR in low light environments or cameras in hazardous and adverse weather conditions. Additionally, this approach can also handle situations where one modality may not be able to detect an object while the other modalities can, thus increasing the overall robustness of the given system (Figures 5.3 and 5.4).

A widely adopted method for combining and matching sensor data at the later stage is through data association, also defined as the linear assignment problem (LAP) - finding a

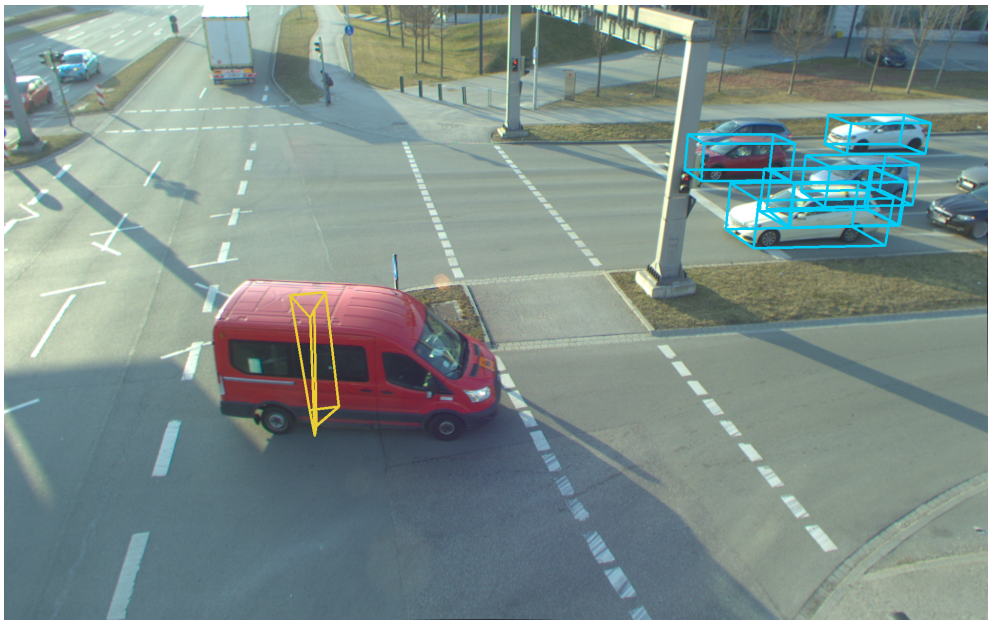

Figure 5.2: South1 Basler camera field-of-view, scene 4, LiDAR-Camera Label Fusion. First thing to identify is the vans bounding box, which is completely squashed, but still relatively correct on the center of the vehicle. This is caused by the orientation of the two labels - while the monocular label is oriented to the right of the view, the LiDAR label is pointing to the left (180 degree difference). Since we are taking the IoU threshold only to match labels, but we are taking the mean average of the corresponding corner points, we arrive at this result. Additionally, there is a false bounding box created between the cars, since they are stationed next to each other and one of the bounding boxes makes an additional intersection with the bounding box of the other car, resulting in a false positive

one-to-one mapping between two sets of elements, such that the sum of the assigned pairwise costs is minimized. The costs can be represented in various ways, for example, in terms of distance, similarity, or any other metric.

Let there be n workers and n jobs, and let the cost of assigning worker i to job j be represented by a cost matrix $C = c_{ij}$, where c_{ij} is the cost of assigning worker i to job j [42]. The goal is to find a set of assignments $x = x_{ij}$ such that each worker is assigned to exactly one job, each job is assigned to exactly one worker, and the total cost is minimal. A perfect matching of minimum weight is called an optimum matching. This can be represented by the following integer linear program (ILP):

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ldots,n \\
& \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\ldots,n \\
& x_{ij} \in \{0,1\}, \ \ i = 1,\ldots,n, \ \ j = 1,\ldots,n
\end{aligned}
$$

Where x_{ij} has a binary value that is equal to 1 if worker i is assigned to job j and 0 otherwise. First and second constraints ensure that each worker is assigned to exactly one job and each job is assigned to exactly one worker. The third constraint ensures that the variables are binary. This problem can be solved using algorithms such as the Hungarian algorithm or the auction algorithm. The Hungarian algorithm (Kuhn-Munkres algorithm or Munkres assignment algorithm) is a combinatorial optimization algorithm that finds a perfect match in a bipartite graph, while the auction algorithm is a heuristic algorithm that

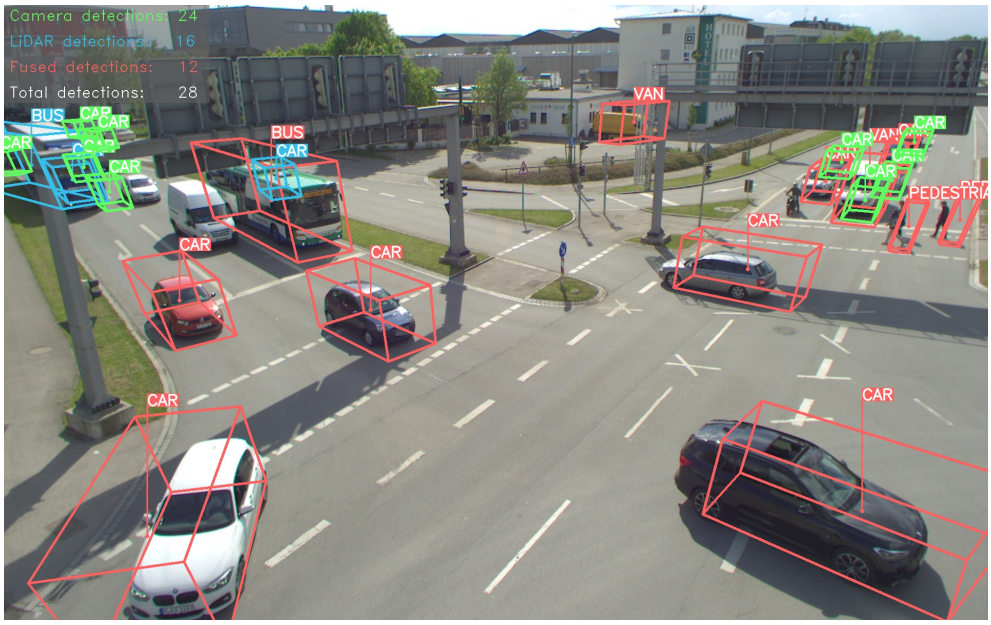

Figure 5.3: South2 Basler camera field-of-view. Among the LiDAR, monocular and fused detections, we can observe a car behind the bus detected by the LiDAR sensor, but is not viewable by the camera. This example showcases one of the strengths of using both modalities for a robust object detection system.

iteratively assigns workers to jobs based on the highest ratio of the reduced cost and the cost of the current assignment.

Another way to solve the problem is by linear programming relaxation which is a relaxation method of ILP where the integrality constraint, $x_{ij} \in \{0, 1\}$, is relaxed to $0 \leq x_{ij} \leq 1$. This method is known as Linear Assignment Problem (LAP).

Jonker-Volgenant algorithm [23] is a method for solving the LAP and is based on augmenting paths. The algorithm starts by finding an initial feasible solution, e.g. by using the Hungarian algorithm [27]. Then, it repeatedly searches for an augmenting path, a path of alternating unmatched and matched elements that starts and ends at an unmatched element and increases the number of assigned elements by one. The algorithm stops when no augmenting path can be found, which means that the solution is optimal.

The *modified Jonker-Volgenant* algorithm [12] is a variation of the original that improves its performance by using a heuristic search strategy. This heuristic builds on the idea of prioritizing the search for augmenting paths that are expected to have a high gain in terms of reducing the total cost.

In this master thesis, the *modified Jonker-Volgenant* algorithm is chosen due to its increased speed ($O(n^3)$ [12]) in comparison to its variants. It can be used to solve large-scale LAP instances in a reasonable amount of time, making it more applicable in real-world scenarios. It also works specifically well with non-integer costs - important to our case, where our metric for cost is represented by the distance of centers between two detections, which is rarely an integer.

5.2.3 Python Implementation

The modified Jonker-Volgenant algorithm [12] without initialization is conveniently implemented in `python` by the `linear_sum_assignment` function [56], part of the `scipy.optimize` package, of the `scipy` API [57].

`Scipy` is an ensemble of mathematical functions built upon the `NumPy` extension of Python

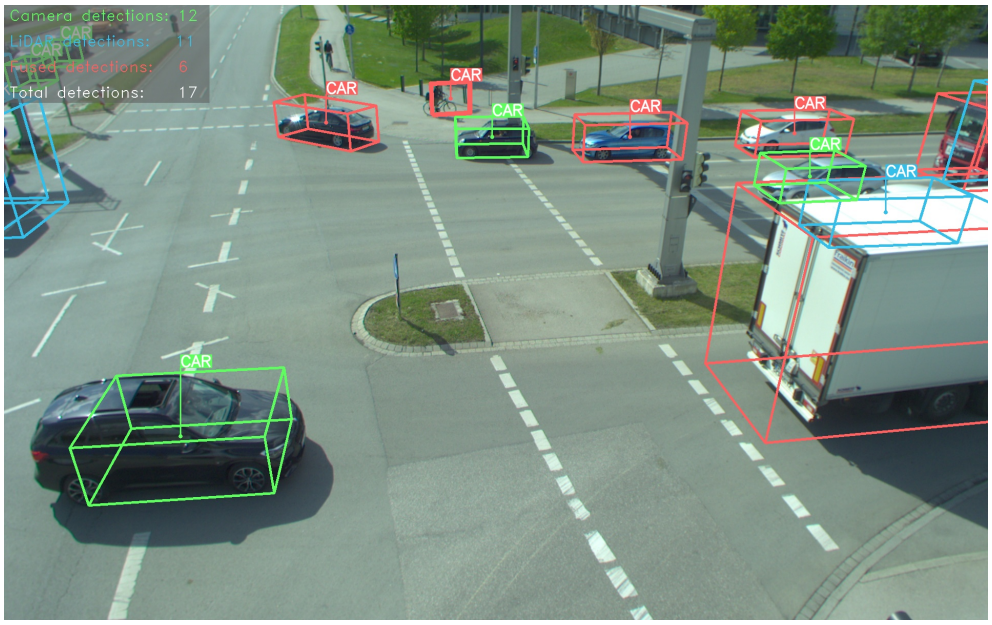

Figure 5.4: South1 Basler camera field-of-view. Another example for the robustness of the system by identifying occluded objects. Here we can observe a car behind the truck detected by the LiDAR sensor, but is not detected by the camera, and its only barely visible.

[57]. It adds significant functionality for the user with high-level API commands and classes, useful for data augmentation, as well as visualization. SciPy distinguishes itself as a data-processing and system-prototyping environment for Python, direct rival to MATLAB and other similar systems.

This version of the `linear_sum_assignment` function can also solve a broader generalization of the classic assignment problem where the cost matrix is not square but has a rectangular form - in the case where there are more columns than rows then not every column needs to be assigned to a row [56].

In our tests, the matching process per frame (from the test set) took 0.0076 ms on average on the test set on an AMD Ryzen 5800X 8-Core CPU with an average of 14.55 objects per frame. In comparison, the `munkres` library [42] in `python` takes around 0.0133 ms on average, tested on the same CPU - a nearly 2x difference in performance.

As these results show, the performance of either implementation is not an inconvenience. But one of the pursuits in this master thesis is to find an implementation that would suite a late-level fusion of a multitude of detections from both modalities, no matter the total number of said detections. As all single modality detectors, that this master thesis approach uses as its base, are quite memory and computationally expensive (YOLOv7, Monocular 3D detector, Supervised and Unsupervised 3D LiDAR Detectors), the fusion process should not additionally impede the systems performance.

5.2.4 Fusion

For the fusion itself we match detections by the distance between the centers of the 3D Bounding boxes (cuboids) from the two modalities - the costs in the cost matrix are the respective distances between the detections, with rows being the monocular and columns the LiDAR detections. However, in our case, we don't necessarily need all of the detections matched - only the ones that actually correspond to one another. In cases where there is no correspondent LiDAR to monocular detection, one of those detections could be matched

to a false one. To resolve this, we introduce a distance threshold - all detections above this threshold get thrown out of the matched pairs set and put into the unmatched pairs set. To determine this parameter we take a look at the German Road Traffic Licensing Regulations (StVZo) for pointers [60] - the center of the traffic lanes in Germany are between 2.7 m and 3.5 m apart and cars have an average length of 4 m (with an additional 0.5-1 m distance between them). Therefore we can safely choose a distance threshold of 3 m for our fusion on an intersection. In the case of a highway, the distance threshold can be set to 3.5 to 4 m, without causing falsely fused detections.

After the fusion process is complete we are left with three sets of detections - matched (monocular and LiDAR), unmatched LiDAR and unmatched monocular. Additionally, for the matched detections we have to decide on a fusion method - which detection attributes to take from which modality - center 3D position, yaw angle, dimensions of the object and the corresponding object category.

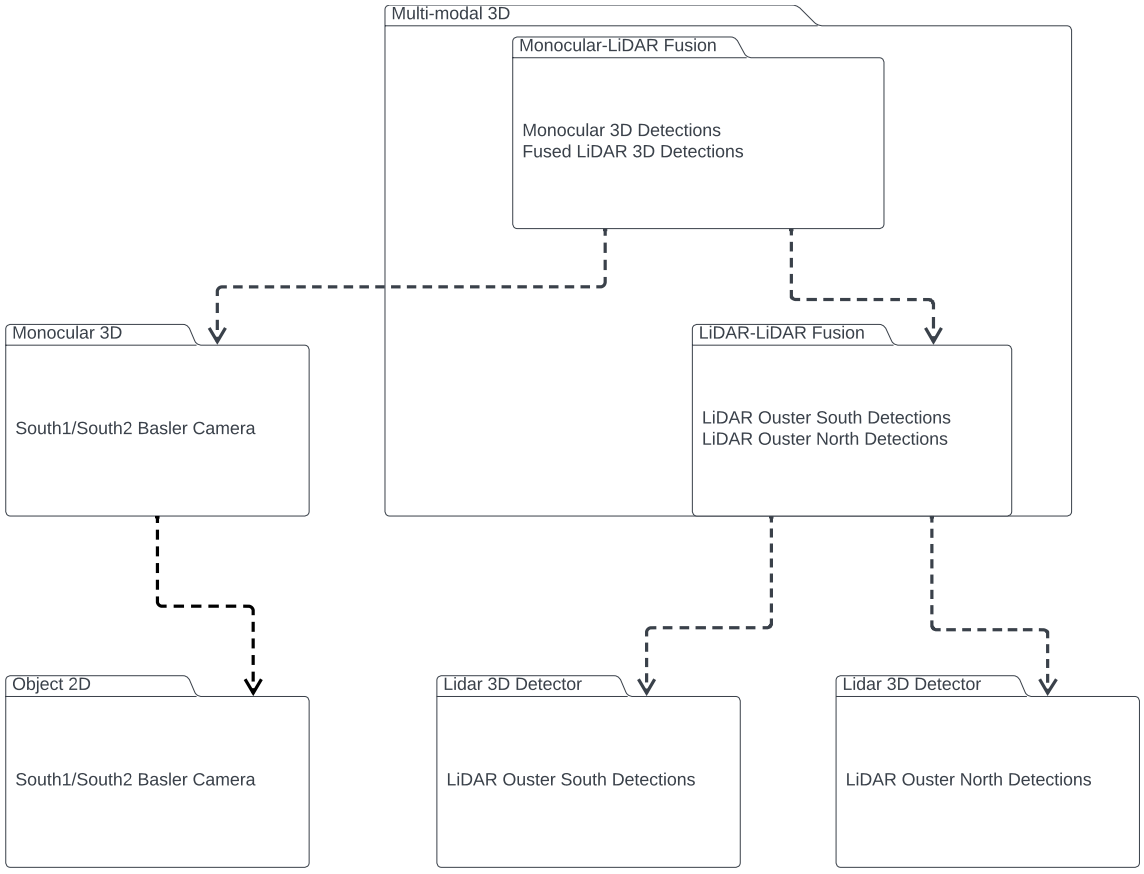

Figure 5.5: Package module structure of the multi modal 3D detector in UML format.

5.3 Multi Modal 3D Detector

The multi modal 3D Detector consists of 2 parts. Initially, a LiDAR-to-LiDAR fusion process is performed, followed by an additional integration step that incorporates the detection results obtained from the Monocular 3D Detector.

5.3.1 Synchronisation

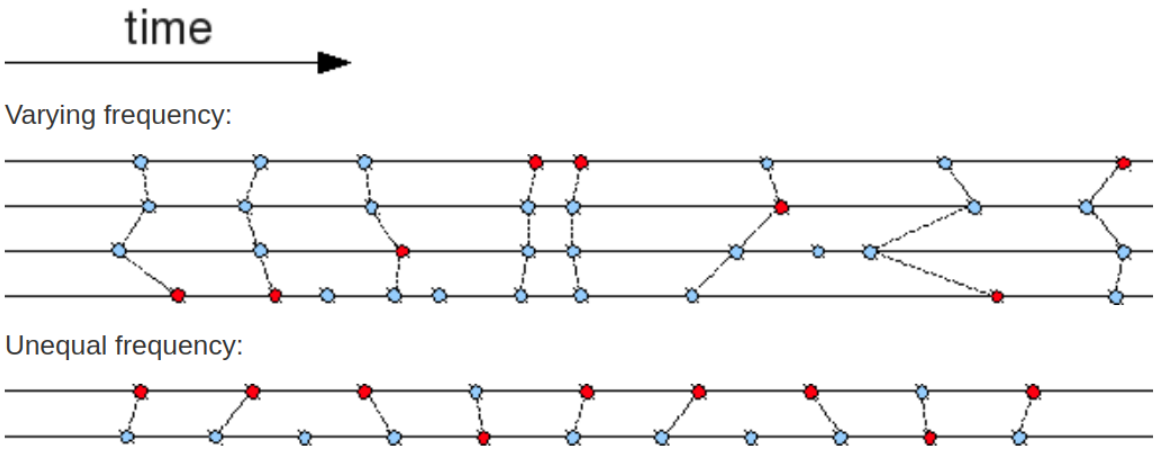

Figure 5.6: `ApproximateTimeSynchronizer`. Dots are messages and lines - topics. Red messages represent the pivot point. The broken lines link the messages in sets. [54]

The Robot Operating System (ROS) is an open source platform for building and integrating components of a robotic system that promotes the development of robust, scalable, and reusable robotic applications. It provides a set of libraries and tools that support the development of robotic applications, including device drivers, algorithms, visualizers, and other essential components.

The ROS `message_filters` package is a library that provides a convenient way to synchronise and filter messages in the ROS environment. It is designed to improve the performance of robotic systems by processing and integrating information from multiple sources in real-time.

The package works by using a time-based or message-based approach to associate and filter messages from different sources. For example, as in our case, it can be used to synchronize data from camera and LiDAR sensors, so that both data streams can be processed together in a meaningful way. This allows the whole system to make use of the complementary strengths of each sensor and to achieve better performance and robustness in the presence of challenging environmental conditions.

The `message_filters` package offers a number of different filters, including `TimeSynchronizer` and `ApproximateTimeSynchronizer`. These filters can be configured with different policies for synchronizing and filtering messages.

For our specific use-case, the `ApproximateTimeSynchronizer` class is used for synchronising incoming detections from both modality detectors. It is a message filter that synchronises multiple ROS topics by their timestamps, using an approximation to reduce the computational cost of the synchronization process. The `ApproximateTimeSynchronizer` works by keeping a buffer of the most recent messages from each topic that it synchronizes, and then selecting a subset of these messages that have similar timestamps and pass them to a callback function for processing. The size of the buffer and the approximation algorithm

used to determine the similarity of the timestamps can be configured to trade off between the accuracy of the synchronization and the computational cost of the filter.

`ApproximateTimeSynchronizer` takes a couple of parameters, the most important of which - the list of message filters, a `slop` argument (defines the synchronization delay in seconds) and the queue size (how many messages can be queued). For the `slop` parameter we take a delay of 1.0 seconds and for the queue we set a size of 100. This means that we can synchronize messages with a difference in capture of maximum 1.0 second and we can queue up to 100 messages for synchronization.

5.3.2 LiDAR-to-LiDAR Late Fusion

As discussed in Chapter 2, one of the most prominent advantages of LiDAR sensors are their ability to provide high-precision depth estimation (Table 2.1). In order to maintain this accuracy in depth assessment (in 3D space), especially important for visualizations, it is essential to perform all subsequent processing steps for the LiDAR-to-LiDAR fusion in the coordinate space of one of the LiDAR sensors. Therefore, we transform the detections obtained by the unsupervised LiDAR detector and the supervised LiDAR detector into the common coordinate system of the south LiDAR sensor. We match 3D detections based on the Euclidian distance between their central positions. The matched detections are merged by selecting the central position and yaw vector of the detected object from the LiDAR sensor closest to the detection. The dimensions of the merged detections are computed by calculating the mean average of the detections from both detectors. Additionally, all unmatched detections are also included in the final result, resulting in an increase in the number of detections compared to using only a single LiDAR sensor.

The reason behind the convoluted approach is the calibration behind the transformation and projection matrices. If we are to perform the fusion step for LiDAR-to-LiDAR in the base coordinate system of the gantry bridge, meaning to transform both LiDARs in s110 base coordinate system and then fuse them, part of the distance information provided in the LiDAR coordinate space is lost, which can negatively impact the efficacy of the distance threshold. Additionally, the projection of the LiDAR detections from the s110 base into the camera images for south1 and south2 is also susceptible to calibration errors. The more optimally calibrated projection matrices pairs, essential for visualization purposes, are LiDAR Ouster north and Camera Basler south1, LiDAR Ouster south and Camera Basler south2.

5.3.3 Camera-LiDAR Fusion

The biggest problem that we encountered during the fusion process was the acquisition of detections. Part of the detections, matched after the fusion process, have a different direction from one another - a 180 degree reversed rotation. Due to this limitation it is not possible to simply take the mean average of the respective corner points of the matched 3D cuboid bounding boxes and thus merge them, as in these not so rare occasions the rotation of the detection is perpendicular to the objects real-world direction orientation. In other terms, the detections may be matched, but their corner points aren't. To solve this issue, two approaches are possible.

Double Hungarian Fusion

This approach consists of applying data association two times per object. As mentioned in Section 5.2.2 we use the modified *Junker-Volgenant* algorithm [12] to match the detections.

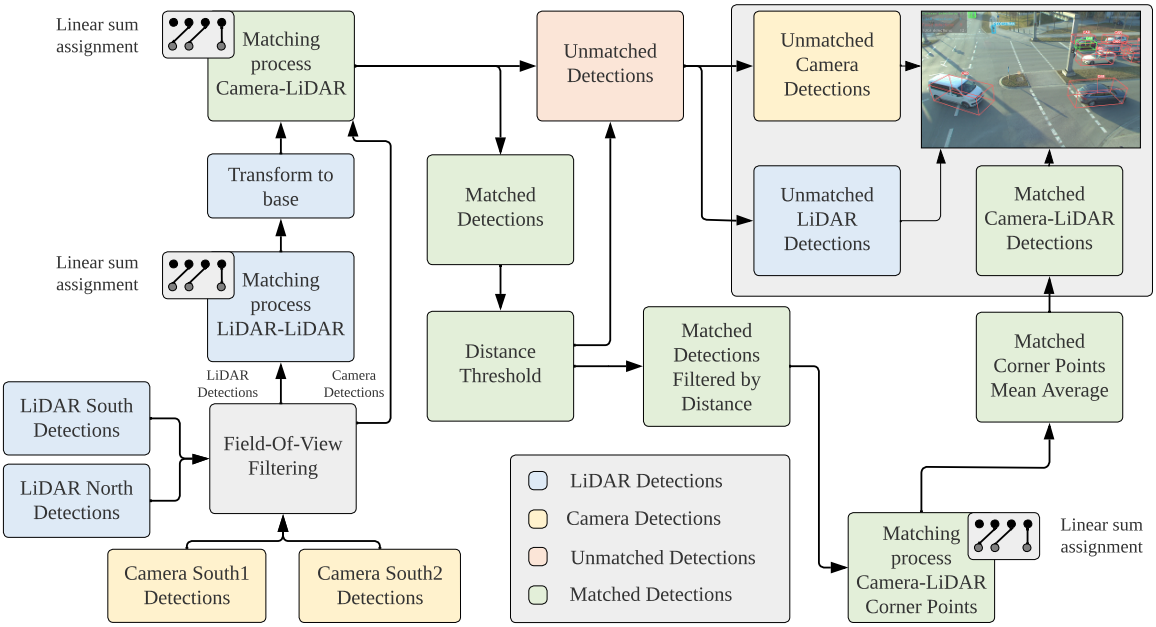

Figure 5.7: Double Hungarian Fusion pipeline. We apply a camera field of view filtering for all detections. LiDAR detections are then matched in the LiDAR coordinate system with a distance threshold of 3 m. After the LiDAR-to-LiDAR matching process, all of the detections are transformed into the base coordinate system of the gantry bridge, the same coordinate system as the camera detections. Camera and LiDAR detections are matched using the modified Jonker-Volgenant algorithm [12]. The matched detections are further filtered by a previously specified distance threshold of 3 m - the same as the LiDAR-to-LiDAR matching process. The filtered matched detections are fused by applying another step of the matching process between their respective corner points. The result is a mean average of the center position, yaw vector and dimensions.

After the matching process, we apply a distance threshold of 3 m to eliminate matched detections that are too far apart to be of the same object. After this step, we go into each and every matched pair of detections and apply an additional step of the linear sum assignment to find matching pairs between the corner points. During this processing step we do not need a distance threshold, as all corner points should find a match. Subsequently, in order to construct the new corner points of the matched detection, we take a mean average of the corner point positions of the matched corner points and thereafter set the new points as the corner points of the new merged detection. This leads to a fused bounding box, where we not only take the mean average of the center position of the 3D cuboid, but also the mean of the yaw angle, as well as the mean of the dimensions. We obtain a complete mean of the 2 detection bounding boxes. The only 2 additional attribute that can be taken from either the monocular or LiDAR side are category and UUID. The score can either be taken as is from either modality or the mean average of both scores can be calculated. We implemented the latter (mean).

Unfortunately, there is a caveat to this solution. As the analysis in chapter 6 shows, LiDAR detections from the supervised approach using Pointpillars [28] prove to be more robust in comparison to the monocular detections in terms of all detection attributes - center location, orientation vector and, especially, dimensions. While providing a more clear-cut solution to the fusion problem, this solution disregards the benefits that the supervised LiDAR detector provides.

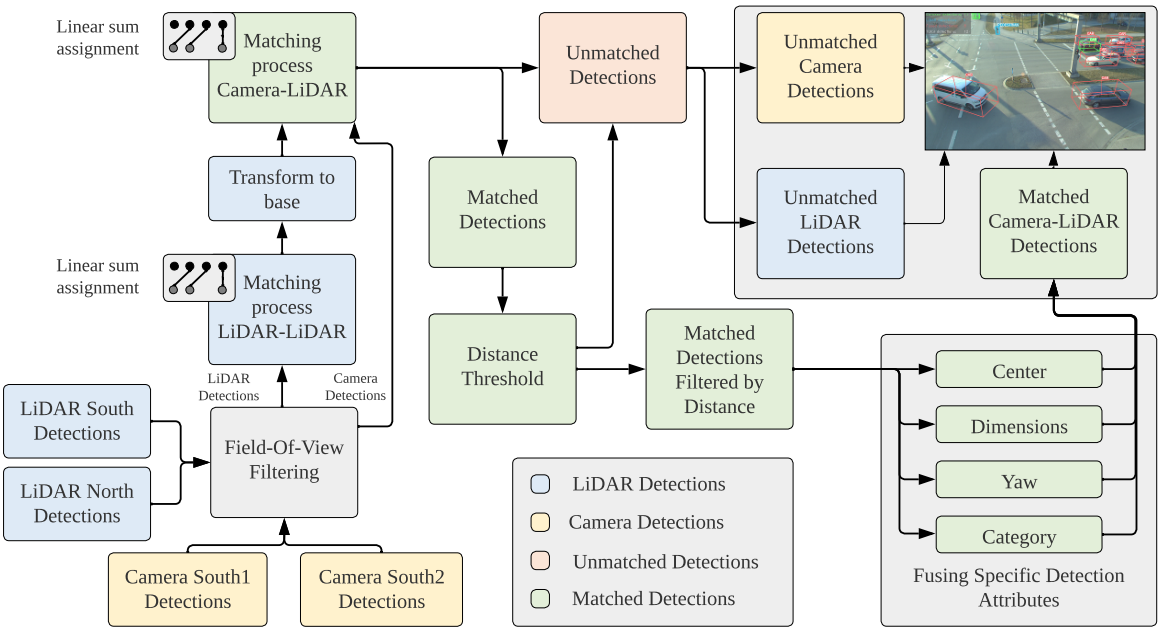

Figure 5.8: Mixed Attributes Fusion pipeline. We apply a camera field of view filtering for all detections. LiDAR detections are then matched in the LiDAR coordinate system with a distance threshold of 3 m. After the LiDAR-to-LiDAR matching process, all of the detections are transformed into the base coordinate system of the gantry bridge, the same coordinate system as the camera detections. Camera and LiDAR detections are matched using the modified Jonker-Volgenant algorithm [12]. The matched detections are further filtered by a previously specified distance threshold of 3 m - the same as the LiDAR-to-LiDAR matching process. The filtered matched detections are fused using the specific detection attributes.

Mixed Attributes Fusion

For the camera-LiDAR fusion, we transform the LiDAR detections into the base coordinate system of the gantry bridge, which serves as the coordinate system for obtaining the monocular detections. This step is crucial for computing the inter-detection distances between camera and LiDAR instances based on their respective center positions. After the linear sum assignment, the matched detections are further filtered by a distance threshold of 3 m. The attributes of the matched detections are merged by eliminating matched camera detections and retaining only matched LiDAR detections, as they demonstrate greater accuracy on average during evaluation. Table 5.1 displays the dependence of the mAP increase on the various attributes. Additionally, a good example of the only matched objects can be seen on Figure 5.9.

Table 5.1: mAP_{3D} scores improvement for matched detections by accepting only LiDAR detection attributes, calculated for camera south1 using early LiDAR-to-LiDAR and late fusion.

Fused Attribute	**Improvement in mAP_{3D}**
Center position	+2.85
Yaw	+0.3
Dimensions	+1.24
Category	+13.11
Total improvement	**+17.5**

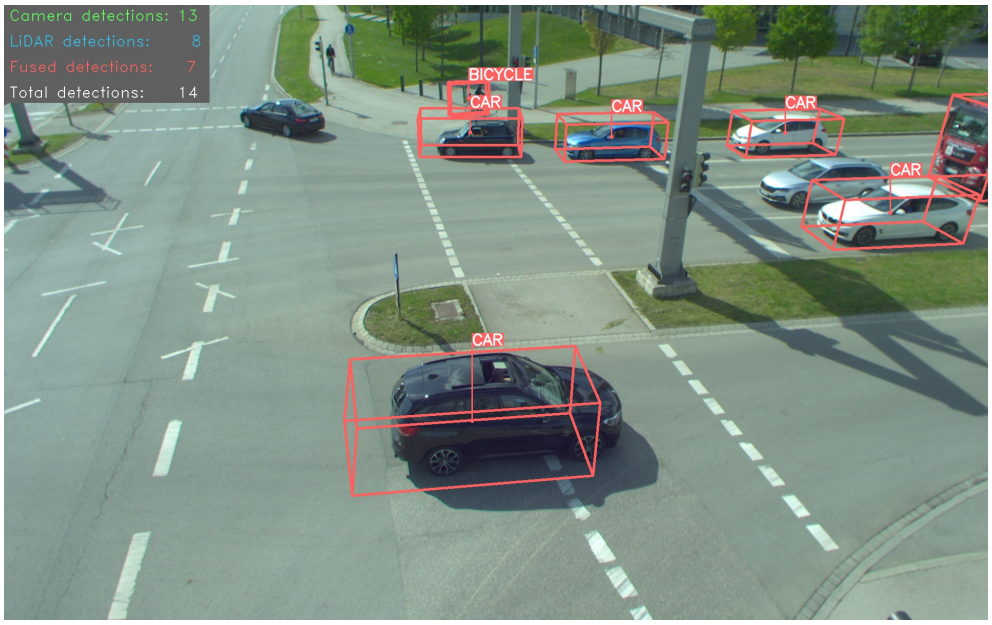

Figure 5.9: South1, test sequence example, only matched vehicles. Visualization example uses LiDAR dimensions and category, Monocular center and yaw vector. Scores and UUID are irrelevant for the visualization and therefore simply the ones from camera are chosen.

5.3.4 Comparison between Fusion Methods

As the IoU threshold performs rather poorly - incorrect fusion of detections, including the creation of false boxes when an additional intersection is present, thus contributing to more false positives, we will not be considering it in this comparison section. Double hungarian and mixed attributes fusion are demonstrated in Figure 5.10. Qualitative analysis shows that by mixing attributes between the two modalities provides better results.

By using the double hungarian approach, we get the mean average of not only the center position and yaw vector, but also the dimensions, which is problematic, as they are more correctly represented by the detections coming from the LiDAR sensors (one of the positive sides of the LiDAR modality - dimension estimation Table 2.1).

Center position and yaw vectors are also a little more accurate when considering only LiDAR attributes (Table 5.1). This indicates an additional factor for consideration - the labels used for evaluation are the ones taken fom LiDAR. Further, we have a synchronization error between LiDAR and camera of around 70 ms on average.

5.4 Problems

5.4.1 LiDAR 3D Unsupervised

Regarding the unsupervised LiDAR detector there were a couple of problems during the integration part.

The detection passed through the Robot Operating System (ros) publisher and subscriber nodes (and also saved detections) had the length and width variables switched, resulting in detections that seemed rotated by 90 degrees. Therefore whenever we are using the unsupervised approach we switch those dimensions for the correctly oriented 3D bounding box.

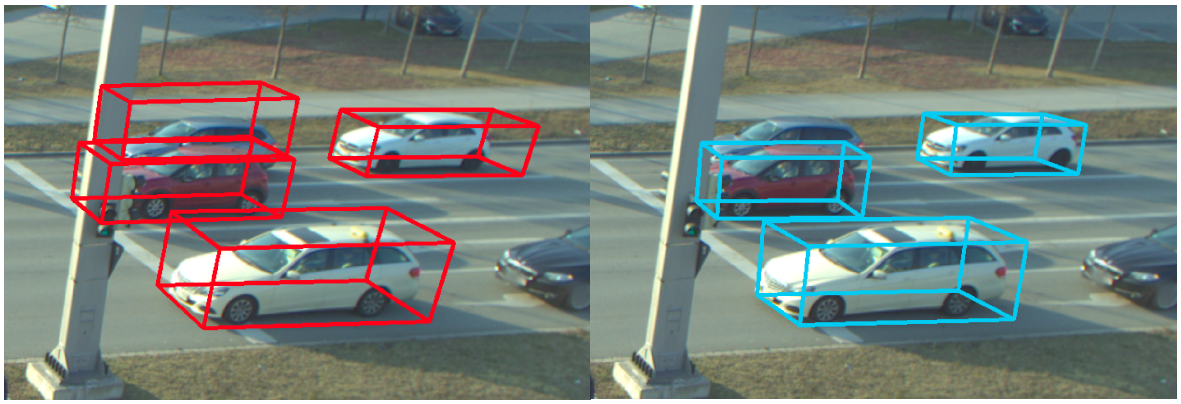

Figure 5.10: South1, day scene, only matched vehicles. *Left:* Double Hungarian Fusion, *Right:* Mixed attribute fusion (LiDAR dimensions, camera center position and yaw). Mixed attribute fusion provides better dimension and center position results.

Additionally, as the detector in itself uses an unsupervised clustering approach based on DBSCAN [55], there are more detections recognised than actually present - false positives. Moreover some of the detected objects are only partially recognized in the point cloud space. This issue produces false dimensions for our detections. This is especially troublesome as the dimension estimation is one of the most beneficial aspects of the detections provided by LiDAR sensors (Table 2.1).

The last problem is fixed by adding an additional step to the multi 3D pipeline - category dimension mapping. We take the category from *Monocular 3D* detection, as classification estimations give relatively better results, in general (Table 2.1), and the dimensions from *LiDAR 3D Unsupervised*. Wherever the dimensions gathered from clustering are not positioned inside a previously specified range for the respective category, the dimensions get mapped to the default minimum (if the dimensions are lower than the minimum for the class) or maximum (if the dimensions are higher than the maximum for the class) dimensions of the respective category/class.

5.4.2 LiDAR 3D Supervised

The *LiDAR 3D Supervised* detector, based on Pointpillars [28], also exhibits an issue regarding false positives. During day and night scenes, occasionally the detector would show a bounding box, where no object resides in. Unfortunately, there is no easy way to solve this issue.

One possibility would be to use and visualize only the matched detections from the fusion, in order to provide a more accurate detection evaluation. The monocular detections act as a filter in a way - they filter out all of detections that do not appear in both modalities. We still take all of the attributes of the LiDAR detections in the end, as they are more accurate and score higher during evaluation (Table 5.1) - +17,87 `mAP` (tested on south1).

Another more obvious issue is present in the south2 scenes - namely one of the road signs on the s110 intersection is recognized as class *PEDESTRIAN* (5.11). As the coloring scheme correctly shows, it is an unmatched detection, meaning only the LiDAR detector identifies it as such. This is due to the data diven approach of the supervised 3D LiDAR detector - the road sign is also present in the labels, mistakenly labeled as a *PEDESTRIAN*. Therefore, the *Pointpillars* detector is falsely trained to misidentify the mentioned sign.

Fortunately, the identified object of class *PEDESTRIAN* is taller than a normal human 2,2 - 2,5 m .To circumvent this obstacle, we simply add a check inside the unmatched LiDAR

detections to eliminate all detections of class *PEDESTRIAN* that are higher or equal to 2,0 m. (Figure 5.11)

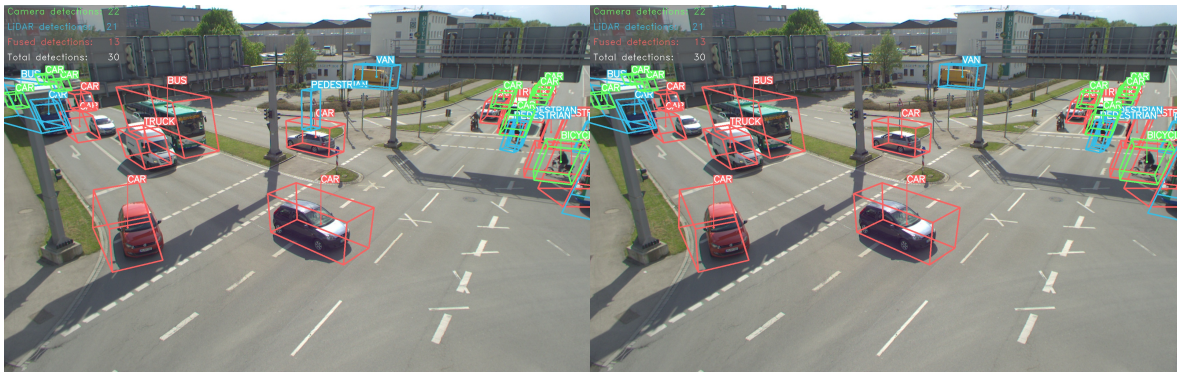

Figure 5.11: South2, day scene, Left: Road sign detected as class *PEDESTRIAN*, Right: Corrected image with height filtering for class *PEDESTRIAN*

Chapter 6

Experiments

6.1 Test Set

The test set consists of sampled frames from four scenarios, including a sequence of frames, throughout two camera sensors and two LiDARs - camera Basler south 1 and camera Basler south 2, LiDAR Ouster south and LiDAR Ouster north. The scenarios contain both day and night scenes, as well as, ten different object classes: *CAR, TRUCK, TRAILER, VAN, MOTORCYCLE, PEDESTRIAN, BICYCLE, EMERGENCY VEHICLE* and *OTHER*.

6.2 Labels

For labels, we have three options - LiDAR Labels, monocular labels or a fusion of both label modalities. Unfortunately, as the monocular labels are not in a 3D coordinate system, they are discarded. They merely depict the corner positions of the bounding boxes on a 2D plane - the camera image. In addition, labels in LiDAR prove to be more robust and accurate, and the dimensions, as well as yaw angles are closer to the ground truth. As such, the labels, used for evaluation, are the LiDAR labels.

We use three types of LiDAR labels: LiDAR Ouster south labels, LiDAR Ouster north labels and merged south and north LiDAR labels. The merged LiDAR labels are created again using the data association technique described in chapter 5. Through LiDAR-to-LiDAR Late Fusion we obtain a much greater number of labels (2.14x increase in merged compared to labels from only LiDAR ouster north). The fusion takes place in the south LiDAR coordinate system. After the matching process we set a distance threshold, upon which we filter the matched detections further, in order to eliminate bad matches. The distance threshold set is the same as the one made for detections - 3 m. The unmatched detections are appended to the final output without additional changes made to them. All of the LiDAR labels are saved in OpenLabel format in the south LiDAR coordinate system.

The result of the fusion - the process introduces more and potentially more accurate labels in each camera without repetitions of labels on the same object. Before the evaluation, the LiDAR labels are further transformed into the base coordinate system of the gantry bridge with the help of the appropriate transformation matrix.

6.3 Performance Metrics

6.3.1 Precision and Recall

In an autonomous driving scenario, object detection is a crucial task, as the safety of passengers and other road users depends on the accuracy of the object detection system. In such scenarios, precision and recall are metrics used to evaluate the performance of a classifier or other type of predictive model.

Precision [63] [2] is a metric that measures the number of true positive (TP) results out of all positive results (true positive (TP) + false positive (FP)). In other words, it measures the accuracy of positive predictions made by the model. Precision is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall, also known as sensitivity or true positive rate [63] [2], is a metric that measures the number of true positive results out of all actual positive results. In other words, it measures the ability of the model to find all the positive instances in the data. Recall is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

In our scenario, it can be summarised as:

True Positives (TP) are the number of instances where the system correctly detects an object in the image.

False Positives (FP) are the number of instances where the system detects an object in the image where there is no such object.

False Negatives (FN) are the number of instances where the system fails to detect an object in the image, although there is one.

It's important to note that high precision and high recall don't always go hand in hand, and the ideal trade-off between precision and recall depends on the specific application. In autonomous driving, a high recall is generally preferred, as it is better to be on the side of caution and detect a potential object, even if it is a false positive, rather than miss a real object and therefore risk an accident.

6.3.2 mAP

In the computer vision domain, Mean Average Precision (mAP) is a commonly used metric to evaluate the performance of object detection. It provides a single number summary of the overall accuracy of the object detection system, taking into account both precision and recall [18].

mAP is calculated as the mean of the Average Precision (AP) scores for each class of objects being detected (e.g., *CAR, TRUCK, BICYCLE*, etc.). The AP score for each class is calculated as the average precision at different recall levels. Precision and recall are calculated as the fraction of true positive detections out of all positive detections and all ground-truth objects, respectively.

To calculate the AP score, the precision-recall curve is first constructed by plotting precision against recall for various threshold values used to determine which detections should

be considered true positive detections. The AP score is then calculated as the area under the precision-recall curve. The mAP score is then the mean of the AP scores for each class.

In the context of autonomous driving, the mAP score provides a comprehensive evaluation of the object detection system's performance in detecting different types of objects in real-world scenarios [48]. A high mAP score indicates that the system has a high accuracy in detecting objects, and that the precision and recall are well-balanced for each object class. This is important for ensuring the safety of passengers and other road users, as the autonomous vehicle must accurately detect and respond to all types of objects in its environment.

6.4 Experiments

All of the extensive experiments are executed on south1 camera field-of-view (on both detections and labels) and LiDAR-LiDAR early fusion registered point clouds and LiDAR-Camera late fusion. The configuration delivering the highest `mAP` score on this scenario and fusion is later taken for the evaluation of the next scenarios and fusions.

6.4.1 A9 south1 - Early Fusion LiDAR-LiDAR and Late Fusion LiDAR-Camera

For this experiment we take the supervised LiDAR Detector PointPillars in its Early Fusion setting for LiDAR-to-LiDAR Fusion - the registered point clouds. We are going to conduct all experiments on the matched and unmatched attribute fusion on the south1 camera field-of-view (FOV) filtered labels. The highest mAP achieving combination will then be used upon the other fusion approaches and FOVs.

Monocular and LiDAR only

Tables 6.1 and 6.2 include only the monocular and LiDAR detections from the respective detectors - MonoDet3D and LiDAR 3D Supervised PointPillars. As can be extracted from said tables, the LiDAR detections produce better results in almost all categories, when considering only mAP, with the exclusion of the class *BICYCLE*, where the Monocular detector performs slightly better. In terms of recall, an also important metric, Monocular produces much better results on the *BICYCLE* class and also marginally better on classes *CAR* and *MOTORCYCLE*. The 2D Object Detector, which the Mono3D Detector uses as its base, is trained exclusively on the MSCOCO Dataset and the results can be seen in the table. There are no detections for classes *TRAILER*, *VAN*, *EMERGENCY VEHICLE* and *OTHER*, as there are no such classes present in the aformentioned dataset.

Matched Monocular and LiDAR only

Tables 6.3 and 6.4 showcase the matched detections after late fusion of Camera-LiDAR, respectively with only monocular and only LiDAR attributes for the fused detections. Again, LiDAR performs almost always better, in terms of mAP, except on the class *BUS*. In terms of recall, monocular achieves higher results in both *BUS* and *BICYCLE* classes. Another thing to note is the occurrence of the predictions in comparison to the ground truth labels - there are only 1360 predictions and 1706 labeled objects. These are all the matched detections from the 1899 LiDAR and 2532 Monocular detections. This means that we are left with 1172 unmatched monocular and 539 unmatched LiDAR detections, respectively. As we cannot

Table 6.1: Monocular detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	1749/1034	56.72	66.70	56.46
TRUCK	370/203	38.51	40.26	38.01
TRAILER	0/132	0.00	0.00	0.00
VAN	0/70	0.00	0.00	0.00
MOTORCYCLE	38/31	75.51	75.14	75.02
BUS	32/33	**80.76**	70.15	**80.37**
PEDESTRIAN	148/131	4.09	4.28	3.80
BICYCLE	195/68	29.27	**80.91**	29.20
EMERGENCY VEHICLE	0/0	0.00	0.00	0.00
OTHER	0/4	0.00	0.00	0.00
mAP	2532/1706	28.49	33.74	28.29

Table 6.2: LiDAR detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	1219/1034	74.24	60.64	73.73
TRUCK	228/203	91.20	85.03	91.03
TRAILER	116/132	73.48	71.06	72.95
VAN	57/70	73.39	70.26	72.86
MOTORCYCLE	27/31	82.72	70.71	82.37
BUS	34/33	**99.93**	**100.00**	**99.93**
PEDESTRIAN	144/131	32.54	25.68	31.19
BICYCLE	72/68	28.04	14.99	26.60
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	1899/1706	58.10	50.47	57.47

fuse attributes in unmatched pairs, they do not produce value for this experiment and are therefore excluded.

Matched Monocular and LiDAR Attribute Fusion

In tables 6.5, 6.6, 6.7 and 6.8, we explore the fusion of the various attributes of the matched detections. Since the comparison made between LiDAR and Camera detections for matched displayed an overall better performance, in terms of mAP, for the LiDAR case, we are taking the LiDAR detection attributes as the baseline and only exchanging specific attributes for ones received from the monocular detections.

In table 6.5, we take all LiDAR attributes except the center position, which we take from the corresponding monocular detection. Unfortunately, the results are poor - 2.85 mAP less than taking LiDAR center positions. The same conclusion can be made for yaw angle (table 6.6), detection dimensions (table 6.7) and object category (table 6.8). LiDAR detections provide better results across the board. The smallest difference is between the yaw attributes from both modalities - 0.3 mAP, achieved thanks to the addition of the HD Maps.

Table 6.3: Matched detections with mono attributes with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	950/1034	48.66	43.72	48.17
TRUCK	193/203	35.57	33.01	34.99
TRAILER	0/132	0.00	0.00	0.00
VAN	0/70	0.00	0.00	0.00
MOTORCYCLE	34/31	75.23	**70.71**	74.95
BUS	24/33	**81.94**	70.15	**81.58**
PEDESTRIAN	69/131	8.50	4.28	7.88
BICYCLE	90/68	14.21	17.76	13.84
EMERGENCY VEHICLE	0/0	0.00	0.00	0.00
OTHER	0/4	0.00	0.00	0.00
mAP	1360/1706	26.41	23.96	26.14

Table 6.4: Matched detections with LiDAR attributes with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	984/1034	67.31	48.62	66.66
TRUCK	136/203	64.62	45.38	63.91
TRAILER	32/132	20.48	4.79	18.89
VAN	46/70	75.30	58.06	74.80
MOTORCYCLE	27/31	**82.72**	**70.71**	**82.37**
BUS	21/33	70.59	52.29	70.00
PEDESTRIAN	73/131	21.03	8.08	19.45
BICYCLE	39/68	32.43	14.42	31.08
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	1360/1706	45.98	30.87	45.12

Table 6.5: Matched detections with LiDAR attributes but mono center with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	984/1034	64.34	46.23	63.63
TRUCK	136/203	53.19	32.94	52.25
TRAILER	32/132	20.48	4.79	18.89
VAN	46/70	71.37	55.19	70.79
MOTORCYCLE	27/31	**82.72**	**70.71**	**82.37**
BUS	21/33	70.59	52.29	70.00
PEDESTRIAN	73/131	7.33	3.44	6.47
BICYCLE	39/68	35.60	16.35	34.31
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	1360/1706	43.11	29.13	42.27

Table 6.6: Matched detections with LiDAR attributes but mono yaw with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	984/1034	67.31	48.62	66.66
TRUCK	136/203	64.62	45.38	63.91
TRAILER	32/132	20.48	4.79	18.89
VAN	46/70	75.30	58.06	74.80
MOTORCYCLE	27/31	**82.72**	**70.71**	**82.37**
BUS	21/33	70.59	52.29	70.00
PEDESTRIAN	73/131	21.03	8.08	19.45
BICYCLE	39/68	29.54	13.15	28.13
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	1360/1706	45.71	30.75	44.82

Table 6.7: Matched detections with LiDAR attributes but mono dimensions with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	984/1034	67.31	48.62	66.66
TRUCK	136/203	64.63	45.80	63.93
TRAILER	32/132	7.72	1.96	6.75
VAN	46/70	73.06	55.19	72.52
MOTORCYCLE	27/31	**82.72**	**70.71**	**82.37**
BUS	21/33	68.63	48.29	68.00
PEDESTRIAN	73/131	25.02	10.43	23.52
BICYCLE	39/68	32.43	14.42	31.08
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	1360/1706	44.70	30.18	43.88

Table 6.8: Matched detections with LiDAR attributes but mono category with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	950/1034	64.20	46.69	63.48
TRUCK	193/203	55.51	41.97	54.62
TRAILER	0/132	0.00	0.00	0.00
VAN	0/70	0.00	0.00	0.00
MOTORCYCLE	34/31	**84.09**	**86.21**	**83.92**
BUS	24/33	77.91	63.91	77.47
PEDESTRIAN	69/131	21.35	8.31	19.78
BICYCLE	90/68	22.39	14.99	20.84
EMERGENCY VEHICLE	0/0	0.00	0.00	0.00
OTHER	0/4	0.00	0.00	0.00
mAP	1360/1706	32.55	26.21	32.01

Final Result

In table 6.9 we combine all the previous detections matched and unmatched together. For the matched we take only the configuration that produced the best results - namely only the LiDAR detections from the matched pairs. Additionally, we add all of the unmatched detections from both modalities. Unfortunately, the outcome of this fusion is worse in terms of mAP - -1.63 mAP less than only relying on the LiDAR Supervised detector.

Table 6.9: Best result achieved on south1 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points for labels 0.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	2018/1034	76.90	87.07	76.44
TRUCK	405/203	85.22	85.57	84.92
TRAILER	116/132	73.48	71.06	72.95
VAN	57/70	73.39	71.59	72.86
MOTORCYCLE	31/31	79.30	75.14	78.89
BUS	42/33	**96.31**	**100.00**	**96.31**
PEDESTRIAN	223/131	16.59	25.68	16.27
BICYCLE	177/68	35.86	77.85	35.75
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	3071/1706	56.25	60.00	55.84

Based on the first two tables (table 6.1 and table 6.2) we can assume that due to the higher recall achieved on both classes *CAR* and *BICYCLE* on monocular, these are the classes that would potentially improve the mAP in the final result. So for the unmatched monocular detections, the ones that do not have a matching LiDAR pair, we are going to take only these two classes and disregard the other ones. With this revision, we finally achieve a higher mAP, recall and precision than only from the supervised LiDAR PointPillars approach. The results can be seen in tables 6.10 and 6.11. The only difference between the outcome of both experiments, is the use of the parameter *num_points*, which serves as a filter for the LiDAR labels. By setting the parameter to 5, all LiDAR object labels with a number of points of less than 5 will be discarded. Thus the number of labels decreases from 1706 to 1667 (2.3% decrease). As is displayed, setting *num_points* to a higher value, results in worse performance, mostly due to the higher difference between detections and actual labels (68.2%) - translating to more false positives, negatively impacting the mAP.

Final Result - six classes only

Table 6.12, on the other hand, demonstrates the results achieved by only considering the six classes - *CAR*, *TRUCK*, *MOTORCYCLE*, *BUS*, *PEDESTRIAN* and *BICYCLE* - the classes contained as well in the MSCOCO dataset, the one that the object 2D detectors based yolov7 is pretrained on. As monocular could not identify the other four classes and LiDAR performed better on these six classes as well (6.2), the results are much better than by taking all ten of them.

6.4.2 A9 south2 - Early Fusion LiDAR-LiDAR and Late Fusion LiDAR-Camera

For the south2 camera Basler case, we can distinguish that the results are far below the ones from the south1 camera Basler detections. This is due to lighting conditions - lens flare and

Table 6.10: Best result achieved on south1 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points 0. For unmatched monocular detections we only take classes CAR and BICYCLE. All other classes lead to a mAP degeneration.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	2018/1034	76.90	87.07	76.44
TRUCK	228/203	91.20	85.03	91.03
TRAILER	116/132	73.48	71.06	72.95
VAN	57/70	73.39	70.26	72.86
MOTORCYCLE	27/31	82.72	70.71	82.37
BUS	34/33	**99.93**	**100.00**	**99.93**
PEDESTRIAN	144/131	32.54	25.68	31.19
BICYCLE	177/68	35.86	77.85	35.75
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	2803/1706	59.15	59.40	58.65

Table 6.11: Best result achieved on south1 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points 5. For unmatched monocular detections we only take classes CAR and BICYCLE. All other classes lead to a mAP degeneration.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	2018/1003	75.95	87.33	75.47
TRUCK	228/203	91.20	85.03	91.03
TRAILER	116/132	73.48	71.06	72.95
VAN	57/67	73.39	70.26	72.86
MOTORCYCLE	27/31	82.72	70.71	82.37
BUS	34/32	**99.93**	**100.00**	**99.93**
PEDESTRIAN	144/128	31.32	25.49	29.95
BICYCLE	177/67	36.56	80.77	36.47
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	2803/1667	59.00	59.70	58.50

Table 6.12: Best result achieved on south1 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points 0. For unmatched monocular detections we only take classes CAR and BICYCLE. Only the 6 classes that yolov7 trained on MSCOCO can distinguish:

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	2018/1034	76.90	87.07	76.44
TRUCK	228/203	91.20	85.03	91.03
MOTORCYCLE	27/31	82.72	70.71	82.37
BUS	34/33	**99.93**	**100.00**	**99.93**
PEDESTRIAN	144/131	32.54	25.68	31.19
BICYCLE	177/68	35.86	77.85	35.75
mAP	2803/1706	69.86	74.39	69.45

reflections that can be seen on Figure 6.1. Additionally most of the objects, detected by the monocular detector, are absent from the LiDAR label set, as they are too far back (Figure

7.7) and as such not identified by the LiDAR sensor and not labeled. Most notably, we also have far more obstructions than in the south1 case and therefore around 2.5x more occluded objects on average. The best results seen are on classes *VAN*, *MOTORCYCLE* and *BUS*. The class *CAR* performs extremely poorly compared to the outcome from the detector on south1. This is due to the 10x more detections than labels on cars - 10x more false positives than true positives.

Figure 6.1: South2 Basler camera lens flare and reflections.

Table 6.13: Best result achieved on south2 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points 5. For unmatched monocular detections we only take classes CAR and BICYCLE. All other classes lead to a mAP degeneration.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	2618/260	30.96	76.37	30.88
TRUCK	111/100	0.33	0.04	0.00
TRAILER	131/73	41.18	16.67	40.00
VAN	399/255	**96.04**	91.38	**95.96**
MOTORCYCLE	19/19	90.20	89.16	90.00
BUS	237/167	67.08	**92.65**	66.95
PEDESTRIAN	502/323	11.81	8.91	10.95
BICYCLE	19/24	0.74	0.31	0.50
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	0/0	0.00	0.00	0.00
mAP	4037/1221	33.83	37.55	33.52

6.4.3 A9 Full

Early Fusion LiDAR-LiDAR, Late Fusion Camera-Camera and Late Fusion Camera-LiDAR

For a9 full, we are evaluating the detections on all labels, without a field-of-view filtering on either labels or detections(table 6.14). In this case we are taking the early fusion approach for

LiDAR-to-LiDAR (the registered point clouds) and late fusion for Camera-to-Camera (south1 and south2) and Camera-LiDAR. We have a far greater number of detections and labels. The results are worse than those from only south1 labels as we have far more detections than labels (2.52x more), thus more false positives.

Table 6.14: Best result achieved on fused south1 and south2 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points 5. For unmatched monocular detections we only take classes CAR and BICYCLE. All other classes lead to a mAP degeneration.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	4538/1161	44.70	80.23	44.59
TRUCK	278/251	45.27	41.66	44.75
TRAILER	190/180	52.42	41.73	51.47
VAN	449/276	**70.17**	73.12	**69.58**
MOTORCYCLE	46/50	49.82	27.53	48.81
BUS	261/190	65.09	**90.20**	64.97
PEDESTRIAN	621/407	5.73	5.16	5.25
BICYCLE	193/91	25.07	43.96	24.81
EMERGENCY VEHICLE	1/0	0.00	0.00	0.00
OTHER	1/4	25.49	6.37	24.00
mAP	6578/2610	38.38	41.00	37.82

Late Fusion LiDAR-LiDAR, Late Fusion Camera-Camera and Late Fusion Camera-LiDAR

In this particular experiment, we take the late-fusion approach for the LiDAR-to-LiDAR fusion, but also for the Camera-to-Camera fusion of detections (south1 and south2, Camera-LiDAR fusion is always late fusion) - table 6.15. The results here are worse, as LiDAR-to-LiDAR late-fusion creates more predicted detections overall - 259 in total. As these are potentially not merged detections that are situated on the same object, we encounter an even higher number of false positives than the registered point clouds approach.

Table 6.15: Best result achieved on fused south1 and south2 camera, matched and unmatched detections with a 3D IoU Threshold of 0.1 for all 10 classes and a minimum number of points 5. For unmatched monocular detections we only take classes CAR and BICYCLE. All other classes lead to a mAP degeneration.

Class	Occurrence (pred/gt)	Precision	Recall	AP
CAR	4547/1161	42.41	80.04	42.30
TRUCK	332/251	63.85	74.13	63.12
TRAILER	264/180	53.15	68.50	52.22
VAN	573/276	46.96	61.11	46.63
MOTORCYCLE	46/50	25.58	23.53	25.04
BUS	223/190	**84.23**	**100.00**	**84.18**
PEDESTRIAN	683/407	5.78	5.65	5.41
BICYCLE	169/91	11.75	16.29	11.41
EMERGENCY VEHICLE	0/0	0.00	0.00	0.00
OTHER	0/4	0.00	0.00	0.00
mAP	6837/2610	33.37	42.93	33.03

Chapter 7

Results

7.1 Quantitative Results

The four object detection modules were evaluated on the A9 Infrastructure dataset with respective views - south1, south2 and full intersection test set (see Tables 6.10, 6.9). The performance of the module on the south1 sub set is around 80% higher on average, compared to south2, due to the absence of reflections (glare) and around 2.5x reduction in the number of obstructed objects. *LiDAR 3D Supervised* performs much better on the test set compared to *LiDAR 3D Unsupervised* since it is a data-driven approach, trained on the A9 infrastructure dataset - unsupervised achieves a `mAP` value of 8.13 on the A9 full set, whereas LiDAR 3D supervised scores a 47.42 `mAP` on its early fusion setting and is therefore excluded from the experiments and further evaluations.

By mixing different attributes from both modalities for the matched detections, we arrive at different `mAP` values. Only LiDAR attributes are considered in the final result, due to their superior performance (Table 5.1). This can be contributed to a possible merging error in the labels. As the *PointPillar's* approach is a data driven approach, it is susceptible to errors in the ground truth data. A further example of this case can be seen in Figure 5.11, where a road sign is identified as class *PEDESTRIAN*, as it is in the labels. This is possibly supporting the bad evaluation results on this class when regarding the monocular detections on south2.

Furthermore, the monocular detector scores poorly on the evaluation, as it cannot recognize certain classes. As the underlying model (YOLOv7 [65]) is trained on the MS COCO dataset [32], it can only identify classes that have been in the train set. This leads to non existent detection results on classes *TRAILER*, *VAN*, *EMERGENCY VEHICLE* and *OTHER*.

Additionally, there are no *EMERGENCY VEHICLES* in the labels. Although we can see a prediction of an *EMERGENCY VEHICLE*, it is considered as a false positive.

Using registered point clouds (early fusion LiDAR-to-LiDAR), only LiDAR attributes for matched detections and only classes *CAR* and *BYCICLE* for unmatched monocular detections, we achieve the highest results (69.45 `mAP`) with our Multi 3D fusion model on the A9 Infrastructure dataset south1 test set by fusing camera and LiDAR detections on a late fusion level. Table 6.10 demonstrates that among all classes, the *BUS* class exhibits the highest average precision in terms of detection accuracy, precision and recall, since it can be covered well by all LiDAR sensors due to its size - it is the largest object on average in the dataset (Tables 4.1 and 4.2). Excluding the *EMERGENCY VEHICLE* and the *OTHER* classes, from which there are not enough examples overall (6 in total in the test set, see Figure 4.3), the worst performing class is *PEDESTRIAN* - the smallest by dimensions object that can be recognized by the detectors (Tables 4.1 and 4.2).

7.2 Qualitative Results

In this section, we will discuss the visualization results from the mixed attributes fusion on both the south1 and south2 images from the test set. First are day scenes, followed by night scenes, occlusions, as well as, far away object detection and visualization. The results are compared in the following manner:

Left images: Matched and unmatched detections. Matched detections consist of only LiDAR attributes: *center position, yaw, dimensions* and *category* (as well as *UUID* and *score*).

Right images: Matched and unmatched detections. Matched detections consist of LiDAR attributes: *dimensions* and Monocular attributes: *center position* and *yaw* (as well as *UUID* taken from monocular and the mean average of *score* from both modalities).

The results show that although during evaluation the LiDAR center position and yaw vector proved to be more accurate in comparison the Monocular ones, when evaluated on the fused and filtered by field-of-view LiDAR labels, the visualization using the monocular center position and yaw demonstrate better accuracy.

Figure 7.1 shows the color mapping of the unmatched monocular, unmatched LiDAR and fused detections. It illustrates the core principle in the late fusion approach - the matched objects are only the intersection between the monocular and LiDAR detections. But since we are also visualizing the unmatched detections, there is a greater number of objects detected than by either of those modalities alone (including only the matched ones).

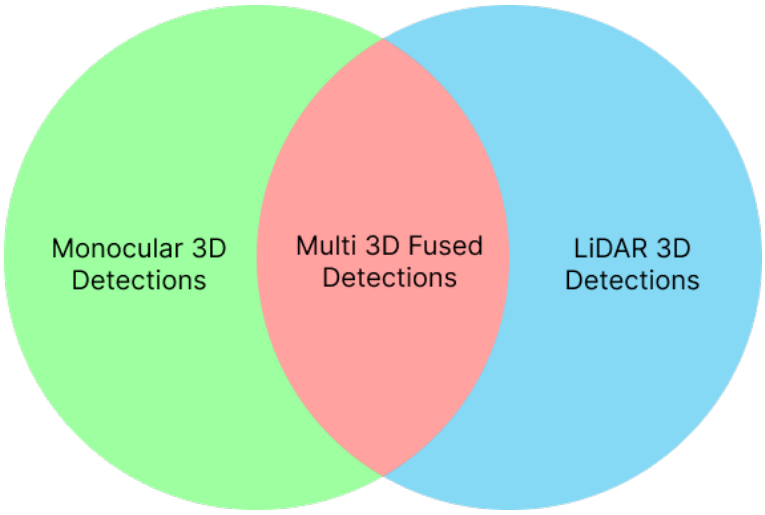

Figure 7.1: The correct colormapped Detections: (Left) Unmatched Monocular 3D Detections, (Center) Matched Multi 3D Fused Detections and (Right) Unmatched LiDAR 3D Detections. All of the detections that have been associated to a partner in the other modality get matched in the Multi 3D Fusion.

7.2.1 Day scenes

Following are a couple of day scenes, representing the majority of frames in the testset. Even here we can identify that the two cars in the middle of the image (or intersection) do not have the correct orientation, which the HD map from the monocular module provides. In terms of center position, we can easily distinguish that the monocular center position achieves better results - both on cars and on the bus, stopped at the traffic light. Whereas on the LiDAR

detection, the box blends with the van from the neighboring lane, the one provided from the Monocular 3D detector is situated quite well in terms of the actual position of the bus and fits neatly into the respective traffic lane.

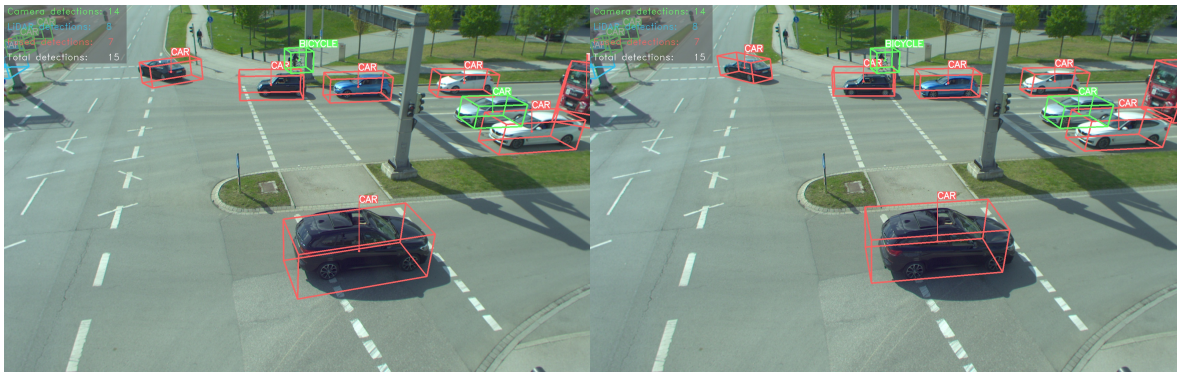

Figure 7.2: South1, Day scene, Left: LiDAR only attributes, Right: LiDAR dimensions, Camera center, yaw and category

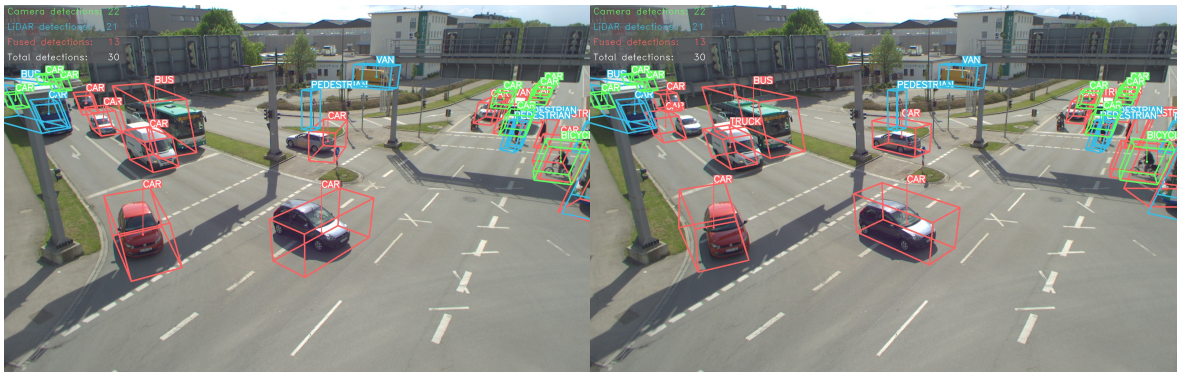

Figure 7.3: South2, Day scene, Left: LiDAR only attributes, Right: LiDAR dimensions, Camera center, yaw and category

7.2.2 Night Scenes

The night scenes provide a more interesting perspective. The LiDAR is able to detect here more objects on average than the camera - 75.26% more in total on the 37 night scene frames - a noticeable difference in comparison to the day scenes, where the camera was able to outperform the LiDAR sensor. The rotation and center position issue is also apparent here, but most noticeably, in south1 scene the LiDAR detector finds an obstacle that is not there - a false positive. This issue is not present in most of the other frames and almost entirely non existent in the south2 FoV. Additionally, the south2 scene, apart from the night and low lighting conditions, includes severe weather conditions - fairly strong rain and thus more reflections. But the LiDAR is still able to achieve a good result in this scenario and the multi 3D system, as a whole, can identify most, if not all, traffic participants, demonstrating robustness even in adverse weather conditions.

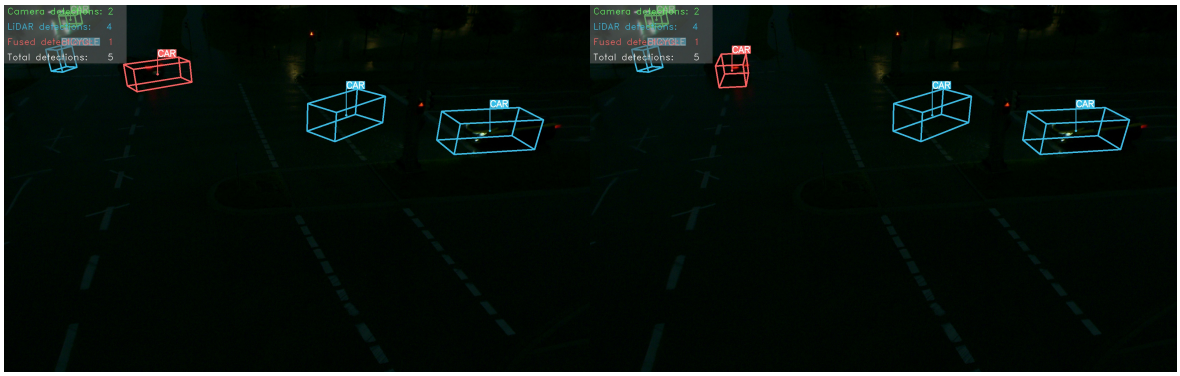

Figure 7.4: South1, Night scene, Left: LiDAR only attributes, Right: LiDAR dimensions, Camera center, yaw and category

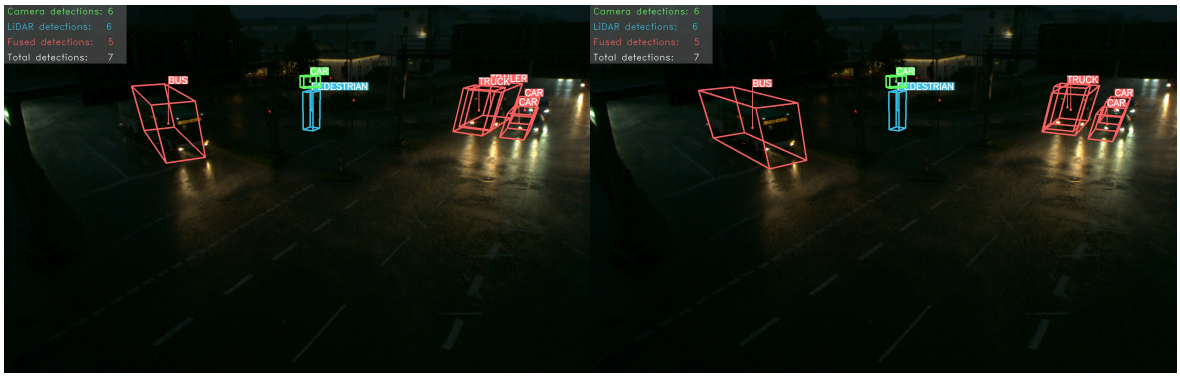

Figure 7.5: South2, Night scene, Left: LiDAR only attributes, Right: LiDAR dimensions, Camera center, yaw and category

7.2.3 Occlusions

In Figure 7.6 we show two examples of frames with plotted point clouds and occluded objects, not recognized by the camera sensor, but identified by the LiDAR sensor and thus from the Multi 3D detector. For visualization purposes we also added the point clouds under the camera frames, to surely position the objects.

During the day scene, we are able to distinguish a car behind the trailer of the truck passing by. On the image, only a small part of the cars roof is visible. Due to the obstruction, the car cannot be seen by the monocular 3D detector. The LiDAR sensors and detector, on the other hand, is easily able to distinguish the car and even plot its correct dimensions.

This is strengthened by the fact that we work with the registered point clouds in this scenario - both LiDARs help to create an abundant point cloud for the intersection. We gain a 41.67% increase in detections over only monocular.

During the night scene, the trailer cannot be identified by the monocular detector, due to the lighting conditions, as well as the road signs, positioned on the gantry bridge. The LiDAR 3D supervised detector, on the other hand, is able to identify and even correctly classify the trailer behind the gantry bridge.

7.2.4 Distant objects

In Figure 7.7 we can clearly distinguish that the monocular camera can detect really distant objects, that the LiDAR sensors cannot. There are no LiDAR labels there as well - potentially

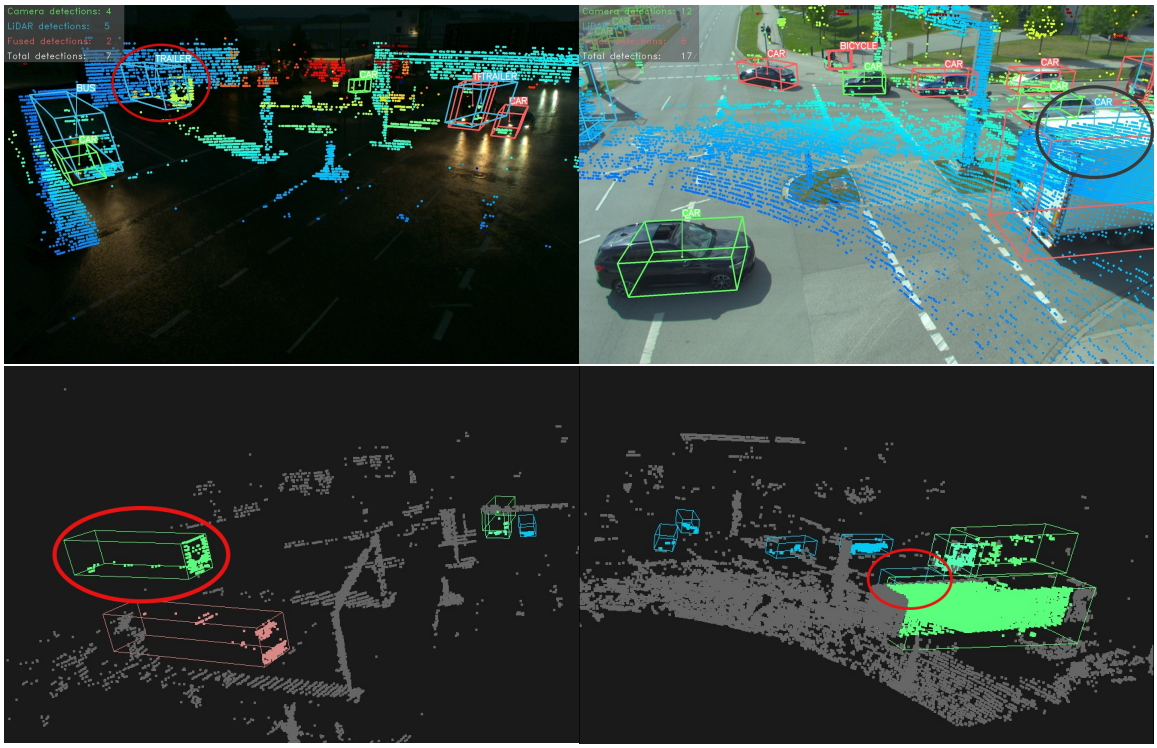

Figure 7.6: Early fusion of LiDAR detections (registered point clouds) and late fusion Camera-LiDAR. We register point clouds from two LiDARs using G-ICP [75] and illustrate them with the LiDAR detection results into the image. *Left column:* South2, Night and rain scene. *Right column:* South1, day scene.

one of the reasons that the monocular detector performs poorly during evaluation - it can detect objects that are not present in the LiDAR labels, which therefore get recognized as false positives.

Figure 7.7: South2, LiDAR dimensions, Camera center, yaw and category. Distant objects.

7.2.5 False category

An additional reason for poor evaluation results for the monocular detections when using monocular category is illustrated in Figures 7.8 and 7.9. Monocular detector often mistakes class *TRUCK* for other object classes.

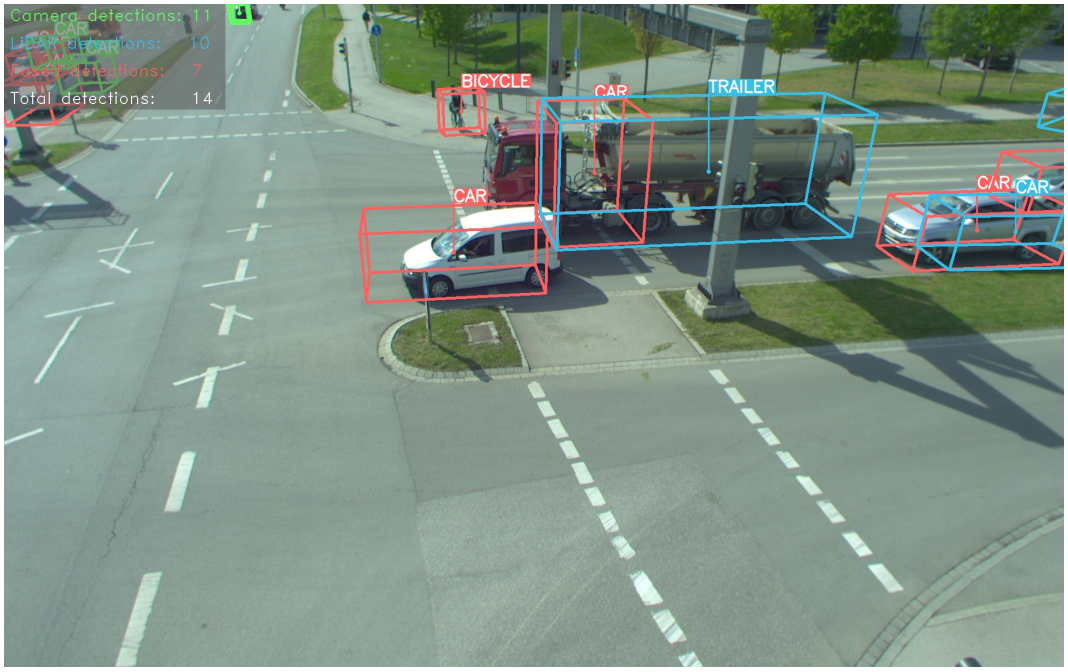

Figure 7.8: South1, LiDAR dimensions, Camera center, yaw and category. False category - the *TRUCK* is identified as class *CAR*.

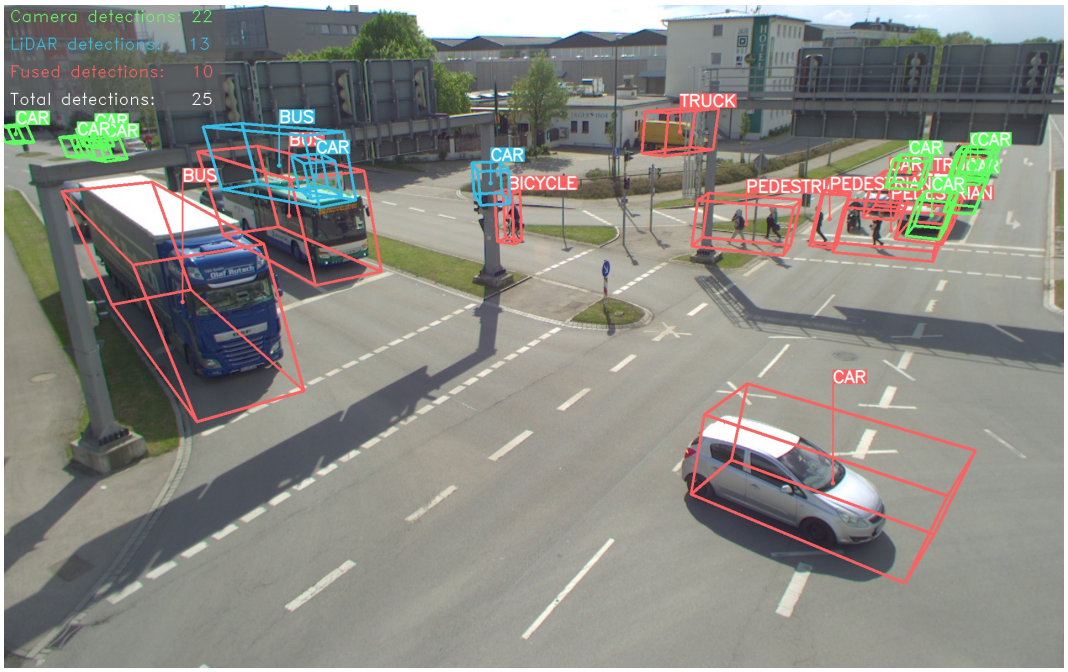

Figure 7.9: South2, LiDAR dimensions, Camera center, yaw and category. False category - the *TRUCK* is identified as class *BUS*.

Chapter 8

Future Work

8.1 Multi-Modal 3D Detector Optimization

For a future optimization of the demonstrated approach, there are multiple things to be considered and multiple augmentations possible, both on the late fusion model itself and its underlying models - Monocular 3D and LiDAR 3D Supervised.

One improvement would be to retrain from scratch or fine-tune the YOLOv7 [65] object 2D instance segmentation module using the A9 infrastructure dataset [11]. As it is currently trained on the MS COCO dataset [32], it does not correctly classify some of the classes, switching classes *BUS* with *TRUCK*, or even does not identify certain object categories all-together (*VAN*, *TRAILER*, *EMERGENCY VEHICLE* and *OTHER*). Another promising solution would be to imploy the newly developed YOLOv8 model [70] - retrain it on MS COCO and fine-tune on A9 dataset, or even train it from scratch on only the A9 dataset.

Regarding the LiDAR 3D supervised model, based on the *PointPillars* architecture [28], the model can be retrained using the whole DAIR-V2X-I dataset [71] and later fine-tuned on the R1 scenes of the A9 infrastructure dataset [11]. Using a larger training dataset should potentially improve the model performance.

Another problem that occurred during the evaluation process was the incorrect rotation of specific detection boxes on vehicles, especially when entering the intersection. This may be due to an initial fusion of LiDAR labels when the yaw angle was taken as the difference between the two yaw angles taken from the two LiDAR labels (LiDAR south and LiDAR north labels). As was mentioned before, labels from either sensor often provide a nearly 180 degree rotation, compared to one another - the orientations of the traffic participants was not considered during the labeling process.

The new approach proposed - to only take the label yaw angle (and other attributes) from the LiDAR sensor closest to each object, would solve this issue. Finally, we would need to retrain PointPillars again on the new and improved labels and should expect better (yaw) results.

For the multi-modal approach itself, one possibility is to adapt the HD Maps at the late fusion step and not during the monocular detection step, or even apply them two separate times, if not too computationally expensive. This would allow for preciser yaw angles during visualization and during evaluation.

8.2 Other Models

Another option is to take an entirely different approach - to train a different multi-modal network from scratch, that utilizes the features from both modalities.

Two such models are BEVFusion [34] and DeepInteraction [68]. These models score second and first, respectively, on the nuScenes leaderboard [45] and thus potentially capable of achieving even higher mAP on the infrastructure dataset.

By training these models on DAIR-V2X-I and later on fine-tuning those models on the A9 R1 infrastructure dataset, we would be able to compare their performance to our current late-fusion technique.

Chapter 9

Conclusion

Roadside sensors play a critical role in the development of smart cities and intelligent transportation systems and provide valuable data there, where the vehicle-based sensors cannot [69]. They are more accurate (without the constant need of calibration), are more cost efficient and grant a greater view of a traffic scene [3].

Two such sensors are camera and LiDAR. Both sensors contribute with a high spatial resolution and accurate object classification, as well as, edge estimation. However, they still have their limitations. LiDARs are expensive sensors that are sensitive to reflections and atmospheric scattering [37]. Cameras performance, on the other hand, can be impacted by the amount of visible light, severely degrading their capabilities during night scenes [26].

Therefore a fusion of both sensors can prove extremely beneficial. By complimenting each other, both modalities can provide more accurate and reliable data. Furthermore a late-fusion approach would contribute to one of the most daunting tasks in the autonomous driving domain - reliability, by producing enough redundancy in the data - less false negatives [15].

Many such fusion approaches have been proposed throughout the years (see Section 3.3). Most of them being data driven [1] [31] [34] [68]. Unfortunately, this process is often too computationally expensive to be able to run in real-time. Even when regarding current data association techniques, there are certain caveats in terms of computational complexity (see Section 3.5). Therefore a computationally inexpensive method would be of distinct value.

The proposed multi-modal 3D object detection package in this thesis is lightweight, with fast inference time, executed even on CPU, to a level that it does not impede the detection performance of the other detectors - monocular 3D and LiDAR supervised 3D. The results display the robustness of the given system and the model, in its current state, outperforms the only LiDAR approach - the 3D Supervised LiDAR Detector based on PointPillars [28].

During the quantitative and qualitative analysis, we have identified the key attributes that solidify each modality's strengths. Although, by taking the LiDAR attributes for the matched instances we gain better evaluation results, visualizations offer a different perspective. During the qualitative results, we identify key monocular attributes for the matched detections with persistent superior performance - detection center position and yaw angle. This could potentially be contributed to the false angles that the labels have gained after the merging process, a problem that can be solved by only taking the detection's yaw from the LiDAR sensor that is closest to said detection.

At the end we arrive at a distinct matched detections attribute combination that utilizes both LiDAR and camera detectors strengths - object category and dimensions from LiDAR, center position and yaw from monocular. Our system provides a good result, with an `mAP` of 69.45 on the south1 camera FoV of the A9 R1 infrastructure, intersection dataset, [11] includes more objects than either of the two modalities, considered separate, while maintaining real-time performance, with a minimal computational overhead.

List of Figures

List of Tables

Bibliography

[1] Bai, X., Hu, Z., Zhu, X., Huang, Q., Chen, Y., Fu, H., and Tai, C.-L. "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1090–1099.

[2] Buckland, M. and Gey, F. "The relationship between Recall and Precision". In: *Journal of the American Society for Information Science* 45.1 (1994), pp. 12–19. DOI: https://doi.org/10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L.

[3] Busch, S., Koetsier, C., Axmann, J., and Brenner, C. "LUMPI: The Leibniz University Multi-Perspective Intersection Dataset". In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, pp. 1127–1134.

[4] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[5] Carrillo, J. and Waslander, S. "Urbannet: Leveraging urban maps for long range 3d object detection". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 3799–3806.

[6] Castanedo, F. "A review of data fusion techniques". In: *The scientific world journal* 2013 (2013).

[7] Choi, H., Park, N.-C., and Kim, W.-C. "Optical system design for light detection and ranging with ultra-wide field-of-view using liquid lenses". In: *Microsystem Technologies* 26 (Jan. 2020). DOI: 10.1007/s00542-019-04490-4.

[8] Chongjian, Y., Liu, X., Hong, X., and Zhang, F. "Pixel-Level Extrinsic Self Calibration of High Resolution LiDAR and Camera in Targetless Environments". In: *IEEE Robotics and Automation Letters* PP (July 2021), pp. 1–1. DOI: 10.1109/LRA.2021.3098923.

[9] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[10] Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. "The cityscapes dataset". In: *CVPR Workshop on the Future of Datasets in Vision*. Vol. 2. sn. 2015.

[11] Creß, C., Zimmer, W., Strand, L., Lakshminarasimhan, V., Fortkord, M., Dai, S., and Knoll, A. *A9-Dataset: Multi-Sensor Infrastructure-Based Dataset for Mobility Research*. 2022. DOI: 10.48550/ARXIV.2204.06527. URL: https://arxiv.org/abs/2204.06527.

[12] Crouse, D. F. "On implementing 2D rectangular assignment algorithms". In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696. DOI: 10.1109/TAES.2016.140952.

[13] Derbel, A., Jemaa, Y. B., Canals, R., Emile, B., Treuillet, S., and Hamadou, A. B. "Comparative study between color texture and shape descriptors for multi-camera pedestrians identification". In: *2012 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE. 2012, pp. 313–318.

[14] Esfahani, M. A. and Wang, H. "Robust Glare Detection: Review, Analysis, and Dataset Release". In: *arXiv preprint arXiv:2110.06006* (2021).

[15] Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges". In: *IEEE Transactions on Intelligent Transportation Systems* 22.3 (Mar. 2021), pp. 1341–1360. DOI: 10.1109/tits.2020.2972974. URL: https://doi.org/10.1109%5C%2Ftits.2020.2972974.

[16] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. "Vision meets robotics: the KITTI dataset". In: *The International Journal of Robotics Research* 32 (Sept. 2013), pp. 1231–1237. DOI: 10.1177/0278364913491297.

[17] Hecht, J. "Lidar for self-driving cars". In: *Optics and Photonics News* 29.1 (2018), pp. 26–33.

[18] Henderson, P. and Ferrari, V. "End-to-End Training of Object Class Detectors for Mean Average Precision". In: *Computer Vision – ACCV 2016*. Ed. by Lai, S.-H., Lepetit, V., Nishino, K., and Sato, Y. Cham: Springer International Publishing, 2017, pp. 198–213. ISBN: 978-3-319-54193-8.

[19] Hoyle, B. and Xu, L. "Ultrasonic sensors". In: *Process Tomography: Principles, Techniques and Applications* (1995), pp. 119–149.

[20] Huang, X., Cheng, X., Geng, Q., Cao, B., Zhou, D., Wang, P., Lin, Y., and Yang, R. "The apolloscape dataset for autonomous driving". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 954–960.

[21] Huang, Y. and Chen, Y. "Autonomous driving with deep learning: A survey of state-of-art technologies". In: *arXiv preprint arXiv:2006.06091* (2020).

[22] *Intersection over Union*. https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Accessed: 2022-02-12.

[23] Jonker, R. and Volgenant, T. "A shortest augmenting path algorithm for dense and sparse linear assignment problems". In: *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*. Springer. 1988, pp. 622–622.

[24] *Kalman Filter*. https://en.wikipedia.org/wiki/Kalman_filter. Accessed: 2022-02-12.

[25] Kelemen, M., Virgala, I., Kelemenová, T., Miková, L., Frankovskỳ, P., Lipták, T., and Lörinc, M. "Distance measurement via using of ultrasonic sensor". In: *Journal of Automation and Control* 3.3 (2015), pp. 71–74.

[26] Krišto, M., Ivasic-Kos, M., and Pobar, M. "Thermal object detection in difficult weather conditions using YOLO". In: *IEEE access* 8 (2020), pp. 125459–125476.

[27] Kuhn, H. W. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[28] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. "Pointpillars: Fast encoders for object detection from point clouds". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12697–12705.

[29] Latha, N. A., Murthy, B. R., and Kumar, K. B. "Distance sensing with ultrasonic sensor and Arduino". In: *International Journal of Advance Research, Ideas and Innovations in Technology* 2.5 (2016), pp. 1–5.

[30] Lee, S., Ratan, R., and Park, T. "The voice makes the car: Enhancing autonomous vehicle perceptions and adoption intention through voice agent gender and style". In: *Multimodal Technologies and Interaction* 3.1 (2019), p. 20.

[31] Li, Y., Yu, A. W., Meng, T., Caine, B., Ngiam, J., Peng, D., Shen, J., Lu, Y., Zhou, D., Le, Q. V., et al. "Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17182–17191.

[32] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[33] Liu, Y., Wang, L., Su, Y., Zhang, Y., Wang, Y., and Wu, Z. "AlScN Piezoelectric MEMS Mirrors with Large Field of View for LiDAR Application". In: *Micromachines* 13.9 (2022), p. 1550.

[34] Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D., and Han, S. "BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation". In: *arXiv preprint arXiv:2205.13542* (2022).

[35] Maier, F. M., Makkapati, V. P., and Horn, M. "Environment perception simulation for radar stimulation in automated driving function testing". In: *e & i Elektrotechnik und Informationstechnik* 135 (Aug. 2018), pp. 309–315. DOI: 10.1007/s00502-018-0624-5. URL: https://doi.org/10.1007/s00502-018-0624-5.

[36] *Mathworks - Introduction to Lidar*. https://de.mathworks.com/help/lidar/ug/lidar-processing-overview.html. Accessed: 2022-02-07.

[37] McGill, M. J. and Starr, D. O. "Lidar Remote Sensing". In: (2002).

[38] *McKinsey - Reserve a seat, the future of mobility is arriving early*. https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/reserve-a-seat-the-future-of-mobility-is-arriving-early. Accessed: 2022-02-05.

[39] Mei, J., Zhu, A. Z., Yan, X., Yan, H., Qiao, S., Chen, L.-C., and Kretzschmar, H. "Waymo open dataset: Panoramic video panoptic segmentation". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIX*. Springer. 2022, pp. 53–72.

[40] *Meta Chart*. https://www.meta-chart.com/. Accessed: 2022-02-12.

[41] Mohammad, T. "Using ultrasonic and infrared sensors for distance measurement". In: *World academy of science, engineering and technology* 51 (2009), pp. 293–299.

[42] *Munkres implementation for Python*. https://software.clapper.org/munkres/. Accessed: 2022-02-07.

[43] *Non Maximum Suppression*. https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/. Accessed: 2022-02-12.

[44] Nowozin, S. "Optimal Decisions from Probabilistic Models: The Intersection-over-Union Case". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[45] *NuScenes Leaderboard*. https://www.nuscenes.org/object-detection?externalData=all&mapData=all&modalities=Any. Accessed: 2022-02-09.

[46] Patel, R., Levin, M. W., and Boyles, S. D. “Effects of autonomous vehicle behavior on arterial and freeway networks”. In: *Transportation Research Record* 2561.1 (2016), pp. 9–17.

[47] Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., Rus, D., and Ang Jr, M. H. “Perception, planning, control, and coordination for autonomous vehicles”. In: *Machines* 5.1 (2017), p. 6.

[48] Prabhakar, G., Kailath, B., Natarajan, S., and Kumar, R. “Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving”. In: *2017 IEEE region 10 symposium (TENSYMP)*. IEEE. 2017, pp. 1–6.

[49] *Providentia++, A9 Dataset*. https://innovation-mobility.com/en/project-providentia/a9-dataset/. Accessed: 2022-01-24.

[50] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. “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.

[51] Raitoharju, M. and Piché, R. “On computational complexity reduction methods for Kalman filter extensions”. In: *IEEE Aerospace and Electronic Systems Magazine* 34.10 (2019), pp. 2–19.

[52] Rezaei, M., Azarmi, M., and Mir, F. M. P. “Traffic-Net: 3D traffic monitoring using a single camera”. In: *arXiv preprint arXiv:2109.09165* (2021).

[53] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[54] *Ros Message Filters, ApproximateTimeSynchronizer*. http://wiki.ros.org/message_filters/ApproximateTime. Accessed: 2022-01-22.

[55] Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN”. In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[56] *Scipy Optimize Linear Sum Assignment*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html. Accessed: 2022-01-30.

[57] *Scipy Python API*. https://docs.scipy.org/doc/scipy/tutorial/general.html. Accessed: 2022-01-30.

[58] Shin, K., Kwon, Y. P., and Tomizuka, M. “Roarnet: A robust 3d object detection based on region approximation refinement”. In: *2019 IEEE intelligent vehicles symposium (IV)*. IEEE. 2019, pp. 2510–2515.

[59] *Stratified Sampling*. https://en.wikipedia.org/wiki/Stratified_sampling. Accessed: 2022-02-09.

[60] *StVZo*. https://de.wikipedia.org/wiki/Stra%C3%9Fenquerschnitt. Accessed: 2022-01-29.

[61] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. “Scalability in perception for autonomous driving: Waymo open dataset”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2446–2454.

[62] Team, O. D. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. https://github.com/open-mmlab/OpenPCDet. 2020.

[63] Torgo, L. and Ribeiro, R. "Precision and Recall for Regression". In: *Discovery Science*. Ed. by Gama, J., Costa, V. S., Jorge, A. M., and Brazdil, P. B. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 332–346. ISBN: 978-3-642-04747-3.

[64] Vora, S., Lang, A. H., Helou, B., and Beijbom, O. "Pointpainting: Sequential fusion for 3d object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4604–4612.

[65] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *arXiv preprint arXiv:2207.02696* (2022).

[66] Wang, C., Ma, C., Zhu, M., and Yang, X. "Pointaugmenting: Cross-modal augmentation for 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11794–11803.

[67] Wang, Y., Yang, B., Hu, R., Liang, M., and Urtasun, R. "Plumenet: Efficient 3d object detection from stereo images". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 3383–3390.

[68] Yang, Z., Chen, J., Miao, Z., Li, W., Zhu, X., and Zhang, L. "Deepinteraction: 3d object detection via modality interaction". In: *arXiv preprint arXiv:2208.11112* (2022).

[69] Ye, X., Shu, M., Li, H., Shi, Y., Li, Y., Wang, G., Tan, X., and Ding, E. "Rope3D: TheRoadside Perception Dataset for Autonomous Driving and Monocular 3D Object Detection Task". In: *arXiv preprint arXiv:2203.13608* (2022).

[70] *YOLOv8 Ultralytics*. https://docs.ultralytics.com/. Accessed: 2022-02-12.

[71] Yu, H., Luo, Y., Shu, M., Huo, Y., Yang, Z., Shi, Y., Guo, Z., Li, H., Hu, X., Yuan, J., et al. "Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 21361–21370.

[72] Zhang, T. and Jin, P. J. "Roadside LiDAR Vehicle Detection and Tracking Using Range and Intensity Background Subtraction". In: *Journal of Advanced Transportation* 2022 (Apr. 2022). Ed. by Armingol, J. M., pp. 1–14. DOI: 10.1155/2022/2771085. URL: https://doi.org/10.1155%5C%2F2022%5C%2F2771085.

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

[74] Zhaohua, L. and Bochao, G. "Radar Sensors in Automatic Driving Cars". In: *2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*. 2020, pp. 239–242. DOI: 10.1109/ICECTT50890.2020.00061.

[75] Zhou, Q.-Y., Park, J., and Koltun, V. "Fast global registration". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer. 2016, pp. 766–782.