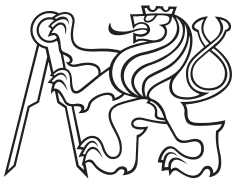
Bachelor Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

# Multimodal RGBD Object Detection for Autonomous Car

## Project JUPITER

**Tomáš Nevole**

**Supervisor: Ing. Vojtěch Šalanský, Ph.D.**
**Supervisor–specialist: doc. Ing. Karel Zimmermann, Ph.D.**
**Study program: Cybernetics and Robotics**
**May 2023**

# BACHELOR‘S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Nevole Tomáš** Personal ID number: **499143**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

## II. Bachelor’s thesis details

Bachelor’s thesis title in English:

**Multimodal RGBD Object Detection for Autonomous Car**

Bachelor’s thesis title in Czech:

**Detekce objektů v okolí autonomního auta v pomocí RGBD dat**

Guidelines:

Detection of the object in the car surrounding is an essential functionality for autonomous cars. It is a crucial input for the tasks such as adaptive cruise control or emergency braking. There is and development of autonomous car Jupiter in the Porsche Engineering which is based on ROS. The goal of this thesis is to develop and implement the algorithm which will project the depth information to the image plane and detect the objects (e.g. cars, pedestrians) using multimodal RGBD data.
(1) Study and describe the state-of-the-art object detection methods such as [1,2,3]
(2) Create a training data for the neural network using a free available dataset [4] by projection of the 3D lidar points to image plane.
(3) Implement and train the neural network that will use RGBD data as an input and detect the bounding boxes of the objects.
(4) Implement the detector as a ROS node for the Jupiter car (do the camera-lidar calibration if needed).
(5) Experimentally evaluate the results on free available dataset, ideally on benchmark. Show the advantage of using the depth modularity. Do the qualitative experiments showing how the detector works on the Jupiter car.

Bibliography / sources:

[1] Wu, Hai and Wen, Chenglu and Li, Wei and Li, Xin and Yang, Ruigang and Wang, Cheng, Transformation-Equivariant 3D Object Detection for Autonomous Driving, AAAI 2023
[2] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 779-788.
[3] Liu, W. et al. (2016). SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springe
[4] Geiger, Andreas & Lenz, P & Stiller, Christoph & Urtasun, Raquel. (2013). Vision meets robotics: the KITTI dataset. The International Journal of Robotics Research. 32. 1231-1237.

Name and workplace of bachelor’s thesis supervisor:

**Ing. Vojtěch Šalanský, Ph.D. Porsche Engineering Services, Prague**

Name and workplace of second bachelor’s thesis supervisor or consultant:

**doc. Ing. Karel Zimmermann, Ph.D. Vision for Robotics and Autonomous Systems FEE**

Date of bachelor’s thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

| Ing. Vojtěch Šalanský, Ph.D. | prof. Ing. Tomáš Svoboda, Ph.D. | prof. Mgr. Petr Páta, Ph.D. |
|---|---|---|
| Supervisor’s signature | Head of department’s signature | Dean’s signature |

## III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

# Acknowledgements

I am grateful for the support and guidance provided by my supervisor, Ing. Vojtěch Šalanský, PhD. I would also like to express my appreciation to Porsche Engineering for allowing me to work on the Jupiter project. Additionally, I extend my thanks to Ing. Tomáš Kozák and Ing. Jan Salášek for their guidance and for providing me with the data I needed for my thesis. Last but not least, I would like to thank my family and friends for their support during my studies.


# Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 23. May 2023

# Abstract

Object detection in the vehicle's surroundings is a key functionality for autonomous automobiles. It allows the car to see and interpret the surrounding environment and to provide input data for the systems, such as adaptive cruise control or emergency braking. Porsche Engineering is dedicated to developing an autonomous vehicle within the Jupiter project. The project utilises customised Porche Cayenne GTS, which is based on ROS. This thesis aims to extend an existing RGB architecture by depth data obtained by LiDAR. The detection network is trained on freely available datasets to achieve the best possible domain transfer to the Jupiter car. The advantage of the depth modularity is tested, and the quality of the model is evaluated on an available benchmark. The final detection model is implemented as a ROS node, which detects objects such as cars and pedestrians in real-time.



**Keywords:** Computer Vision, Autonomous mobility, LiDAR, object detection, ROS2

**Supervisor:** Ing. Vojtěch Šalanský, Ph.D.



# Abstrakt

Detekce objektů v okolí vozidla představuje klíčovou funkci pro autonomní automobily. Umožňuje vozidlu sledovat a interpretovat okolní svět a poskytnout tak vstupní data pro systémy jako například adaptivní tempomat nebo nouzové brzdění. Společnost Porsche Engineering se v rámci projektu Jupiter věnuje vývoji autonomního vozu. Projekt využívá speciálně upravené Porsche Cayenne GTS, které je možné řídit v rámci operačního systému ROS. Cílem této práce je rozšířit existující RGB architekturu o hloubková data získaná z LiDARu. Detekční síť je naučena na volně dostupných datasetech tak, aby bylo dosaženo co nejlepšího přenosu na auto z projektu Jupiter. Výhoda použití hloubkové modularity je otestována a kvalita detekčního modelu je ověřena na dostupném benchmarku. Výsledný detekční model je implementován jako ROS node, který v reálném čase detekuje objekty zájmu, např. vozidla a chodce.



**Klíčová slova:** Počítačové vidění, Autonomní mobilita, LiDAR, detekce objektů, ROS2

**Překlad názvu:** Detekce objektů v okolí autonomního auta pomocí RGBD dat — Project JUPITER

# Contents


**1 Introduction** 1

**2 State-of-the-art** 3

2.1 Race for autonomous car . . . . . 3

2.1.1 Introduction . . . . . 3

2.1.2 Waymo, the state-of-the-art driverless technology . . . . . 4

2.2 State-of-the-art detection model . 5

2.2.1 Introduction . . . . . 5

2.2.2 Real-time object detection . . . 6

2.2.3 YOLO - the state-of-the-art detection algorithm . . . . . 6

2.2.4 R-CNN . . . . . 7

2.2.5 SSD . . . . . 7

**3 Project JUPITER** 9

3.1 Introduction . . . . . 9

3.2 Car specifications . . . . . 9

**4 Theory** 13

4.1 Introduction . . . . . 13

4.2 Neural Networks . . . . . 13

4.2.1 Convolutional Layers . . . . . 14

4.2.2 Pooling Layers . . . . . 14

4.2.3 Activation Functions . . . . . 15

4.2.4 Fully Connected Layers . . . . . 16

4.2.5 Dropout Layers . . . . . 17

4.2.6 Batch Normalisation Layers . 17

4.2.7 Loss functions . . . . . 17

4.2.8 Optimiser . . . . . 19

4.3 Learning . . . . . 19

4.3.1 Updating model weights - backpropagation . . . . . 20

4.3.2 Validation . . . . . 20

4.3.3 Dataset . . . . . 20

4.3.4 Augmentation . . . . . 21

4.4 YOLO architecture . . . . . 21

4.4.1 YOLOv5 . . . . . 21
4.5 LiDAR . . . . . 23
4.6 Camera-LiDAR transformation . 24
4.7 Robotic Operating System 2 . . . 24
4.8 Dataset specification . . . . . 25
4.8.1 Kitti dataset . . . . . 25
4.8.2 Waymo open dataset . . . . . 26
4.8.3 Dataset comparison . . . . . 27

**5 Implementation 29**

5.1 Introduction . . . . . 29
5.2 Datasets . . . . . 29
5.2.1 Domain transfer . . . . . 30
5.3 Object detection . . . . . 30
5.3.1 Used model . . . . . 30
5.3.2 Model modifications . . . . . 31
5.3.3 Model training . . . . . 31
5.4 Camera-LiDAR calibration . . . . . 32
5.5 ROS Node . . . . . 33
5.5.1 Node overview . . . . . 34

**6 Experiments and Results 35**

6.1 Introduction . . . . . 35
6.2 Domain transfer . . . . . 35
6.3 Advantage of depth modularity . 36
6.4 Mask . . . . . 37
6.5 KITTI Benchmark . . . . . 38
6.6 ROS node detections . . . . . 40

**7 Conclusion 43**

**A 45**

A.1 Used codes . . . . . 45
A.2 Attached files . . . . . 45

**B Bibliography 47**

# Figures

3.1 There are currently 3 JUPITER cars. . . . . 10

3.2 JUPITER platform used in this thesis. This car is used by Porsche Engineering CZ. . . . . 10

3.3 Arrangement of the sensors with description. . . . . 11

3.4 Jupiter sensor set (schematic representation). . . . . 11

4.1 Example of Max-Pooling operation [GK20]. . . . . 15

4.2 Example of Average Pooling operation [GK20]. . . . . 15

4.3 The YOLO detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. [RDGF15]. . . . . 22

4.4 The network architecture of Yolov5. It consists of three parts: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head: Yolo Layer. The data are first input to CSPDarknet for feature extraction, and then fed to PANet for feature fusion. Finally, Yolo Layer outputs detection results (class, score, location, size) [XLL+21]. . . . . 23

4.5 Recording Platform. The VW Passat station wagon is equipped with four video cameras (two colour and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system [GLSU13]. . . . . 25

4.6 Sensor layout and coordinate systems [SKD+19]. . . . . 27

4.7 Example of images in the datasets. . . . . 28

5.1 Training overview. The first row shows in the first three graphs the training loss and in the following two the training mAP. The second row shows the same information for validation. . . . . 32

5.2 Projection of points before camera-LiDAR calibration. . . . . 33

5.3 Projection of points after camera-LiDAR calibration. . . . . 33

6.1 Comparison of models on Jupiter car data. KITTI-learned model achieved mAP of 0.558 and Waymo-learned model achieved mAP of 0.947. . . . . 36

6.2 Results on the KITTI 2d object benchmark dataset of the RGB (mAP 0.814) and RGBD (mAP 0.915) models on car detection. . . . 37

# Tables

4.1 Dataset comparison [SKD+19]. . 27

5.1 Selected YOLOv5 models with selected specifications [Joc20]. . . . . 31

5.2 Selected hyperparameters and their value. . . . . 32

5.3 Used message types with description. . . . . 34

6.1 Results of the RGB and RGBD models of detecting a car. . . . . 37

6.2 Results of the models with and without the mask of detecting a car. 38

6.3 Definition of detection difficulties. 39

6.4 Results of selected models on the hard part of the KITTI 2d object detection dataset. The models are ranked by the mAP. . . . . 40

6.3 Comparison of model with and without the mask. The mAP of the model with mask is 0.874 and without the mask is 0.915. . . . . 38

6.4 Results of the selected detection models. The model implemented in this thesis is highlighted. . . . . 39

6.5 Example of ROS Node detections. 40

6.6 Example of ROS Node detections. 41

# Chapter 1

# Introduction

A self-driving car has been a fascinating concept for generations, and it has the potential to revolutionize the automotive industry and transform transportation as we know it today. Driverless cars promise to improve mobility in various aspects. As we look ahead, advancements in autonomous mobility are expected to shape how we travel, work, and live, paving the way for a future where transportation is safer, more sustainable, and more accessible to all.

Many companies are taking the initiative to develop an autonomous car. Porsche Engineering[1] is an internationally active premium engineering service provider for automobile manufacturers, their suppliers, and other industries. As a Porsches daughter company, Porsche Engineering is developing its autonomous car. For this purpose, the company has a Porsche Cayenne GTS specially equipped with sensors such as a camera, lidar and radar and computing power. The vehicle uses a Robotic Operating System (ROS) to control the car. All sensors are integrated within the system and can be easily accessed in various applications - for example, the pursuit of this thesis - object detection.

Computer vision plays a pivotal role in the functionality of autonomous cars by enabling them to perceive and understand their environment through sensor-based information and data processing. It allows the vehicle to detect objects, recognize traffic signs, and understand road scenes, all that is necessary information regarding making decisions, ensuring safety, and interacting with humans. Computer vision also forms the foundation of the perception

[1] Porsche Engineering, `https://www.porscheengineering.com/peg/en/`

and decision-making capabilities essential for autonomous driving, making it a fundamental technology for advancing self-driving cars.

As mentioned above, Porsche Engineering is taking the initiative to develop its autonomous vehicle. To achieve this task, project JUPITER [HSPM22] was started. This project uses a customised Porsche Cayenne GTS as a development platform. Because the project is in the early stages, no object detection is implemented on the car. This thesis aims to implement a code to detect and classify vehicles and pedestrians. For the detection, data from the camera and LiDAR are used. As an object detection model, YOLOv5 by Ultralytics [Joc20] is used. The detection algorithm is implemented in a ROS node, which can be used within a Robotic Operating System 2 ROS) [MFG+22] system on a JUPITER car. This thesis also compares the results of the implemented model on an available benchmark.

The Chapter 2 presents and describes state-of-the-art methods in autonomous driving and object detection. Chapter 3 introduces the project Jupiter and the platform on which this thesis is implemented. Chapter 4 presents the theory needed to comprehend the implementation. The implementation is described in Chapter 5. Chapter 6 presents the experiments and results. The implemented model is tested on a free available dataset. The advantage of using depth modularity over sole RGB data and how the detector works on the Jupiter car are shown. The last chapter summarises the thesis and concludes the results.

# Chapter 2

# State-of-the-art

## 2.1 Race for autonomous car

### 2.1.1 Introduction

Self-driving cars have been researched and developed by many universities, research centres, car companies, and companies of other industries worldwide since the middle 1980s. One of the significant events that catalysed the race for the autonomous car was the DARPA Grand Challenge[1] that took place in 2004. Since then, autonomous mobility has evolved rapidly. With Technology Giants and automotive companies joining the initiative, the research accelerated. In 2009, Google initiated the development of self-driving vehicles, which in the year 2016 became a standalone company called Waymo[2]. In 2015 Tesla[3] announced the 'autopilot feature'. It used the fusion of data from cameras, radar and sonar technology. Another significant milestone was in the year 2018, when Nvidia launched a car chip called 'Xavier' with artificial intelligence (AI) capabilities and partnered with Volkswagen for the next generation of self-driving cars[4]. Today autonomous vehicles are being tested and deployed in various locations worldwide. Even though it may seem

[1] The Grand Challenge, https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles

[2] Waymo, https://waymo.com/

[3] Tesla, https://www.tesla.com/

[4] 10 Major Milestones in the History of Self-Driving Cars, https://blog.getmyparking.com/2019/09/25/10-major-milestones-in-the-history-of-self-driving-cars/

that the problem of driverless cars is solved, many obstacles are still yet to be overcome.

There are two main approaches to achieving driverless cars at this moment. The first one is executed by Tesla, using only camera information to determine the vehicle's surroundings. This method can have many drawbacks, including the disability to measure the distance of obstacles on the road or low-grade video quality under bad weather conditions. These issues can be solved by adding extra sensors such as lidar or radar. This approach is enforced by other companies such as Waymo, Volkswagen, Porsche and others. This technique uses the fusion of sensor data to make an image of the environment around the car. It simplifies determining the distance of the objects. It is also more robust under all conditions, i.e. in the rain or at night. This thesis uses depth data and images to detect the objects in front of the car. For this reason, the latter approach will be examined.

### 2.1.2 Waymo, the state-of-the-art driverless technology

Among many driverless cars, one stands out. Waymo is a company that makes driverless technology that can be installed on various platforms, such as Chrysler Pacifica or Jaguar I-PACE[5]. It started as a Google initiative in 2009[6]. Since then, significant progress has been made in all aspects of self-driving cars. Waymo has set the bar by developing its own technologies from the ground up. One of the most remarkable technologies Waymo has designed is custom lidar, which makes a 3D image of the car's surrounding environment.

To make a complete picture of the vehicle's surroundings, Waymo does not use only one camera or lidar. Many sensors are placed in selected positions around the car to complement each other. There are 19 cameras on Chrysler Pacifica and 29 on Jaguar I-PACE[7]. One is a front-facing long-range camera and a 360 vision system. A high-resolution 360° LIDAR is mounted on the top of the car.

The technology used by Waymo in developing autonomous vehicles is state-of-the-art[7], utilising a combination of advanced camera, lidar, and radar data to create a holistic view of the car's surroundings. What sets Waymo apart is that they have designed and built its custom sensors and developed various

[5]Waymo, `https://waymo.com/`
[6]Waymo - Our history, `https://waymo.com/company/#story`
[7]Waymo - Waymo Driver `https://waymo.com/waymo-driver/`

advanced algorithms to enable its autonomous vehicles to navigate safely and efficiently. With millions of miles on public roads and billions in simulation in more than 13 states in the United States[7], Waymo has accomplished a milestone in being the first autonomous ride-hailing service available to the public. The Waymo Driver stands as a beacon of safety and is leading the entire industry forward[7].

## 2.2 State-of-the-art detection model

### 2.2.1 Introduction

The advance in driverless vehicles would not be possible without computer vision. Computer Vision is a domain of Artificial Intelligence (AI) that uses computational models to interpret the data from cameras and (or) other sensors. Thanks to new technologies, more powerful hardware and an immense quantity of visual data available, Computer Vision has significantly improved over the past years. In less than a decade, today's systems have reached 99 per cent accuracy from 50 per cent, making them more accurate than humans at quickly reacting to visual inputs[8].

Since its origins, Computer Vision has reached many milestones. The progress can be separated into two periods. Before the year 2014, when traditional object-detecting methods were used, such as image processing. After 2014 came the Deep Learning period with object detection algorithms. To name a few: RCNN and SPPNet (2014), FastRCNN (2015), YOLO (2016) and RetinaNet (2017) [ZSGY19]. The most recent YOLO model is YOLOv8[9], released in January 2023.

Diverse computer vision applications include Classification, Object Detection and Instance Segmentation. Object Classification tells us what type of object is present in the picture. Object Detection is used to classify and localise the object in the image. Instance Segmentation determines what pixels of the image belong to an instance of an object. In this thesis, the Object Detection approach is used. Thus, the attention in this chapter is directed towards this approach.

[8] `https://towardsdatascience.com/everything-you-ever-wanted\-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e`

[9] Ultralytics, YOLOv8, `https://ultralytics.com/yolov8`

### 2.2.2 Real-time object detection

In general,object detectors need to balance speed and accuracy. Two-stage methods are generally more accurate (R-CNN [GDDM13], Mask R-CNN [HGDG17]). Faster but less accurate are one-stage detectors (YOLO [RDGF15], SSD [LAE+16], RetinaNet [LGG+20]). Two-stage architectures compose of two tasks. First, the object region proposal is followed by object classification in the proposed region. One-stage detectors skip the first part and predict the bounding boxes without suggesting the areas of interest. In this thesis, the focus is on one-stage models.

### 2.2.3 YOLO - the state-of-the-art detection algorithm

You Only Look Once (YOLO) is an open-source, state-of-the-art, real-time object detection algorithm that was first released in 2016 by Joseph Redmon *et al.* [RDGF15]. The newly presented architecture was significantly faster than any other object detector at the time. It posed the object detection task as a regression problem instead of a classification, as it was implemented in different algorithms.

YOLO was the first to utilise a single neural network to analyse the entire image in one go, rendering it highly efficient. This network divides the image into regions and simultaneously predicts bounding boxes and probabilities for each region[ZSGY19]. Despite its substantial improvement in detection speed, YOLO suffers from a drop in localisation accuracy compared with two-stage detectors, especially for some small objects[ZSGY19]. The follow-up YOLO models focused on this problem achieving better results.

Over the years, YOLO has undergone several versions, each improving upon its predecessor. The original YOLO algorithm (2016) was followed by YOLOv2, released in 2017 [RF17]. YOLOv2 improved upon the speed and accuracy of the original algorithm by introducing a few fundamental changes, such as multi-scale training and batch normalisation. Following YOLOv2, YOLOv3 was released in 2018 [RF18], which further improved object detection accuracy. The YOLOv4 algorithm [BWL20] was released in 2020 and employed more advanced architecture.

The most recent version of YOLO is the YOLOv7 [WBL22] detection algorithm. The YOLOv7 algorithm, released in 2021, is the fastest and most accurate version of YOLO yet. It uses a multi-scale approach that allows for

the detection of objects at different scales, improving detection accuracy even in challenging conditions. Compared to other neural networks, it demands several times cheaper hardware and can be trained much faster on small datasets without any pre-trained weights[10]. Overall, the YOLOv7 detection algorithm is a significant improvement over the previous versions of YOLO and is a testament to the continued development and innovation in the field of computer vision.

### 2.2.4 R-CNN

The R-CNN model, or Region-based Convolutional Neural Network, is a deep learning architecture used for object detection in images. It proposes regions in an image where objects may be located by selective search[ZSGY19]. Then it uses a convolutional neural network to predict the presence of an object in each proposed area and to classify them[USGS13]. R-CNN models were introduced in 2014. They presented significant progress at the time but still had plodding prediction speed, around 14s with GPU[ZSGY19].

### 2.2.5 SSD

SSD was proposed in 2015[LAE+16]. It stands for Single Shot MultiBox Detector and is known for its speed and accuracy in detecting objects in images and videos. The algorithm works by dividing the image into a grid of boxes, where each box is responsible for detecting objects within its boundaries. The SSD detector also uses multiple scales of feature maps to capture objects of varying dimensions, making it more robust in detecting objects of different sizes and orientations. SSD has advantages in terms of both detection speed and accuracy[ZSGY19].

[10]YOLOv7: The Most Powerful Object Detection Algorithm (2023 Guide), `https://viso.ai/deep-learning/yolov7-guide/`

# Chapter 3

# Project JUPITER

## 3.1 Introduction

The project JUPITER (Joint User Personalised Integrated Testing and Engineering Resource) is a project by Porsche Engineering. The project uses modified Porsche cars to develop future ADAS[1] systems [HSPM22]. The project aims to build one unified applied research and collaboration platform within Porsche Engineering. It is used for fast proof of concepts, for integration of close-to-series projects and integration of algorithms from partner locations or customers. The platform is in Fig. 3.1 and 3.2.

## 3.2 Car specifications

The model of the car used in this thesis is Porsche Cayenne GTS. The car is equipped with sensors such as a camera, LiDAR and radar. In this thesis, a camera and LiDAR are used. The camera is a ZED2 stereo camera, and the LiDAR is the Livox Lidar Horizon, with a range of up to 260 meters. Only front LiDAR is utilized. The sensors' arrangement is shown in Fig. 3.3, and the schematic representation of the sensors is in Fig. 3.4. The development platform uses a powerful Server-CPU with 512 GB of RAM and a high-performance GPU for AI-applications [HSPM22].

[1] ADAS stands for Advanced Driver Assistance Systems


**Figure 3.1:** There are currently 3 JUPITER cars.

**Figure 3.2:** JUPITER platform used in this thesis. This car is used by Porsche Engineering CZ.

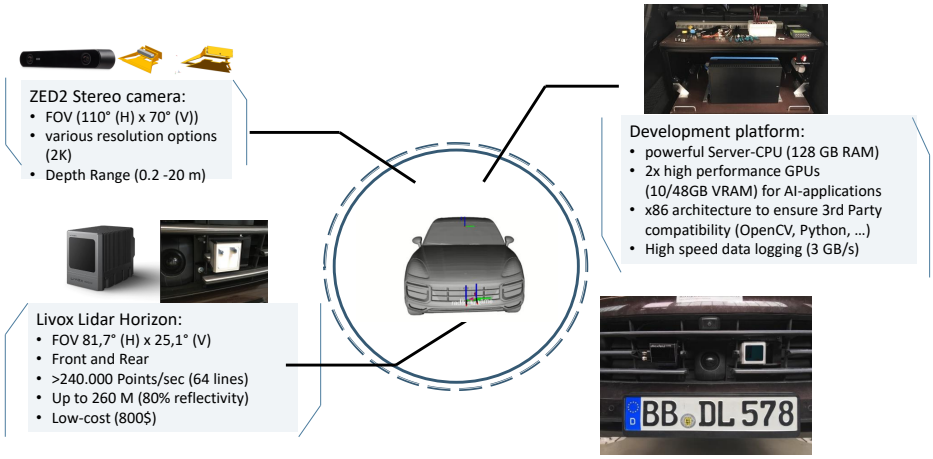



**Figure 3.3:** Arrangement of the sensors with description.

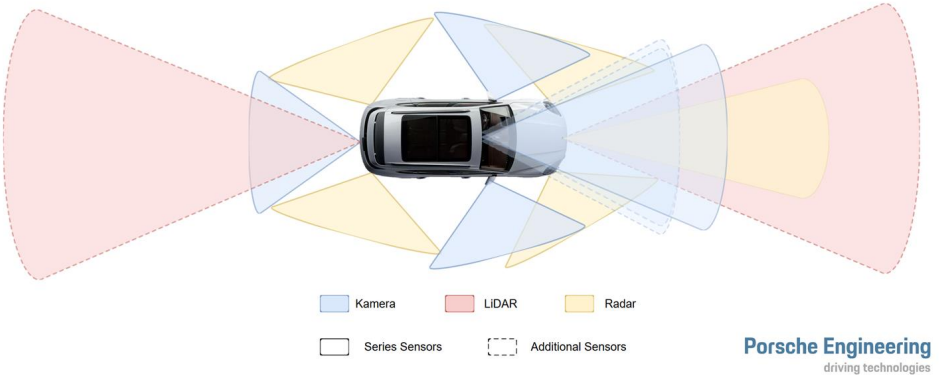



**Figure 3.4:** Jupiter sensor set (schematic representation).

# Chapter 4

# Theory

## 4.1 Introduction

This chapter provides the theory needed to understand the problematics of the implementation of object detection in the JUPITER car. First, I explain how real-time object detection works and describe the components of a Convolutional Neural Network (CNN) and how they work. Then I clarify the model training process. After that, I present the YOLO architecture. Then I illustrate how the LIDAR is used and explain the approach to using the provided data. Last but not least, I describe the working principle of ROS. Lastly, I present the KITTI and Waymo dataset, which are used to learn the network.

## 4.2 Neural Networks

A neural network is a type of machine learning model. There are various types of neural networks used for multiple purposes. The most used ones are Recurrent Neural Networks (RNN)[She20], Convolutional Neural Networks (CNN)[ON15], Long Short-Term Memory Networks (LSTM)[She20], and Generative Adversarial Networks (GAN)[GPAM+14]. In this thesis, the Convolutional Neural Network is used. This type of network is commonly used for image recognition tasks, such as image classification or object detection.

CNN compose of various layers, each serving a distinct purpose. These layers form a Hidden Layer, typically consisting of convolutional, pooling, fully connected, and normalisation layers with an activation function.

### 4.2.1 Convolutional Layers

Convolution is a foundational concept in computer vision. It extracts or creates a feature map from the input image. Inputs to convolution are two matrices. The first one, the input matrix, consists of pixels with constant values. It could be either an output from another layer or the input image itself. The second input is called the kernel. It is a smaller matrix with variable values used to help to extract essential features from the input matrix. It also contextualises the pixel within its location, which allows the network to learn the fundamental components such as edges, corners, and textures.

The convolution creates a feature map. The feature map values are calculated according to the equation (4.1)[1], where the input image is denoted by f and our kernel by h. The indexes of rows and columns of the result matrix are marked with m and n, respectively.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]. \tag{4.1}$$

### 4.2.2 Pooling Layers

As convolutional layers, the pooling layers [GK20] are one of the essential building blocks of convolutional neural networks. Convolutional layers produce a location-dependent feature map, and pooling layers provide translational invariance, so the CNN will detect the object even if the input of the CNN is translated. Commonly used are Max Pooling see Fig. 4.1 and Average Pooling see Fig. 4.2.

[1] Gentle Dive into Math Behind Convolutional Neural Networks, `https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional\-neural-networks-79a07dd44cf9`

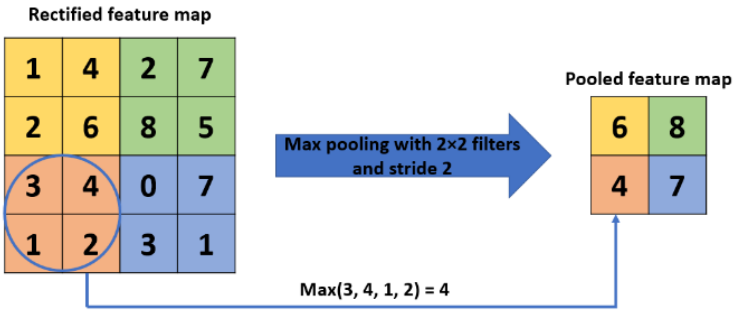



**Figure 4.1:** Example of Max-Pooling operation [GK20].

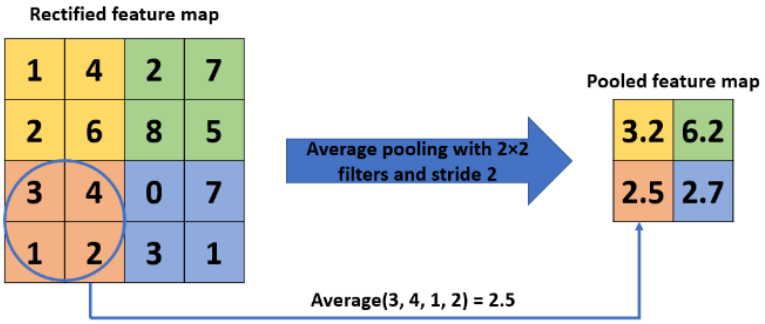



**Figure 4.2:** Example of Average Pooling operation [GK20].

### 4.2.3 Activation Functions

In the real world, many correlations and patterns are complex. In other words, many of the patterns are non-linear. To model the patterns, non-linear activation functions are used. Activation functions must be differentiable, zero-centred and easy to compute. Popular functions are, for example, sigmoid, Tanh or Leaky ReLU [DSC22].

#### Sigmoid

The equation (4.2) defines the sigmoid activation function. Because the output of this function is between 0 and 1, this function is used for the models that predict the probability.

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{4.2}$$

### Hyperbolic Tangent

The Tanh activation function is suitable for classification tasks because it can generate output values within the range of -1 and 1. It is a scaled and shifted version of the sigmoid function, which makes it symmetric around the origin. However, when used in deep neural networks, it can suffer from the vanishing gradient problem. The equation (4.3) describes the Tanh activation function.

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{4.3}$$

### Rectified Linear Unit - ReLU

ReLU is a popular activation function for neural networks. It is described by the equation (4.4). It helps mitigate the vanishing gradient problem in deep networks and enhances performance. The drawback of this function is that all negative values output zero, which decreases the model's ability to fit the data correctly.

$$f(x) = max(0, x). \tag{4.4}$$

### Leaky ReLu

Leaky ReLU is an activation function used in neural networks to address the "dying ReLU" problem mentioned before by introducing a small negative-value slope. This feature helps prevent the gradient of the activation function from becoming zero and improves the learning capability of the model. The equation (4.5) defines the leaky ReLU, where $a$ is a real number determining the slope.

$$f(x) = max(ax, x). \tag{4.5}$$

### 4.2.4 Fully Connected Layers

Fully Connected Layers form the last few layers in the network. The input is vectorised output of the convolutional or pooling layer. It is a linear layer that represents the function $y = Wx + b$, where $W$ represents the weights and $b$ is the bias. The output of the final layer is then used to determine the probability of individual object classes being present in the input data.

### 4.2.5 Dropout Layers

Dropout layers are used to avoid model overfitting. Overfitting occurs when the model learns the statistical noise of the data used for training. This phenomenon causes problems when the model is used for detection because the model fails to generalise. Dropout Layers are introduced to solve this problem as they randomly zeros out the value of some of the layer's neurons in every iteration.

### 4.2.6 Batch Normalisation Layers

The Batch Normalisation layer is typically placed between hidden layers. It normalises the output from a hidden layer and passes it as an input to the next hidden layer. Normalising the input data to a deep learning model is standard practice. The batch normalisation layer normalises the mean and variance of the input batch along each feature dimension and then scales and shifts the normalised values using learnable parameters, making it less sensitive to changes in the input data distribution. This technique can significantly improve the training process and the model's performance on the validation and test data [IS15].

### 4.2.7 Loss functions

Loss functions are one of the core aspects of neural networks directly responsible for the model fitting to the given data. They measure the deviation between the model's output and the ground truth training data. Based on this deviation, the weights in each layer are updated to minimise the value of the loss function. The loss function must be continuous, convex, differentiable, and appropriate for the specific task. The typically used functions in CNNs are as follows.

#### Cross-entropy loss

Cross-entropy loss is a common loss function used for classification tasks in CNNs. It measures the difference between the predicted probability

distribution and the actual probability distribution of the classes in the training data. The cross-entropy loss function is defined as

$$L_{CE} = -\sum_{i=0}^{n} t_i log(p_i), \tag{4.6}$$

for $n$ classes, where $t_i$ is the truth label and $p_i$ is the Softmax probability for $i^{th}$ class[2].

### Mean Squared Error (MSE)

MSE is a standard loss function used for regression tasks in CNNs. It measures the difference between the predicted output and the true output, squared and averaged over all samples in the training data. The MSE is defined as

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2, \tag{4.7}$$

where $N$ is the number of data points, $y_i$ is the output value of the model and $\hat{y}_i$ is the true value for data point $i$.

### Focal loss

The Focal loss function [ZS18] is designed to address the problem of class imbalance in classification tasks, where the number of samples in one class is much higher than in the other classes. It introduces a weighting factor to the cross-entropy loss to give higher importance to the samples in the minority class. For binary case, Focal loss is defined by the equation

$$Focal\,loss(p_t) = -(1-p_t)^{\gamma} log(p_t), \tag{4.8}$$

where $\gamma$ is a tunable parameter $\gamma \geq 0$ [ZS18] and $p_i$ is the model's estimated probability for the class defined as

$$p_t = \begin{cases} p & \text{if } y = 1, \\ 1-p & \text{otherwise.} \end{cases} \tag{4.9}$$

[2]Cross-Entropy Loss Function, `https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e`

#### BCE loss

The BCE loss function, short for Binary Cross-Entropy loss, is a commonly used loss function in machine learning for binary classification problems. It measures the difference between predicted probabilities and actual binary labels and penalises the model for incorrect predictions. The model learns to make more accurate predictions and improve its overall performance by minimising BCE loss. The formula[3] for BCE loss is

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i), \tag{4.10}$$

where $N$ denotes the number of samples, $y_i$ the true value and $\hat{y}_i$ the predicted value.

### 4.2.8 Optimiser

Optimisers are another essential component of neural networks. They are used during the model's training to find the optimal model weights. The most common optimisation technique is gradient descent [Rud16]. Many optimisers, such as Stochastic Gradient Descent (SGD), Adagrad[4], and Adam [KB14], are based on this approach.

## 4.3 Learning

Learning is a process during which the computer vision model is presented with annotated data. During this process model learns the internal parameters and adjusts the weights to make better and more correct detections of desired objects in given data.

[3] Understanding PyTorch Loss Functions: The Maths and Algorithms (Part 2) https://towardsdatascience.com/understanding-pytorch-loss-functions-the-maths-and-algorithms-part-2-104f19346425

[4] AdaGrad https://optimization.cbe.cornell.edu/index.php?title=AdaGrad

### 4.3.1 Updating model weights - backpropagation

The objective of the detection model is to make the best possible detections. To achieve this, the model has to go through the learning process. During each learning iteration - epoch, the loss function is calculated. The backpropagation aims to minimise the loss function by modifying the model weights. This algorithm computes the gradient with respect to all parameters and changes the parameters to minimise the loss function. The amount of the parameter change is determined by the gradient, which is multiplied by the learning rate. The learning rate is a hyperparameter set at the beginning and can be modified during the training.

### 4.3.2 Validation

At the end of every learning epoch, model performance is measured. This part of learning is called validation. The model is validated on a set of data that has not been presented while learning the weights to ensure that it can generalise well to new data. The model's ability to generalise to new data can be assessed by using a separate validation set, and any overfitting or underfitting can be detected early in the training process, allowing adjustments to be made to improve the model's performance.

### 4.3.3 Dataset

The dataset is an integral part of the learning process. It consists of labelled data, meaning there is a file describing the objects in the image for every image in the dataset. The optimal dataset contains all possible scenarios and objects in different conditions. The dataset used for learning the object detection model in an autonomous car should include various environments, objects, and light and weather conditions. The size of the dataset can vary on the task. Some datasets may contain only a few hundred images, while others may contain millions. For the learning process, datasets are split into three parts. The first part is the training part, used for training the model's weights. The second is used for validating the model. The validation set is a smaller subset of the dataset. It is essential to present the model data that has not been used while setting the weights to avoid overfitting and other undesired phenomena. The last subset of the dataset is used for testing to evaluate the final detection model. It assesses the model's performance on a representative data sample not used in the training phase. The dataset is

usually split in a ratio where 80% of images are used for training, 10% for validation and 10% for testing. Alternative ratios can also be used, i.e. 70% for testing and 15% for validating and testing, respectively.

### 4.3.4 Augmentation

The quality of the detection model is determined by how well the model can generalise. The better the model is, the better it can detect the objects under all conditions. The conditions are not just outer elements, such as the difference between day and night, clear sky and rain or fog. Different cameras will have slightly different parameters, which can cause a slight difference in image quality, colour accuracy, and other aspects of the picture. Data augmentation is used to make the model more robust and to increase the model's ability to withstand these conditions [PW17]. There are many techniques to augment the data, such as flipping the image, changing the colours of the image, resizing and skewing the image, and other more complex techniques, such as mosaic, which merges more pictures into one.

## 4.4 YOLO architecture

YOLO (You Only Look Once) [RDGF15] is a popular one-stage object detection algorithm that uses a single neural network to simultaneously predict object bounding boxes and class probabilities in an input image. YOLO divides the image into a grid and applies convolutional neural networks to each grid cell to generate a set of candidate bounding boxes, along with confidence scores and class probabilities. Non-maximum suppression is then used to remove duplicate boxes and select the most likely detections. This approach allows YOLO to achieve real-time performance while maintaining high accuracy, making it a popular choice for applications such as self-driving cars, robotics, and surveillance systems. The architecture scheme of the YOLO model is shown in Fig. 4.3.

### 4.4.1 YOLOv5

YOLOv5 [Joc20] is an advanced object detection algorithm introduced in 2020 by Ultralytics. It is a neural network-based approach that can detect

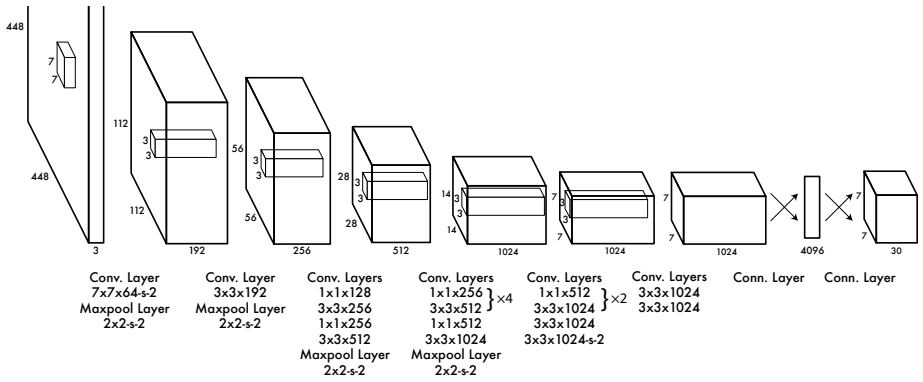



**Figure 4.3:** The YOLO detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating $1 \times 1$ convolutional layers reduce the features space from preceding layers. [RDGF15].

and classify multiple objects in an image with high accuracy and speed. At its release, YOLOv5 achieved top performances on two object detection datasets: Pascal VOC [EEG+14] and Microsoft COCO [LMB+14]. With its high accuracy and speed, YOLOv5 is being used in a variety of applications, including autonomous vehicles, surveillance systems, and robotics. Its ability to detect and classify objects in real-time makes it a valuable tool for many industries and this thesis. The network architecture (Fig. 4.4) incorporates cross-stage partial network (CSPNet) [WLY+19] into Darknet, creating CSP-Darknet53 as its backbone. CSPNet tackles the issue of duplicate gradient data in extensive backbones. It incorporates the changes in gradient into the feature map, decreasing the model's parameters and FLOPS (floating-point operations per second). This feature efficiently enhances the inference speed and accuracy while reducing the model size. YOLOv5 also involves a path aggregation network (PANet) [WLZ+19] as its neck to increase information flow. The PANet system includes a new feature pyramid network (FPN) structure that enhances the low-level features' propagation through an improved bottom-up path. Further, adaptive feature pooling links the feature grid and all levels to ensure that valuable information propagates directly to the following subnetwork. By improving the utilisation of accurate localisation signals in lower layers, PANet significantly enhances object location accuracy. Finally, the Yolo layer of YOLOv5 generates three different feature map sizes (18 x 18, 36 x 36, 72 x 72) to enable multi-scale [RF18] prediction and handle small, medium, and large objects.

The YOLOv5 loss consists of three parts. Class loss (BCE loss), objectness loss (BCE loss) and location loss (CIoU loss) [Joc23]. The final loss [Joc23] is computed as

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}. \tag{4.11}$$

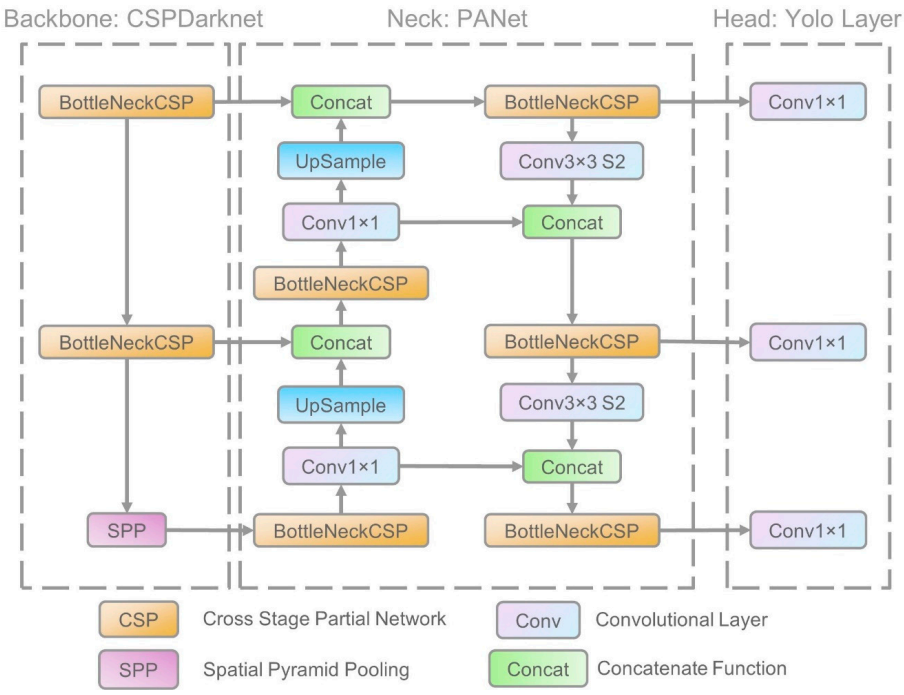



**Figure 4.4:** The network architecture of Yolov5. It consists of three parts: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head: Yolo Layer. The data are first input to CSPDarknet for feature extraction, and then fed to PANet for feature fusion. Finally, Yolo Layer outputs detection results (class, score, location, size) [XLL+21].

## 4.5 LiDAR

LiDAR, which stands for Light Detection and Ranging, is a technology that uses laser light to create highly detailed 3D maps or point clouds of objects and surfaces from a distance. It does this by emitting short pulses of laser light and measuring the time it takes for the light to bounce back after hitting an object. By repeating this process many times per second and combining the measurements, LiDAR can create a comprehensive 3D representation of the environment. In autonomous vehicles, LiDAR is a critical obstacle detection and localisation component, providing accurate and real-time 3D measurements of the surrounding environment.

## 4.6 Camera-LiDAR transformation

Converting 3D points from LiDAR into a camera frame is essential in this thesis. This process involves adjusting the position, orientation, and scale of a set of points in 3D space. To transform the points from one coordinate system to another, various types of transformations, such as translation, rotation, and scaling, are applied. These transformations can be performed individually or combined to create more complex transformations. One efficient method of transforming 3D points from LiDAR is by using matrices. A matrix can represent each transformation, and when multiplied together, they can be used to apply the entire transformation to a set of points [Sri16]. The intrinsic camera matrix[5] represents a camera's internal parameters, such as focal length, image sensor size, and principal point. This process converts 3D world points into 2D image points, projecting the points on the image plane.

## 4.7 Robotic Operating System 2

ROS 2[6] [MFG+22] is a collection of open-source software libraries and tools for robotic application development. It provides a comprehensive range of tools for building complex and sophisticated robot systems. ROS 2 represents a significant upgrade from its predecessor, the Robotic Operating System 1, and offers several new features, such as improved modularity, enhanced security, and better real-time performance.

The communication between ROS nodes is based on topics[7]. Nodes, called publishers, publish specific messages that contain data. These messages are published on a particular topic. A subscriber node must be created to access the data in a code. This node subscribes to the topic and can receive the messages and access the data.

[5] Camera matrix , https://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf

[6] ROS 2 Huble, https://docs.ros.org/en/humble/index.html

[7] ROS 2, Understanding topics, https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html

## 4.8 Dataset specification

### 4.8.1 Kitti dataset

The KITTI raw dataset [GLSU13] is a popular computer vision benchmark dataset for object detection, tracking, scene understanding, and other tasks. It was created in 2012 by the Karlsruhe Institute of Technology and the Toyota Technological Institute in Chicago. The dataset has been recorded in and around Karlsruhe in Germany. It includes camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system.

#### Car specification

The dataset was recorded from a moving platform, Volkswagen Passat, equipped with four video cameras, a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system. The recording platform is in Fig. 4.5.

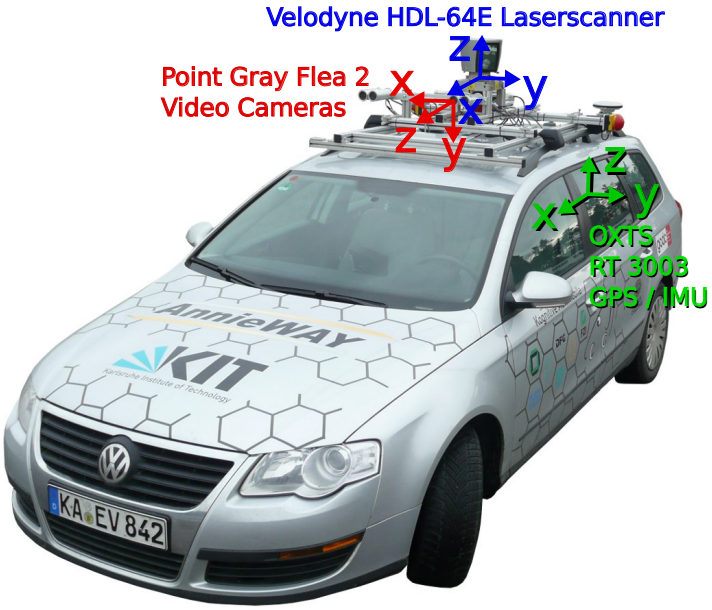



**Figure 4.5:** Recording Platform. The VW Passat station wagon is equipped with four video cameras (two colour and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system [GLSU13].

In this thesis, I used only the following sensors. The RGB data are recorded with the PointGray Flea2 colour camera (FL2-14S3C-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter. Only the data from the left camera were used. The lidar data were obtained from a Velodyne HDL-64E rotating 3D laser scanner, 10 Hz, 64 beams, 0.09-degree angular resolution with 2 cm distance accuracy, collecting approximately 1.3 million points/second, field of view: 360° horizontal, 26.8° vertical and range: 120 m [GLSU13].

#### Dataset specification

The KITTI dataset contains recordings from various environments, such as roads, cities and residential areas. The dataset was recorded on the 26th, 28th, 28th, and 30th of September and on the 3rd of October 2011 during the daytime. The dataset consists of camera images, Velodyne scans and OXTS (GPS/IMU), which are not used in this thesis. For each sequence, a calibration file is provided. For a subset of the dataset, 3D tracklet annotations are provided. The annotations had to be modified for this thesis purpose to produce 2D bounding boxes and labels for the images [GLSU13].

### 4.8.2 Waymo open dataset

The Waymo Open Dataset [SKD+19] is composed of two datasets - the perception dataset with high-resolution sensor data and labels for 2,030 segments and the motion dataset with object trajectories and corresponding 3D maps for 103,354 segments. It was collected by Waymo's autonomous vehicles. The segments are recorded in a variety of driving conditions, such as daytime and nighttime, urban and suburban areas, and different weather conditions.

#### Sensor specifications

The dataset was captured with five LiDAR sensors and five high-resolution pinhole cameras. The layout of the dataset relevant sensors is in Figure 4.6. The range of the LiDAR data is restricted to 75 meters (TOP sensor) and 20 meters (other sensors). The images from all cameras are downsampled and cropped from the raw images to the resolution of 1920x1280.

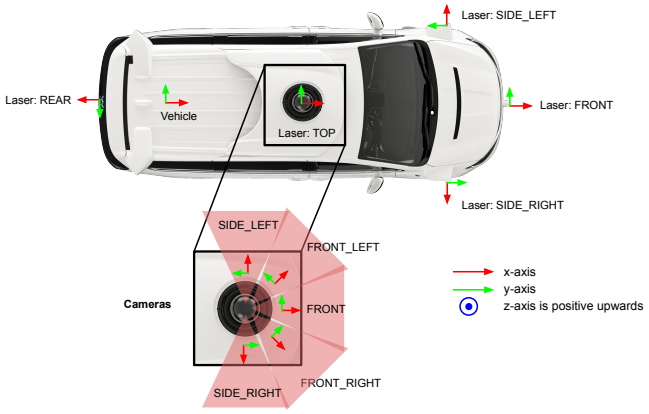



**Figure 4.6:** Sensor layout and coordinate systems [SKD+19].

| | KITTI | Waymo |
|---|---|---|
| Scenes | 22 | 1150 |
| Ann. Lidar Fr. | 15K | 230K |
| Hours | 1.5 | 6.4 |
| 2D Boxes | 80K | 9.9M |

**Table 4.1:** Dataset comparison [SKD+19].

## Dataset specification

The Waymo open dataset is the largest and most diverse multimodal autonomous driving dataset. It contains images recorded by multiple high-resolution cameras and sensor readings from high-quality LiDAR scanners mounted on a fleet of self-driving vehicles [SKD+19]. The recordings were captured in many cities in the United States and various conditions, such as daylight, night, rain, fog and others. The dataset contains around 12 million camera box annotations, which were manually labelled and reviewed.

### 4.8.3 Dataset comparison

The Scalability in Perception for Autonomous Driving: Waymo Open Dataset [SKD+19] article provides a comparison of the abovementioned datasets. Table 4.1 provides a shortened version of this comparison. The Waymo open dataset is significantly larger in all mentioned aspects. The Waymo dataset also contains images with different weather and lighting conditions. Example of selected images from the datasets is in Fig. 4.7.


**Figure 4.7:** Example of images in the datasets.

# Chapter 5

# Implementation

## 5.1 Introduction

In this chapter, an insight on implementation will be provided. The implementation is divided into two separate parts. The first part is the object detection model, which inputs RGBD data and outputs an image with detected bounding boxes and object classes. The second part implements a ROS node structure to execute the task within the robotic operating system. This node subscribes to a camera image and lidar point cloud, pre-processes them for the detection model, and after the detection is done, it publishes the image with detections.

## 5.2 Datasets

For this thesis, the used datasets had to be modified. The final structure of the dataset used for training the model consisted of four folders. The first one contains images. Original images from the datasets were used without further modification. The second folder contains lidar files. Lidar points from the files were transformed using provided matrices from calibration files. The transformation was from a lidar coordinate system to a camera image. Then the lidar points were linearly interpolated to fill in missing measurements and

saved as a numpy[1] array. The third folder contains mask files. The mask is a numpy array representing the positions of the measured distances by lidar before interpolation. The files were saved as a numpy array. The last folder contains the label files. The labels were defined in different formats in both KITTI and Waymo datasets. The labels had to be processed to correspond with the format required by the YOLOv5 model by Ultralytics. The labels were stored as .txt files.

### 5.2.1 Domain transfer

Domain transfer is a crucial aspect of this thesis. The model that has been learned must be able to function with the Jupiter car's different camera and lidar. The transfer to Jupiter data was tested after the model was trained on the KITTI dataset. Upon reviewing the results, I was not satisfied with the outcome. To improve the model, I decided to search for alternative datasets. The one that stood out was the Waymo open dataset. This dataset is described in chapter 4.8.2. The final model was learned on the Waymo dataset, as it achieved better results on the data from the Jupiter car. The results are discussed in chapter 6.2.

## 5.3 Object detection

Object detection is the core of this thesis. In order to detect objects, a model must first be trained. In this thesis, a YOLOv5 model [Joc20] implemented by Ultralytics[2] was used. This model was released in June 2020 and outperformed other detection models such as Faster-RCNN, SSD and YOLO models [TNJ+21]. The model was trained to detect two classes - Car and Pedestrian.

### 5.3.1 Used model

Ultralytics provide various sizes of the YOLOv5 model. A comparison of models' parameters is in Table 5.1[3]. For this thesis, the YOLOv5l model was

[1] Numpy, `https://numpy.org/doc/stable/index.html`
[2] Ultralytics, `https://ultralytics.com/`
[3] Ultralytics YOLOv5, `https://github.com/ultralytics/yolov5#pretrained-checkpoints`

trained.

| Model | params (M) | FLOPs (B) |
|---|---|---|
| YOLOv5n | 1.9 | 4.5 |
| YOLOv5s | 7.2 | 16.5 |
| YOLOv5m | 21.2 | 49.0 |
| YOLOv5l | 46.5 | 109.1 |
| YOLOv5x | 86.7 | 205.7 |
| YOLOv5x6 | 140.7 | 209.8 |

**Table 5.1:** Selected YOLOv5 models with selected specifications [Joc20].

### 5.3.2 Model modifications

The YOLOv5 model by Ultralytics is implemented to work with RGB images. To implement RGBD detection, certain modifications in the code had to be made. The model was changed to work with five-channel data. Three channels are the image, the fourth channel is the interpolated depth, and the last channel is the mask, which contains information about the locations of lidar measurements.

First, I created new functions to load the lidar files and the masks. I took the existing function that loaded the images and modified it for a different data type. The modifications were done to the files *train.py*, *val.py* and *utils/dataloaders.py*.

Another modification had to be done to data augmentations (*utils/augmentations.py*). The functions that altered the image had to be modified to separate it from the lidar and mask, as the input data is a 5-dimensional array. However, there was no need to alter other functions, such as random perspective.

### 5.3.3 Model training

The models were trained using 4 GPUs, NVIDIA A100 40 GB[4]. The final models were trained for 30 epochs. The batch size for the models trained on the KITTI dataset was 16, while the model trained on the Waymo dataset had a batch size of 8. These batch sizes were chosen due to the GPU memory limitations. The selected hyperparameters are presented in Table

[4]NVIDIA A100, `https://www.nvidia.com/en-us/data-center/a100/`

5.2. Following augmentations were used. HSV, Flip, random perspective and from Albumentations[5] functions *RandomResizedCrop*, *Blur*, *MedianBlur*, *ToGray*, *CLAHE*, *RandomBrightnessContrast* and *RandomGamma* were employed.

During the training, the loss functions and the mAP on the validation set were observed to determine the best model's weights. The plot of these parameters is in Fig. 5.1.

| Hyperparameter | Value |
|---|---|
| Learning rate | 0.01 |
| Momentum | 0.937 |
| Weight decay | 0.0005 |
| Warmup epochs | 3 |
| Warmup momentum | 0.8 |

**Table 5.2:** Selected hyperparameters and their value.

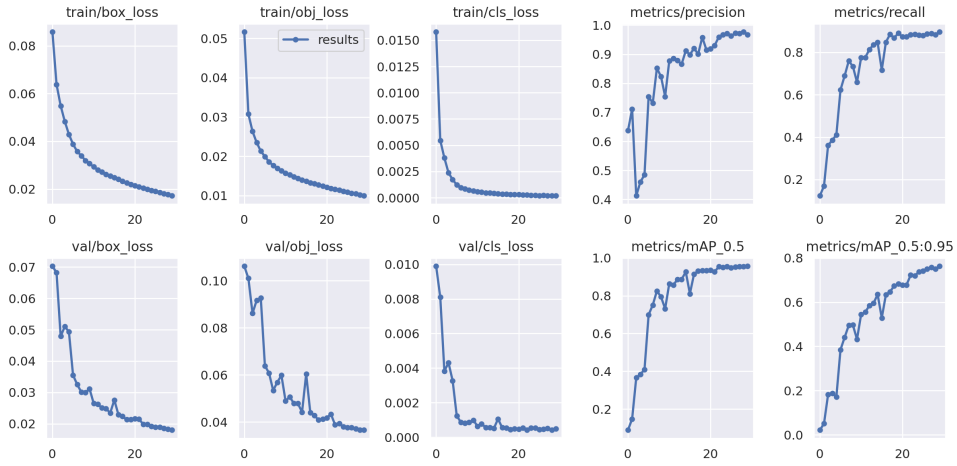



**Figure 5.1:** Training overview. The first row shows in the first three graphs the training loss and in the following two the training mAP. The second row shows the same information for validation.

## 5.4 Camera-LiDAR calibration

The transformation matrix from the LiDAR frame to the camera frame was obtained by manually measuring the distances on the car. This method provided a rough estimation of the matrix values. To this translation matrix,

[5] Albumentations, `https://albumentations.ai/`

a rotation matrix was added. The camera's tilt angle was measured, and based on this information, a rotation matrix around the $y$ axis was computed and then experimentally adjusted. Another rotation was needed because the lidar slightly offsets the $z$ axis. The results of this calibration are presented in Fig. 5.2 and Fig. 5.3.

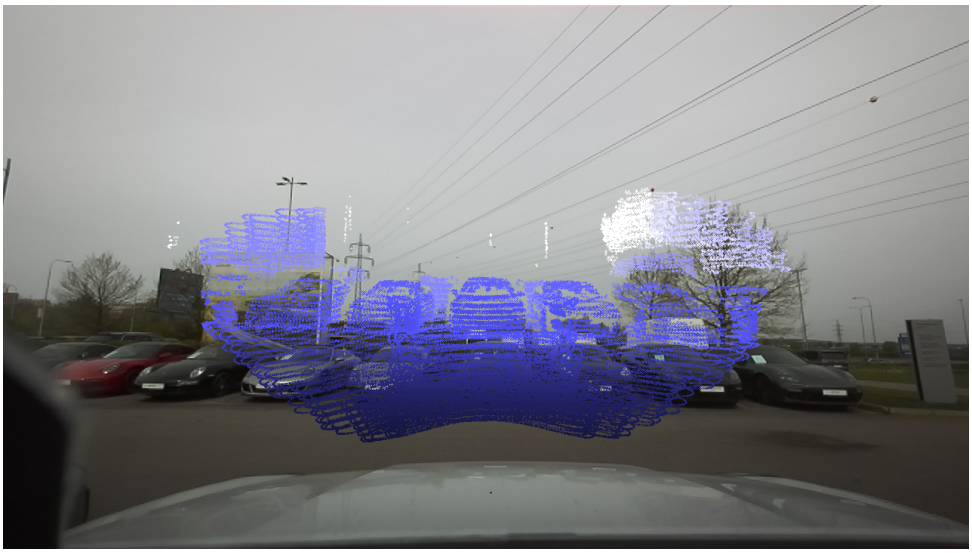


**Figure 5.2:** Projection of points before camera-LiDAR calibration.

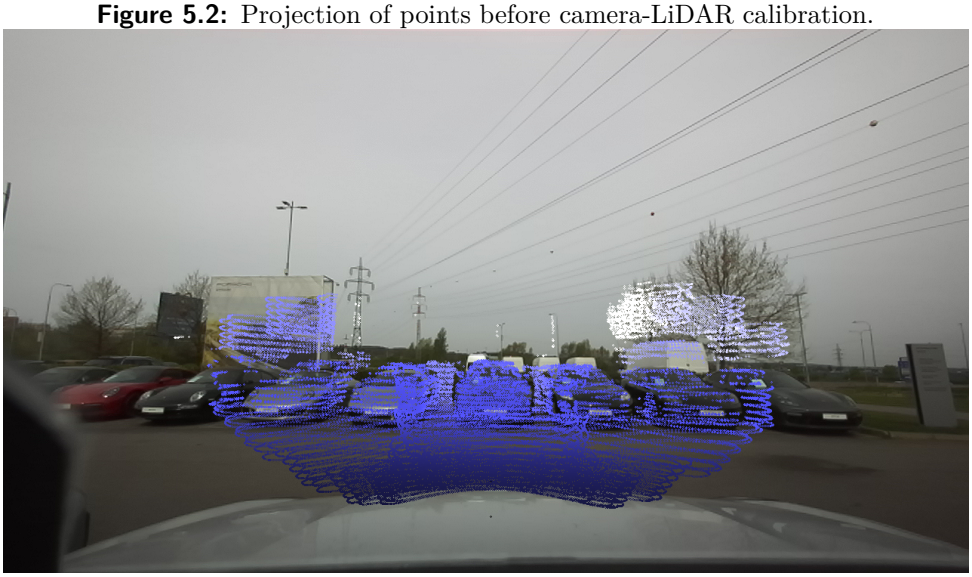


**Figure 5.3:** Projection of points after camera-LiDAR calibration.

## 5.5 ROS Node

This thesis aims to implement an object detector as a ROS node for the Jupiter car. The node is implemented for ROS 2 in Python 3.

| Message type | Description |
|---|---|
| Image | RGB image from camera |
| PointCloud2 | Lidar points in lidar coordinate system |
| tf2_msgs | Transformation from lidar to camera frame |
| sensor_msgs/CameraInfo | Intrinsic camera matrix |

**Table 5.3:** Used message types with description.

### 5.5.1 Node overview

The ROS node works as follows. The node subscribes to four topics. The first topic is the camera image. This topic contains the image captured by the camera on the Jupiter car. The second topic includes a PointCloud2 message. This message contains the lidar-measured points in a lidar coordinate system. To be able to convert these points and produce a camera projection, the node subscribes to a topic with a transformation matrix from the lidar coordinate system to the camera coordinate system and a topic containing the intrinsic camera matrix. The topics used are summarised in Table 5.3.

After the necessary transformations, the depth data are pre-processed to correspond with the data format used during the model's learning. This means that the lidar points are masked and linearly interpolated. At the end of this process, the lidar data are concatenated to the RGB image as a five-dimensional array.

After the pre-processing, the five-dimensional data are input to the trained detection model. The model makes the predictions, which are then processed and visualized as bounding boxes in the camera image. The image with detections is then published and can be displayed in RViz[6].

[6]ROS.org, Rviz, `http://wiki.ros.org/rviz`

# Chapter 6

# Experiments and Results

## 6.1 Introduction

In this chapter, the experiments and results will be discussed. Firstly, the domain transfer will be explained. Secondly, the benefits of depth modularity and mask utilization will be highlighted. Then, the results on the KITTI benchmark will be presented and compared with multiple models. Lastly, the detections of the ROS node will be demonstrated. For evaluation, the precision-recall curves for object detection are computed.

## 6.2 Domain transfer

As was mentioned before, domain transfer is a critical part of this thesis. This experiment uses two RGB models. The models were trained using different datasets: first on the KITTI raw dataset and the second on the Waymo open dataset. The testing dataset containing 42 images containing 146 objects from the Jupiter car was created for this experiment. The images were labelled using Robflow[1]. The results of this experiment are in Fig. 6.1. Only the results of detecting the car are presented. The model trained on the KITTI dataset achieved an mAP of 0.558, and the model trained on the Waymo

[1]Robflow, `https://app.roboflow.com/`

dataset acquired an mAP of 0.947, significantly outperforming the first model.

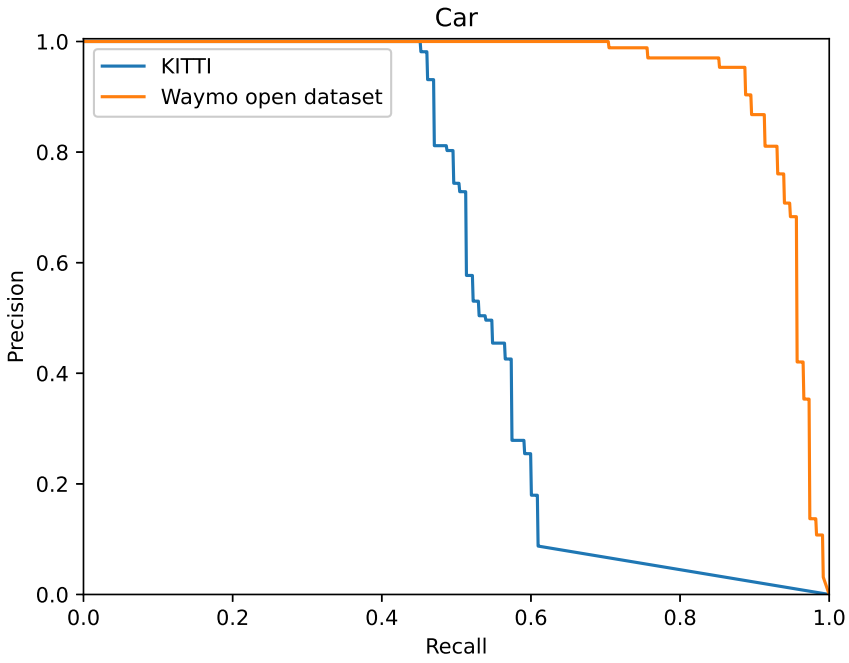



**Figure 6.1:** Comparison of models on Jupiter car data. KITTI-learned model achieved mAP of 0.558 and Waymo-learned model achieved mAP of 0.947.

## 6.3 Advantage of depth modularity

This experiment compares the results of the RGB and RGBD models. Both models were trained on the KITTI raw dataset. RGB model used only the images and RGBD both images and Velodyne points. Both models used the same augmentation methods as described in 5.3.3, batch size 16, and were trained for 30 epochs. The models were tested on the KITTI object detection evaluation[2] (2012), which was not presented to the models during the learning or validation. The results for detecting a car are in Fig. 6.2. The RGB model achieved an mAP of 0.814, and the RGBD model achieved an mAP of 0.915. The results are assessed in Table 6.1.

[2]Object Detection Evaluation 2012 `https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d`

| Model | mAP (Car) |
|---|---|
| RGB | 0.814 |
| RGBD | 0.915 |

**Table 6.1:** Results of the RGB and RGBD models of detecting a car.

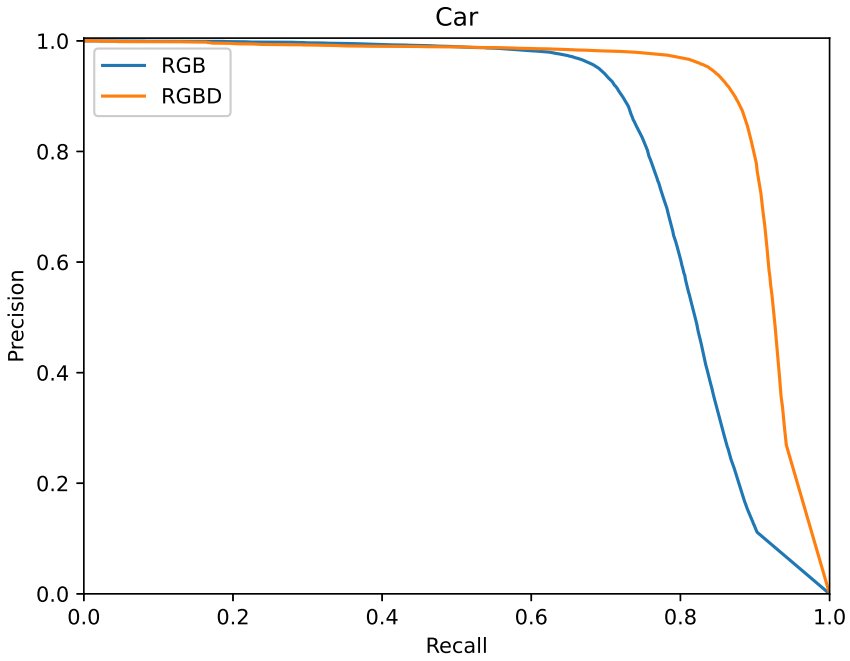



**Figure 6.2:** Results on the KITTI 2d object benchmark dataset of the RGB (mAP 0.814) and RGBD (mAP 0.915) models on car detection.

## 6.4 Mask

During the model training, I observed that lower mask values lead to better model performance. This remark resulted in this experiment. The experiment shows the results of two models learned on the KITTI raw dataset and tested on the 2d object detection benchmark dataset. The results of this experiment are in Fig. 6.3. The first model uses values of 0.1 on the position of measured lidar distances, and the second model was trained without the mask. The model with the mask performed an mAP of 0.874, and the model without the mask accomplished an mAP of 0.915. The results are assessed in Table 6.2.

| Model | mAP (Car) |
|---|---|
| with mask | 0.874 |
| without mask | 0.915 |

**Table 6.2:** Results of the models with and without the mask of detecting a car.

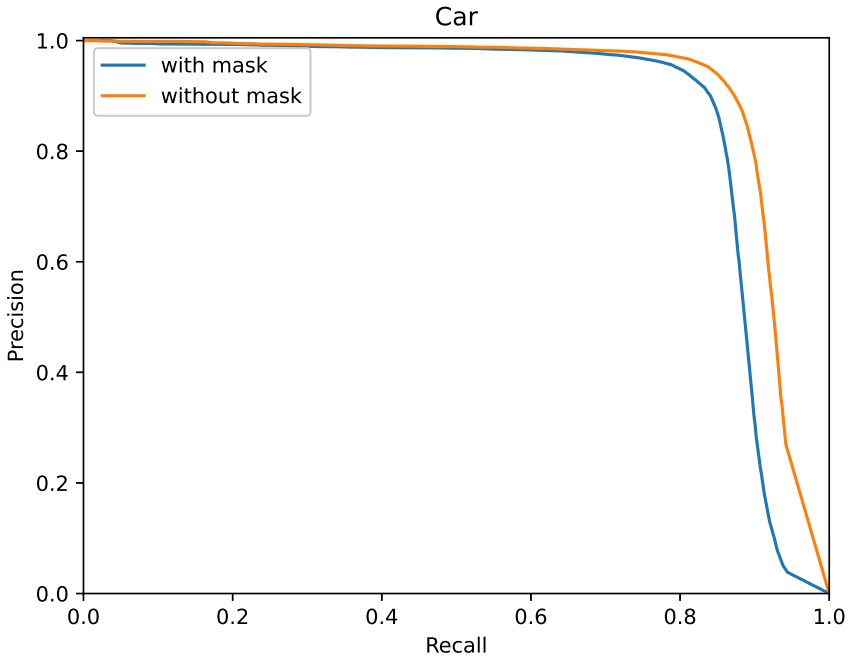



**Figure 6.3:** Comparison of model with and without the mask. The mAP of the model with mask is 0.874 and without the mask is 0.915.

## 6.5 KITTI Benchmark

This section compares the model implemented and trained in this thesis with selected models from the KITTI benchmark results table[3]. The chosen models are VirConv-S [WWSW23], SE-SSD [ZTJF21], YOLOv5x6_1920 [Jeo22] and Complexer-YOLO [SAK+19]. The VirConv-S model was selected because it has the best results out of the documented models[4]. The SE-SSD model is the top-performing model that utilizes point clouds obtained from a Velodyne laser scanner. The YOLOv5x6_1920 model was picked as a YOLO-based model with the best performance. Lastly, the Complexer-YOLO is the best-performing YOLO model with a published paper. It is important to note that

[3]KITTI Object Detection Evaluation 2012, `https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d`

[4]This model is currently (May 2023) ranked 2nd in the benchmark.

| Difficulity | min. bounding box height | max. occlusion level | max. truncation |
|---|---|---|---|
| Easy | 40 Px | Fully visible | 15% |
| Moderate | 25 Px | Partly occluded | 30% |
| Hard | 25 Px | Difficult to see | 50% |

**Table 6.3:** Definition of detection difficulties.

KITTI does not provide the dataset used for the benchmark of the models in the results table. The provided benchmark dataset is only a similar subset of the benchmark dataset. The results of the model trained in this thesis are thus only approximate.

The KITTI benchmark splits the dataset by the difficulty of object detection. The criteria that specify the difficulty are presented in Table 6.3. The comparison presented in this thesis is made with the results containing hard detections. The results are shown in Figure 6.4 and Table 6.4.

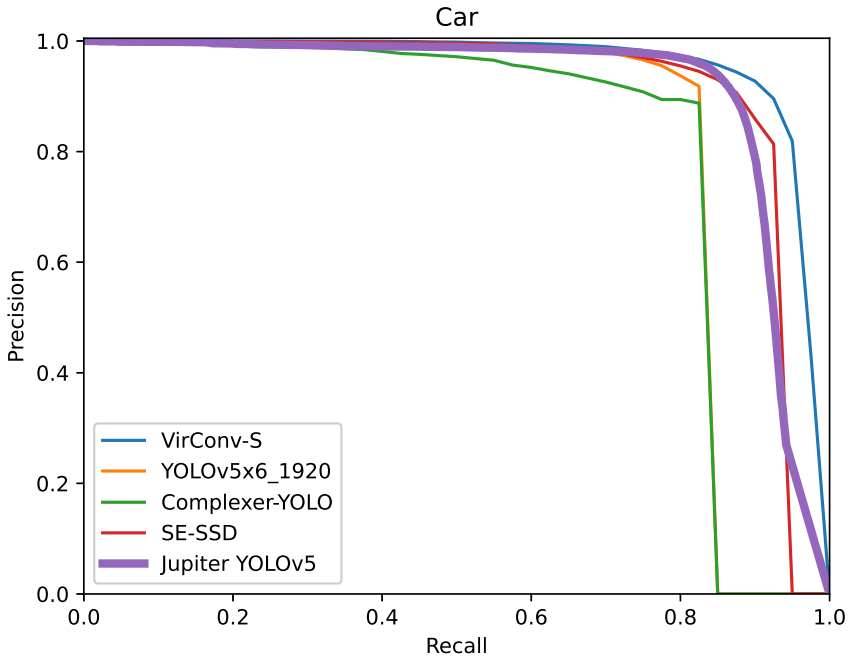



**Figure 6.4:** Results of the selected detection models. The model implemented in this thesis is highlighted.

| Model | mAP (Car) |
|---|---|
| VirConv-S | 0.945 |
| **Jupiter YOLOv5** | **0.915** |
| SE-SSD | 0.905 |
| YOLOv5x6_1920 | 0.815 |
| Complexer-YOLO | 0.796 |

**Table 6.4:** Results of selected models on the hard part of the KITTI 2d object detection dataset. The models are ranked by the mAP.

## 6.6 ROS node detections

The final section of this chapter shows two situations selected from a drive of the Jupiter car. Figures 6.5 and 6.6 display the images with detected bounding boxes. The attached video (*ros_node_demo.webm*) shows the functionality of the ROS node in real time. In the experiment, the model runs on the GPU NVIDIA GEFORCE RTX 3060. The detection time is also affected by the used hardware.


**Figure 6.5:** Example of ROS Node detections.


**Figure 6.6:** Example of ROS Node detections.

# Chapter 7

# Conclusion

This thesis aimed to implement object detection on a real car. The state-of-the-art detection methods were described in Chapter 2, and the necessary theory was presented in Chapter 4. The object detection model, YOLOv5 [Joc20], was utilized due to its optimal balance between quality and user-friendliness. The detection algorithm is implemented within a Robotic Operating System 2 [MFG+22], and the implementation is discussed in Chapter 5. The final results of this thesis are presented in Chapter 6.

From the results, the following can be concluded. The model achieved better detection results on the JUPITER car (section 6.2) when the model was trained on a more versatile and complex dataset containing images taken in different weather and light conditions. It was also shown that the depth modularity increases the model's performance on the testing dataset (section 6.3) as it adds additional information. The results also show the comparison of the trained model in this thesis and selected models on the KITTI 2d object benchmark (section 6.5).

Further work can be done in implementing more complex data augmentation methods for the detector with depth modularity, as it can increase the model's performance. Another possibility is to merge various datasets to create a larger dataset for the model's training. An interesting experiment would be to analyse how the model's size affects results on the JUPITER car. Smaller models may lead to quicker detection times.

The advantage of the implemented detector is that it is implemented as a ROS node. In this thesis, the node publishes the detections as an image

with bounding boxes representing the detections. It is a convenient way to demonstrate the functionality. However, the output of the node can be easily changed. This way, the node can serve as an information input for a higher-level application, which can then utilise the detections. In conclusion, the node can demonstrate the ability to detect objects to customers and can be further used in other projects within Porsche Engineering.

# Appendix A

## A.1 Used codes

- **YOLOv5 Ultralytics** - https://github.com/ultralytics/yolov5
- **KITTI raw data development kit** - https://s3.eu-central-1.amazonaws.com/avg-kitti/devkit_raw_data.zip
- **KITTI Object data transformation and visualization** - https://github.com/kuixu/kitti_object_vis
- **Lidar-Camera Projection** - https://github.com/darylclimb/cvml_project/tree/master/projections/lidar_camera_projection
- **Waymo Open Dataset** - https://github.com/waymo-research/waymo-open-dataset
- **ROS2 Node that subscribes to PointCloud2 messages and visualizes them using Open3D.** - https://gist.github.com/SebastianGrans/6ae5cab66e453a14a859b66cd9579239

## A.2 Attached files

- ros-node-demo.webm
- RGBDdetectnode.py

# Appendix B

# Bibliography


[BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-yuan Liao, *Yolov4: Optimal speed and accuracy of object detection*, 04 2020.

[DSC22] Shiv Ram Dubey, Satish Singh, and Bidyut Chaudhuri, *Activation functions in deep learning: A comprehensive survey and benchmark*, Neurocomputing **503** (2022).

[EEG+14] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman, *The pascal visual object classes challenge: A retrospective*, International Journal of Computer Vision **111** (2014), 98–136.

[GDDM13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2013).

[GK20] Hossein Gholamalinejad and Hossein Khosravi, *Pooling methods in deep neural networks, a review*, 09 2020, pp. 2980–2988.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, *Vision meets robotics: The kitti dataset*, International Journal of Robotics Research (IJRR) (2013).

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Gangenerative adversarial nets*, Journal of Japan Society for Fuzzy Theory and Intelligent Informatics (2014).

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick, *Mask r-cnn*, 10 2017.

[HSPM22] Johann Haselberger, Bernhard Schick, Marcel Pelzer, and Steffen Müller, *Jupiter – ros based vehicle platform for autonomous driving research*, 11 2022.

[IS15] Sergey Ioffe and Christian Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, Proceedings of the 32nd International Conference on Machine Learning (Lille, France) (Francis Bach and David Blei, eds.), Proceedings of Machine Learning Research, vol. 37, PMLR, 07–09 Jul 2015, pp. 448–456.

[Jeo22] Hyung-Joon Jeon, *You-only-look-once version 5 variant x6 [yolov5x6_1920]*, `https://www.cvlibs.net/datasets/kitti/eval_object_detail.php?&result=6d3ed1742c582d0ca4da9ea4a4a03ed1faffb41b`, 2022, Accessed: 2023-05-18.

[Joc20] Glenn Jocher, *Yolov5, ultralytics*, `https://github.com/ultralytics/yolov5`, 2020, Accessed: 2023-05-18.

[Joc23] ———, *Architecture summary*, `https://docs.ultralytics.com/yolov5/tutorials/architecture_description/#43-eliminate-grid-sensitivity`, 2023, Accessed: 2023-05-18.

[KB14] Diederik Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, International Conference on Learning Representations (2014).

[LAE+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, *SSD: Single shot MultiBox detector*, Computer Vision – ECCV 2016, Springer International Publishing, 2016, pp. 21–37.

[LGG+20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, *Focal loss for dense object detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **42** (2020), no. 2, 318–327.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, *Microsoft coco: Common objects in context*, Computer Vision – ECCV 2014 (Cham) (David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, eds.), Springer International Publishing, 2014, pp. 740–755.

[MFG+22] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall, *Robot operating system 2: Design, architecture, and uses in the wild*, Science Robotics **7** (2022), no. 66.

[ON15] Keiron O'Shea and Ryan Nash, *An introduction to convolutional neural networks*, ArXiv e-prints (2015).

[PW17] Luis Perez and Jason Wang, *The effectiveness of data augmentation in image classification using deep learning.*

[RDGF15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi, *You only look once: Unified, real-time object detection*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015), 779–788.

[RF17] Joseph Redmon and Ali Farhadi, *Yolo9000: Better, faster, stronger*, 07 2017, pp. 6517–6525.

[RF18] Joseph Redmon and Ali Farhadi, *Yolov3: An incremental improvement*, 04 2018.

[Rud16] Sebastian Ruder, *An overview of gradient descent optimization algorithms.*

[SAK+19] Martin Simon, Karl Amende, Andrea Kraus, Jens Honer, Timo Samann, Hauke Kaulbersch, Stefan Milz, and Horst Michael Gross, *Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.

[She20] Alex Sherstinsky, *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*, Physica D: Nonlinear Phenomena **404** (2020), 132306.

[SKD+19] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov, *Scalability in perception for autonomous driving: Waymo open dataset*, CoRR **abs/1912.04838** (2019).

[Sri16] Manoj Srivastav, *Transformation of an object in computer graphics: A case study of mathematical matrix theory*, Elixir International Journal (2016).

[TNJ+21] Dr. Narina Thakur, Preeti Nagrath, Rachna Jain, Dharmender Saini, Nitika Sharma, and Jude D, *Object detection in deep surveillance.*

[USGS13] Jasper Uijlings, K. Sande, T. Gevers, and A.W.M. Smeulders, *Selective search for object recognition*, International Journal of Computer Vision **104** (2013), 154–171.

[WBL22] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-yuan Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 07 2022.

[WLY+19] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh, *Cspnet: A new backbone that can enhance learning capability of cnn*, 2019.

[WLZ+19] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng, *Panet: Few-shot image semantic segmentation with prototype alignment*, 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (2019), 9196–9205.

[WWSW23] Hai Wu, Chenglu Wen, Shaoshuai Shi, and Cheng Wang, *Virtual sparse convolution for multimodal 3d object detection*, CVPR, 2023.

[XLL+21] Renjie Xu, Haifeng Lin, Kangjie Lu, Lin Cao, and Yunfei Liu, *A forest fire detection system based on ensemble learning*, Forests **12** (2021), 217.

[ZS18] Zhilu Zhang and Mert Rory Sabuncu, *Generalized cross entropy loss for training deep neural networks with noisy labels*, ArXiv **abs/1805.07836** (2018).

[ZSGY19] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye, *Object detection in 20 years: A survey*, Proceedings of the IEEE **111** (2019), 257–276.

[ZTJF21] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu, *Se-ssd: Self-ensembling single-stage object detector from point cloud*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2021, pp. 14494–14503.