DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

# Improved Detection Techniques in Autonomous Vehicles for Increased Road Safety

Gabriel Passos Moreira Pinheiro

Brasília, 18 de dezembro de 2020

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

# Improved Detection Techniques in Autonomous Vehicles for Increased Road Safety

**Gabriel Passos Moreira Pinheiro**

*Dissertação de Mestrado em Sistemas Mecatrônicos submetida ao Departamento de Engenharia Mecânica como requisito parcial para obtenção do grau de Mestre em Sistemas Mecatrônicos*

**FICHA CATALOGRÁFICA**

PINHEIRO, GABRIEL PASSOS MOREIRA
Improved Detection Techniques in Autonomous Vehicles for Increased Road Safety [Distrito Federal] 2020.
xvi, 67 p., 210 x 297 mm (ENM/FT/UnB, Mestre, Sistêmas Mecatrônicos, 2020).
Dissertação de Mestrado em Sistemas Mecatrônicos - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Mecânica

| | |
|---|---|
| 1. Autonomous Vehicles | 2. Computational Vision |
| 3. Pedestrian Detection | 4. Traffic Sign Recognition |
| I. ENM/FT/UnB | II. Título (série) |

**REFERÊNCIA BIBLIOGRÁFICA**

PINHEIRO, G.P.M. (2020). *Improved Detection Techniques in Autonomous Vehicles for Increased Road Safety*. Dissertação de Mestrado em Sistemas Mecatrônicos, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 67 p.

Gabriel Passos Moreira Pinheiro
SQS 314 - Bl. F - Ap. 106
Asa Sul
CEP 70383-060 - Brasília - DF - Brasil

## Dedication

*To the colleagues who are part of my academic journey for all the support in challenges and enduring friendship.*

*Gabriel Passos Moreira Pinheiro*

## Acknowledgments

*I thank God Almighty for the countless graces and blessings in my life and Mary Most Holy for the constant intercession. I am grateful to my parents, Alexandre and Thaís, and my sisters, Sophia and Constanza, for the love, support and incentives to my life of studies, and to my entire family.*
*I thank Prof. Dr.-Ing. João Paulo Carvalho Lustosa da Costa for the opportunity to complete this master's degree and for all the support and trust; to Dr. Ricardo Kehrle for his guidance at the beginning of my master's; and to all professors at UnB for their valuable contributions during this stage.*
*I also thank all my friends who accompanied me in all challenges of university and of my personal life. In particular, to Caio Fábio Oliveira for his support and encouragement; to Bruno Justino for his companionship on the journey to Germany and for all his help; to Miguel Pachá for his assistance. I am grateful to my friends from university both in undergraduate and master's degrees, for their cooperation in the pursuit of our goals. Also to all the great friends that I do not mention by name, but who made huge contributions to who I am today.*
*I am grateful to everyone who received me during my stay in Ingolstadt, Germany. In particular, to Ms. Johanna Hirschmann and Mr. Herbert Hirschmann for welcoming me into their home. I also thank my colleagues at Elektronische Fahrwerksystem GmbH for their valuable lessons and professional experience. In particular, to my supervisor Lothar Weichenberger for the internship opportunity; to Lukac Branimir, Tobias Behn and Andreas Schustek for their support during the internship. I also thank all the friends I made during my stay abroad. In the same way, I would like to thank the students I supervised at Technische Hochschule Ingolstadt, in particular Redge Castelino, Kairav Patel and Gokulesh Danapal, for their fruitful contributions, knowledge exchange and work accomplished.*
*I thank the entire academic community for the valuable knowledge employed in this research, and all professors and researchers with whom I published or contributed, in particular to Giovanni Santos. Also to the CAPES foundation for the financial support during the course of my master's degree. I also thank the open-source community for the important voluntarily contributions used in this work.*

*Gabriel Passos Moreira Pinheiro*

## ABSTRACT

The future widespread use of Autonomous Vehicles has a significant potential to increase road safety for drivers and pedestrians alike. As reported by the U.S. Department of Transportation, up to 94% of transit accidents are caused by human error. With that reality in mind, the automotive industry and academic researches are striving to achieve fully automated driving in real scenarios in the upcoming years. For that, more sophisticated and precise detection algorithms are necessary to enable the autonomous vehicles to take correct decisions in transit. This work proposes an improved technique for pedestrian detection that increases precision up to 31% over current benchmarks. Next, in order to accommodate current traffic infrastructure, we enhance performance of a traffic sign recognition algorithm based on Convolutional Neural Networks. Our approach substantially raises precision of the base model considered. Finally, we present a proposal for early data fusion of camera and LiDAR data, which we show to surpass detection using individual sensors and late fusion by up to 20%.

## RESUMO

A futura adoção em massa de Veículos Autônomos traz um potencial significativo para aumentar a segurança no trânsito para ambos os motoristas e pedestres. Como reportado pelo Departamento de Transportes dos E.U.A., cerca de 94% dos acidentes de trânsito são causados por erro humano. Com essa realidade em mente, a indústria automotiva e pesquisadores acadêmicos ambicionam alcançar direção totalmente automatizada em cenários reais nos próximos anos. Para tal, algoritmos mais precisos e sofisticados são necessários para que os veículos autônomos possam tomar decisões corretas no tráfego. Nesse trabalho, é proposta uma técnica melhorada de detecção de pedestres, com um aumento de precisão de até 31% em relação aos benchmarks atuais. Em seguida, de forma a acomodar a infraestrutura de trânsito já existente, avançamos a precisão na detecção de placas de trânsito com base em Redes Neurais Convolucionais. Nossa abordagem melhora substancialmente a acurácia em relação ao modelo-base considerado. Finalmente, apresentamos uma proposta de fusão de dados precoce, a qual mostramos surpassar abordagens de detecção com um só sensor e fusão de dados tardia em até 20%.

# SUMMARY

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

## Acronyms

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| AI | Artificial Intelligence |
| AV | Autonomous Vehicle |
| BPSK | Binary Phase Shift Keying |
| CCD | Charge-Coupled Device |
| CLPD | Complex Programmable Logic Device |
| CMOS | Complementary Metal Oxide Semiconductor |
| CNN | Convolutional Neural Network |
| CMY | Cyan, Magenta, Yellow |
| DAC | Digital-to-Analog Converter |
| DC | Direct Current |
| DL | Deep Learning |
| ETSI | European Telecommunications Standards Institute |
| FAD | Fully-Automated Driving |
| FCC | Federal Communications Commission |
| FMCW | Frequency-Modulated Continuous Wave |
| GNSS | Global Navigation Satellite System |
| GPIO | General Purpose Input/Output |
| GPU | Graphics Processing Unit |
| GTSRB | German Traffic Sign Recognition Benchmark |
| HOG | Histogram of Oriented Gradients |
| HSO | Hue, Saturation and Intensity |
| IC | Integrated Circuit |
| IF | Intermediate Frequency |
| IMU | Inertial Measurement Unit |
| ISM | Industrial, Scientific and Medical |
| ITU | International Telecommunication Union |
| LiDAR | Light Detection And Ranging |
| LNA | Low-Noise Amplifier |
| MCU | Microcontroller Unit |
| ML | Machine Learning |
| RBF | Radial Basis Function |
| RF | Radio-Frequency |
| RGB | Red, Green, Blue |
| SAE | Society of Automotive Engineers |

| | |
|---|---|
| SDR | Software-Defined Radio |
| SMA | SubMiniature version A |
| SVR | Support Vector Regression |
| SVM | Support Vector Machine |
| ToF | Time of Flight |
| TSR | Traffic Sign Recognition |
| VCO | Voltage-Controlled Oscillator |
| VGA | Variable Gain Amplifier |
| YOLO | You Only Look Once |

# 1 INTRODUCTION

With the recent developments in Autonomous Vehicles (AV) and sensors, a great potential to reduce human error in driving and improve safety in city streets and roads is becoming reality. Drivers will enjoy increased convenience, while pedestrians and cyclists will also benefit from a safer traffic environment. According to the National Highway Traffic Safety Administration of the U.S. Department of Transportation [1], approximately 94% of vehicle accidents occur due to human error.

Some of the challenges faced by AVs are to adapt and operate in harmony with existing traffic scenarios. That way, it is essential that AVs be able to detect pedestrians in various different situations, be it in adverse weather conditions or when they might be occluded. Besides, as AVs are expected share the roads with conventional vehicles until wider adoption is achieved, it is necessary to adjust to the current infrastructure. Thus, the detection of other vehicles and road information — such as traffic signs and traffic lights — is another major point of interest.

As many sensor-based applications are proposed, a higher data volume requires processing power and coordination to increase accuracy and generate better detection results. Specially in AVs, the novel area of sensor fusion techniques is giving rise to the processing of integrated sensors data with substantial improvements over previous approaches. By combining data from different sources, AVs are able to perform detection tasks in a much more accurate form and exploit sensor synergies to overcome existing limitations. Additionally, more advanced Artificial Intelligence (AI) and Machine Learning (ML) algorithms enable innovative and previously unseen detection and recognition applications.

This work proposes advancements on detection and recognition algorithms in Autonomous Vehicles. With a focus on improving road safety, we intend to increase precision and accuracy of pedestrian detection in challenging scenarios. Next, we propose and validate specific architectural enhancements in detection of traffic signs using neural networks, to accommodate existing road infrastructure. Finally, we introduce a framework that enables early sensor fusion of color camera and LIDAR data. The presented framework aims at surpassing individual sensor techniques and another data fusion implementation in object detection applications.

## 1.1 MOTIVATION

Potential safety gains with greater adoption of Autonomous Vehicles in roads make this area a trending field in both research and industrial applications. However, many challenges are currently under active discussion. Techniques applied must consider various aspects in order to successfully deploy a fully automatic autonomous vehicle. With a perspective of measuring development progress in mind, the Society of Automotive Engineers (SAE) has proposed a level system to classify the degree of automation in vehicles [2]. From SAE level 0, in which no automation is present, the gradual classification system culminates in SAE level 5, also called Fully-Automated Driving (FAD). To reach this level, a vehicle must be able to drive to any location under any circumstances, without requiring driver intervention.

With the continuously increasing automation levels in mind, this work intends to accommodate currently existing traffic infrastructure for improving AV behavior and decision-making, consequently contributing to safer transportation. Accounting for pedestrians in the streets, AVs must be able to recognize their presence even in scenarios where they are partially occluded or obstructed. Also, since autonomous vehicles will share space with non-automated vehicles, they must be able to follow the same traffic rules. Thus, it is necessary that an AV be able to access information displayed in traffic signs.

In parallel, the multiple sensors present in an AV enable cutting-edge methods to exploit their synergies and particularities. Allied with advancements in machine learning algorithms, we present a framework to make possible the use of sensor fusion using camera and LiDAR data on the well-established YOLO algorithm.

## 1.2 PUBLISHED WORKS

In complement to this work, the author has collaborated in areas of computer science, embedded systems, digital communications and autonomous vehicles. The resulting publications during the undertaking of the Master's Degree are presented next:

1. R. M. Castelino, **G. P. M. Pinheiro**, B. J. G. Praciano, G. A. Santos, L. Weichenberger and R. T. de Sousa Jr., "Improving the Accuracy of Pedestrian Detection in Partially Occluded or Obstructed Scenarios," 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020, pp. 834-838, DOI: 10.1109/ACIT49673.2020.9208877.

2. D. G. Rega, R. K. Miranda, E. Javidi, J. P. A. Maranhão, J. P. C. L. da Costa and **G. P. M. Pinheiro**, "ESPRIT-Based Step Count for Wearable Devices," 2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS), Gold Coast, Australia, 2019, pp. 1-5, DOI: 10.1109/ICSPCS47537.2019.9008702.

3. R. C. Ribeiro, E. D. Canedo, B. J. G. Praciano, **G. P. M. Pinheiro**, F. L. L. de Mendonça and R. T. de Sousa Jr.. (2020). Front End Application Security: Proposal for a New Approach.In Proceedings of the 22nd International Conference on Enterprise Information Systems - Volume 2: ICEIS, ISBN 978-989-758-423-7, pages 233-241. DOI: 10.5220/0009393202330241

4. E. D. Canedo, F. L. L. de Mendonça, G. D. A. Nze, B. J. G. Praciano, **G. P. M. Pinheiro** and R. T. de Sousa Jr.. (2020). Performance Evaluation of Software Defined Network Controllers.In Proceedings of the 10th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER, ISBN 978-989-758-424-4, pages 363-370. DOI: 10.5220/0009414303630370

5. G. Danapal, G. A. Santos, J. P. C. L. da Costa, B. J. G. Praciano and **G. P. M. Pinheiro**, "Sensor fusion of camera and LiDAR raw data for vehicle detection," 2020 Workshop on Communication Networks and Power Systems (WCNPS), Brasilia, Brazil, 2020, pp. 1-6, DOI: 10.1109/WCNPS50723.2020.9263724.

6. G. A. Santos, J. P. C. L. da Costa, D. V. de Lima, M. R. Zanatta, Bruno J. G. Praciano, **Gabriel P. M. Pinheiro**, Fábio L. L. de Mendonça, Rafael T. de Sousa Jr., "Improved localization framework for autonomous vehicles via tensor and antenna array based GNSS receivers," 2020 Workshop on Communication Networks and Power Systems (WCNPS), Brasilia, Brazil, 2020, pp. 1-6, DOI: 10.1109/WCNPS50723.2020.9263757.

## 1.3 OUTLINE

After this introduction, this work is structured as follows: Chapter 2 presents related works and the datasets used in the context of vehicular applications. Chapter 3 discusses the theoretical background employed for the development of the present applications. Chapter 4 introduces the proposed techniques, enhancements applied and the respective performance metrics considered to evaluate their performance. Chapter 5 demonstrates the results alongside comparisons to benchmarks present in literature to validate the improvements achieved. Chapter 6 concludes this work by summarizing the obtained outcomes and proposing future works.

# 2 LITERATURE REVIEW

This chapter presents a summary of current state of the art research and further works related to our proposed contributions in Section 2.1. Next, we present the datasets used for our autonomous vehicle applications in Section 2.2. The current focus is on pedestrian detection, traffic sign recognition and sensor fusion using cameras and LiDAR sensors. Each dataset is associated with a performance benchmark regarding each detection application. We adopt these criteria in the following chapters to evaluate the suitability of our proposed techniques.

## 2.1 RELATED WORKS

This section presents literature relevant to the areas of pedestrian detection, traffic sign recognition and sensor fusion. Works discussed are complementary to this work and pertain either to similar approaches to those we propose presently or pertinent its various scenarios of applicability.

Paper [3] discusses using basic statistical operators to adapt support vector regression (SVR) for the classification of pedestrians. By extracting image features using either HOG or Haar methods, authors validate the proposal over the Daimler Chrysler Pedestrian dataset. The achieved accuracy of 85% with HOG and 76.07% using Haar features, both using a SVR classifier.

In work [4], the author offers a method using HOG feature extractor with a SVM classifier, similar to our approach. The dataset used is organized by the author. From the overall detection rate achieved, the author concludes that the combination of HOG feature extractor provides accuracy and speed improvements in comparison to other methods.

A distinct approach is presented in [5]. Authors use thermal infrared information for the task of pedestrian detection. The proposed system is then capable of working during nighttime and under adverse lighting conditions. By acquiring thermal information from an infrared camera, data is input to a Faster R-CNN with a region decomposition branch for detection. Preliminary results show improved detection in the scenarios in which other approaches did not perform well, validating their work.

Authors in [6] explore the SVM classifier technique using a Haar feature extractor and Adaboost. After validation, the authors conclude that performance of pedestrian detection systems is hindered in obstructed or occluded scenarios. After discussing the challenges faced, the authors do not specify any datasets or benchmarks used.

Authors in [7] propose a YOLO-based approach for pedestrian detection, named YOLO-R. The novel method consists of three passthrough layers on top of base YOLO. In turn, these layers consist of a "Route" layer and a "Reorg" layer. They are tasked with connecting pedestrian feature maps of shallow layers to deeper layers, linking high and low resolution features. The authors use the INRIA dataset for validation. They reach a missed rate of 10.05%, lower than that achieve in YOLO v2 model of 11.29%.

In [8], authors focus on detecting pedestrians in different poses and perspectives. For this, they propose a multi-class detection network for distorted pedestrian images. Based on a Faster R-CNN algorithm, the authors trained and validated the system for classifying pedestrians in three levels of distortion. The authors achieved a missed rate between 13.4% and 42.3%, claiming increased speed and precision in distorted field-of-view scenarios.

The authors in [9] use a pyramidal part-based model to improve performance of pedestrian detection in a context similar to that treated in this work. The proposed method aims to reach more accurate predictions in a majority vote of the confidence score mechanism of visible pedestrian parts by cascading the pyramidal structure. The paper reached a 96.25% accuracy on the INRIA dataset and 81% accuracy rate on the PSU dataset. Our proposed approach surpasses the presented metrics for the latter dataset.

A similar approach to ours is shown in [10], in which a HOG feature extractor is paired with SVM for pedestrian classification. The method is trained using the INRIA dataset along with other samples from autonomous vehicles. The implemented image preprocessing and classifier techniques reach improved performance under optical flow changes and variation on lighting conditions.

Traffic sign recognition is addressed by several methods throughout literature. The review provided in [11] cites techniques such as the IECAM97 model, color indexing, edge detection features and using the Hough Transform. Other schemes using different datasets are presented next.

In [12], the authors propose a deep learning algorithm for traffic sign recognition over the GTSRB dataset. For this task, they employ a modified LeNet-5 network that extracts particular representations for better recognition. By using a CNN with its convolutional layers' outputs connected to a Multilayer Perceptron, they achieve close to 97.5% accuracy with their model.

The authors of [13] describe a method for traffic sign recognition in deep learning models. They perform preprocessing over images in order to focus on important features, followed by a Hough Transform to detect areas of interest. The output of this preprocessing stage is fed into a CNN for classification. With this method, authors achieve 98.2% accuracy on the GTSRB dataset.

In [14], the author proposes a novel CNN for image classification using the GTSRB dataset. The proposed algorithm includes spatial transformer layers and specific modules for detecting local and global features simultaneously. It is also claimed the network proposed is more robust to image deformations. The model is trained using two NVIDIA Tesla K40c Graphics Processing Units (GPU) and achieved an accuracy of 99.57% with Google Inception and 99.81% using a modified approach. We believe our results are significant in terms of processing, as we carry out training on Google Colab with fewer resources.

Paper [15] presents a CNN approach aimed at larger-scale traffic sign recognition. By using the R-CNN mask and a full recognition procedure, the authors propose substantial improvements over existing models that work on a smaller scale. To validate the approach, a novel dataset is also presented, containing 200 categories of traffic signs. Based on the overview provided, the authors achieve error rates below 3% on traffic sign inventory management applications.

Authors of paper [16] present a traffic sign detection and recognition system using color information. The implemented method consists of two modules, one for detection and another for classification and recognition. In the former, authors implement a color space conversion over images for segmentation to detect traffic sign presence. Next, the sign is highlighted, normalized and classified through a CNN. The experimental data shown claims a detection rate of more than 90% and the accuracy of recognition of above 88%.

In [17], the authors employ traffic sign recognition to extract speed limit information. The proposed method uses a color probability model to apply HOG feature extraction over images. The recovered features are then classified using SVM, removing all but the speed limit signs. Then, the numeric speed information is extracted, reaching a best-case scenario detection rate of 98.4% in daylight and approximately 96% in adverse lighting conditions.

For the context of real-time applications, [18] presents two modules for traffic sign detection and classification, respectively. Authors use a HOG extractor and a SVM classifier to detect the signs of the GTSDB dataset from color information inputs. Classification is then carried out by a CNN algorithm on the positive samples from the previous module. Performance is similar to state-of-the-art methods investigated, and authors emphasise their approach runs 20 times faster than comparable methods.

The authors of [19] introduce a 3D object detection framework called PointFusion. This method takes advantage of 3D point cloud form LiDAR combined with digital images. The framework is envisioned to work with simple concept and without having specific applications. Each sensor is individually processed by CNN and PointNet algorithms independently. Next, results are fused in a novel process to predict 3D bounding boxes. The framework is validated using the KITTI dataset driving scenes and a the SUN-RGBD dataset in indoor environments. The method shown reached better or equivalent performance on both datasets in comparison to other models.

In [20], authors present a road detection method with LiDAR-camera fusion. The novelty introduced consists of exploiting color and range information through a conditional random field framework. On the side of LiDAR, range data is upsampled from specific calibration parameter and processed along with denser road parameters from images. On the other sensor, a CNN model is used to perform the detection task. Data is then fused in the same framework using both road detection results from KITTI dataset samples. Accuracy is marginally improved in all tested scenarios in comparison to other individual and sensor-fusion methods.

In [21], authors tackle a road segmentation task using a LiDAR-Camera sensor fusion technique. They use synergies of the high-resolution, but sparse, data from LiDAR with the better visual context, but succeptible to noise, images from a camera. Data from the different sensors are fused by first converting them to a compatible format adaptively. Next, segmentation is performed using a CNN approach and tested over the KITTI road dataset.

Another YOLO-based proposal is presented in paper [22]. The author fuse LiDAR and RGB camera data in order to balance out the sensors' characteristics. The system carries out a weighted-mean approach to increase robustness of a YOLO object detection model in traffic scenarios trained on the KITTI dataset. Data fusion is performed at decision-level by averaging the contributions of each stage's YOLO outputs, outperforming individual sensor predictions.

Authors in [23] investigate the problem of multi-sensor fusion for 3D obstacle detection. The presented method performs a fusion of LiDAR and camera sensor data to estimate depth information by merging neighborhood information. Simultaneously, the method calculates the uncertainty of estimation to generate sampling points of interest. Following, the manual LiDAR gimbal rotation angle calculation serve to resample missed detection areas. Authors achieve an adaptable performance in detecting 3D obstacles in self-organized experiments.

A SegNet-based fast LiDAR-camera fusion process is presented by authors in [24]. In order to combine the distinct data formats, the authors tranform the LiDAR's height into spherical coordinates to increase data density. Then, the RGB camera channels are also projected onto this coordinate system. The resulting fusion leads to a faster processing speed due to reduction in the amount of data. The fused sensor data channels from the KITTI dataset are fed into a proposed specialized CNN for road segmentation. Performance shown reaches significantly faster running time with similar accuracy, but with more frequent occurrences of false positives and false negatives in comparison to other methods.

In [25], authors propose raw data fusion technique in a context different from ours. The Enet-CRF-LiDAR employed adopts Delaunay Triangulation to increase density of LiDAR-provided data and next combine it with color camera images. The resulting data is thus processed through an Efficient Neural Network and a Conditional Random Field to perform semantic segmentation over the KITTI dataset. Improvements in time efficiency range from 75.96% to 94.35%, while maintaining similar accuracy performance to the compared methods.

A 3D SVM object classifier is presented in [26]. By applying sensor fusion to LiDAR and camera technologies, authors look into ensuring high detection accuracy in pedestrian detection systems. For this, 3D point cloud data is used to enhance performance in occluded scenarios. The multi-step approach starts with a region proposal algorithm that is passed onto a second stage for classification redundancy. The detection validation presented reached an average of 99.16% accuracy rate for detecting pedestrians.

## 2.2 AUTOMOTIVE DATASETS

With the advent of autonomous vehicles technology, many applications and novel techniques have been proposed to increase performance and introduce innovative detection techniques. Several such techniques have shown significant improvements over the state of the art, as discussed in the last section. From this expanding research scenario, it is necessary to establish referential performance measurements. Many institutions then began to publish open datasets, that are used to measure effectiveness of algorithms over the same benchmarks.

In the area of AV research, multiple datasets provide material such as images or other sensor data (e.g. LiDAR, Radar). Many also provide labeled data for usage as ground-truth and benchmark results to compare and validate improvements proposed. Since improving road safety in the context of this work includes multiple areas of detection, we present next the available datasets and discuss their applicability in our work.

### 2.2.1 Pedestrian Detection

Human detection in roads has been a significant demand in AV applications. Since traffic accidents involving pedestrians tend to result in more harm to them, avoiding such hazards is a concern even at the lowest levels of SAE automation. Thus, many companies and universities have developed datasets with the intent to support pedestrian detection. Some of the best known collections include the INRIA [27], CALTECH [28], Daimler [29] and CVC [30] datasets.

Due to ongoing research, benchmark results are frequently updated for each dataset, showing the accuracy of detection schemes applied on them. This enables researchers and developers to reference improvements achieved over these datasets. Nonetheless, no dataset includes all possible scenarios in pedestrian detection or all challenges that may appear in these cases. Since a FAD vehicle needs to function under any conditions, many other datasets considering distinct scenarios are introduced in research publications and industrial development. Images portraying situations such as adverse weather, improper lighting conditions, among others, are produced to refine these edge cases in pedestrian detection.

Our focus in this part of the work is to consider the real-world scenarios in cases that pedestrians are obstructed or partially occluded in the vision of cars. As this poses additional challenges, current algorithms have difficulties to correctly detect humans under these circumstances. This is undesirable since failure in detection can lead to wrong maneuvers, causing accidents.

From the dataset proposed in [31], the authors employ an approach to perform human detection combining HOG descriptors and SVM. The presented INRIA dataset consists of 1800 human images in different backgrounds with various poses.

In order to better represent everyday scenarios in Asian street environments, the authors of [32] present the PSU dataset. Said dataset contains a total of 1551 images, consisting of 1051 positive samples and 500 negative samples. For setting the benchmark performance, the authors present a HOG feature extractor technique, employing linear kernel SVM to detect pedestrians on the novel dataset and cross-validade on the INRIA dataset. The paper claims 48% detection rate on the PSU dataset and 54% over the INRIA dataset.

Our work proposes an improved technique in scenarios of occluded or partially obstructed pedestrians. Based on the approach proposed in [32], we validate our approach over both the PSU and INRIA datasets, showing significant improvement over both benchmarks.

### 2.2.2 Traffic Sign Detection

In the context of currently installed transit infrastructure, traffic signs are an essential form of communicating immediate traffic regulations and local points of interest, such as sharp turns of pedestrian crossings. Each country implements their own system of traffic signs, including some widely adopted standards and necessary particularities for each location's traffic scenarios.

Training traffic sign recognition models require an unified dataset, containing a determined set of samples. For this application, we utilize the German Traffic Sign Recognition Benchmark (GTSRB) dataset, provided in [33]. Traffic sign samples in the GTSRB dataset are extracted from 1-second video sequences. That is, each instance of real scenarios yields 30 examples with usually increasing resolution as the camera approaches the sign. The sizes of each sample set varies from 15 to 250 elements.

Table 2.1 shows the categorized data distribution — training, validation and testing. Next, Figure 2.1 illustrates the sample traffic sign images in the dataset.

Table 2.1: German Traffic Sign Recognition Benchmark samples distribution

| **Category** | **Number of Images** |
|---|---|
| Training Data | 31 367 |
| Validation Data | 7 842 |
| Testing Data | 12 630 |

As a popular dataset for this application, using the GTSRB enables us to compare our results to those published by other researchers.

### 2.2.3 Camera and LiDAR Sensor Fusion

On the other parts of this work, we consider datasets consisting of only images. In order to expand the scope, it is necessary to consider that AVs contain a wide array of sensors, included, but not limited to, radar, LiDAR and sonars. Thus, it is crucial for the long-term success of autonomous driving to take advantage of sensor fusion techniques. Thus, we now consider multi-sensor datasets for our current application.

A widely know dataset is the The KITTI Vision Benchmark Suite, proposed by [34]. Refined over the year by the Karlsruhe Institute of Technology in partnership with the Toyota Technical University in Chicago, the dataset presents extensive samples to a multitude of computer vision applications. The dataset was generated by equipping a vehicle with two high-resolution color and grayscale video cameras and laser scanner. Also, a GPS measurement unit provides global

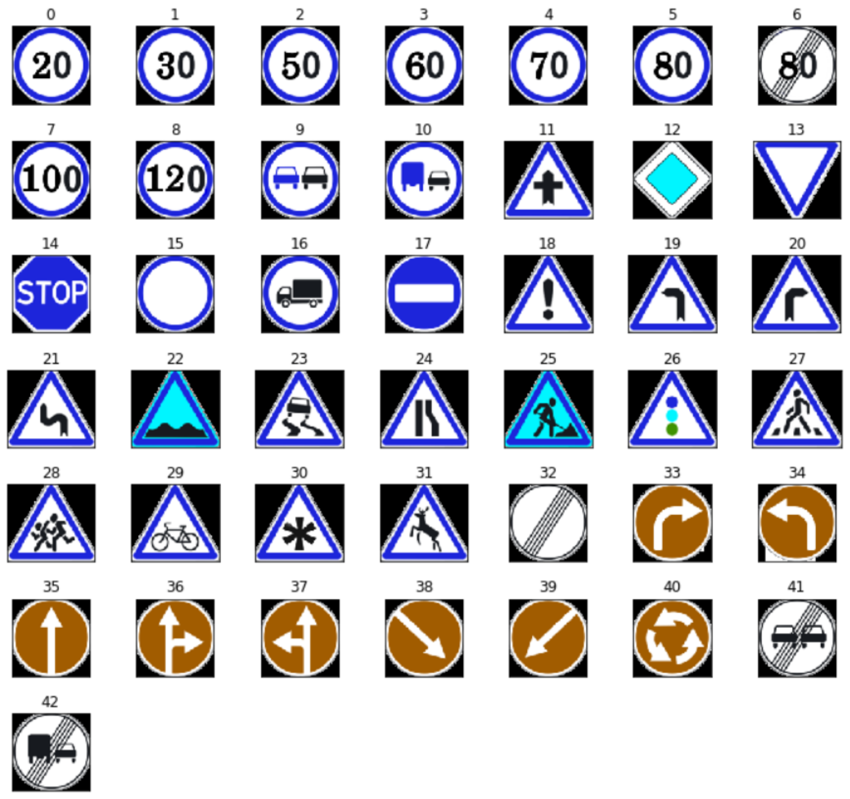


Figure 2.1: Sample data from the German Traffic Sign Recognition Benchmark.

coordination of the vehicle in each sampling instant.

The KITTI dataset is a widely used collection. Our specific subset used consists of 7,481 training images and 7,518 testing images, including a total of 80,256 labeled objects. Besides the images, the laser scanner point clouds are available for each respective image. The rotation and translation matrices used for calibration and projection from various coordinate system are also provided. With this dataset, we are able to perform raw camera and LiDAR data fusion with sufficient data.

# 3 THEORETICAL BACKGROUND

This chapter presents the theoretical background employed during this work. We begin by providing an overview of sensor equipment in AV. With a main focus on cameras and LiDAR — used in our algorithms —, we explore how environmental traffic sensing is achieved and the particularities of each sensor. Then, we contextualize the area of sensor fusion in regards to the application that is proposed in Chapter 4. Following that, we discuss the ML and AI frameworks applied in this work targeted at the image recognition and object detection tasks.

## 3.1 SENSORS IN AUTONOMOUS VEHICLES

Sensor devices are an integral part of an autonomous vehicle navigation systems. In order to reach FAD, it is necessary to sense the surroundings of a vehicles even better than a human driver can. Thus, so as to emulate and augment perception, cutting-edge sensors are being produced as of recently.

The fundamental tasks of sensors in an AV system is to provide precise localization and to track surrounding objects, such as road lanes, other vehicles and bystanders. In order to enforce device specialization and separation of concerns, [35] presents a framework that separates motion and sensing tasks, shown in Figure 3.1. To the left of Figure 3.1 are represented the blocks in which sensing devices are utilized, namely the Dead Reckoning and Perception stages.

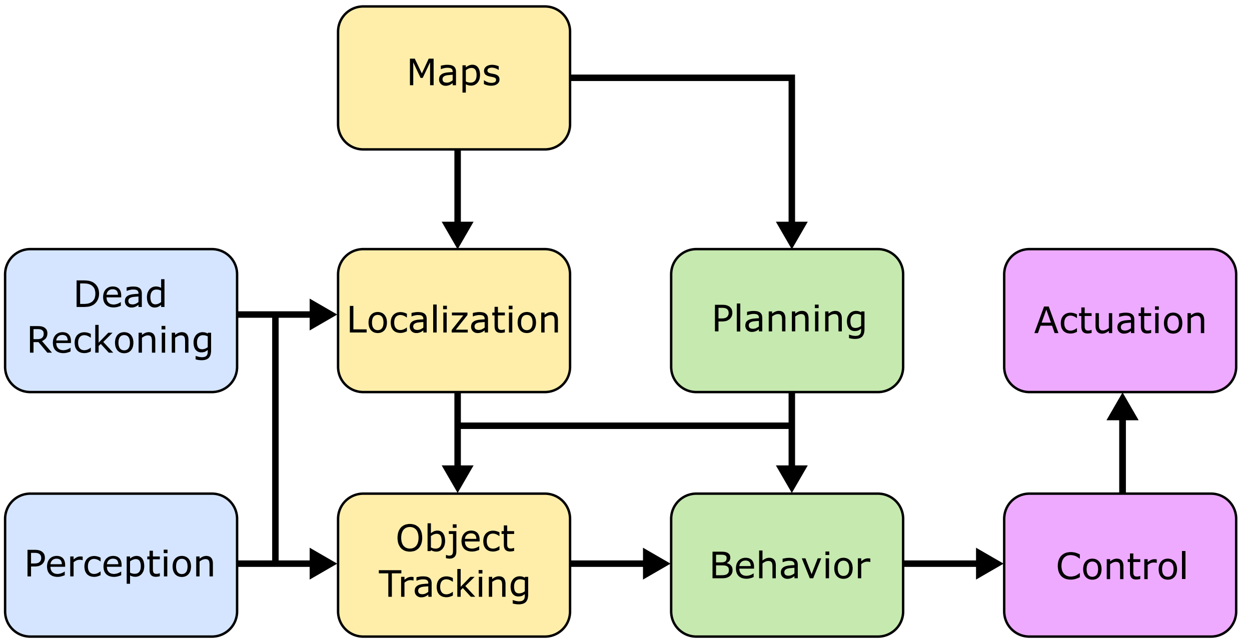


Figure 3.1: Framework considered for an autonomous vehicle system.

Dead Reckoning tracks the internal motion of the vehicle and its localization in a local and global level. The main gadgets used in this stage are the Inertial Measurement Unit (IMU) — for tracking motion — and a Global Navigation Satellite System (GNSS) — for keeping localization. Accessory devices such as a compass for orientation and angular encoders to assist the IMU are commonly present as well.

In terms of sensors, the focus of this work lies in the Perception block, responsible for providing data for object identification and road tracking. Multiple sensors are necessary to gather all necessary data, requiring the operation of a wide array of devices. Active and passive sensors provide different data sources over many perspectives of the surrounding environment.

More than five types of sensors are currently employed in AVs. Even in lower levels of automation, Perception is already applied to assist drivers. Automotive systems such as cruise control, auto-braking with collision detection, auto-parking and parking assist are examples in which Perception help drivers to have a safer and more comfortable ride experience. Perception systems include external input data and feedback, with limitations due to characteristics of the hardware employed. In some situations, combining data from multiple devices, in what is called sensor fusion, is advantageous to balance their limitations or increase detection precision.

In the following subsections, we describe the main types of devices used for Perception, with a focus on cameras and LiDAR. Subsequent to that, we discuss techniques of sensor fusion, in which data from different sources are combined to generate augmented view, create redundancy and adjudicate decisions.

### 3.1.1 Cameras

Digital cameras are widespread devices used for image capturing and video recording in a multitude of applications. Currently, cameras are already used in automobiles for rear-view assistance and as dashcams. Digital images provide crucial visual information about surroundings. Individual cameras output 2D vector in either full-color mode in three channels Red-Green-Blue (RGB), or as a grayscale image in one single intensity channel. Depending on the composition of the image sensor and lens filter, cameras can capture either visible or infrared (IR) light spectrum. Digital camera sensors have a lower dynamic range than the human eye, hence it is not possible to perfectly reproduce images in the same intensity distinction as humans can see.

From the beginning of AV research, cameras have been an ubiquitous equipment. Due to their reduced costs by popularization and technological maturity, extracting external information with higher performance and accuracy using cameras are an ongoing interest in this research area. Well-established digital image processing techniques along with novel artificial intelligence algorithms makes cameras an indispensable addition to any AV.

In relation to other sensors considered later in this work, cameras have some unique characteristics. They work for long and short range detection and have large field of vision and angular resolution. Besides, as an established technology, it is more flexible and economical to rely on cameras for some applications than on recent LiDAR devices, for instance. Furthermore, cameras are currently the only viable option for tasks such as traffic light classification and turn signal detection, as they readily make available fundamental color information.

However, some concerns must be discussed relating to the drawbacks of camera sensors. As a passive device — only capturing incoming light signals —, it is greatly hindered by adverse weather and lighting conditions. Besides, since it outputs 2D vectors, no depth data is present in a single camera's output. On top of that, cameras with higher resolution stream large volumes of data that need to be processed in real-time for traffic application. This situations increases the processing power and energy necessary for the camera, making its operation resource-intensive.

Next, we discuss different forms of camera construction and considerations regarding digital sensors. We also evaluate quantitatively the output and parameters of a digital image.

A digital image is represented as a 2D matrix, in which each pixel corresponds to an intensity value in the matrix. From this, its resolution is given by its number of pixels. By multiplying its pixel height by width, it results in the total number of pixels — its resolution —, usually expressed in Megapixels or dots per inch (dpi).

To form a digital image, the camera is equipped with a device capable of sampling incoming light values and output its intensity. These digital image sensors have two main construction types, namely Complementary Metal Oxide Semiconductor (CMOS) or Charge-Coupled Device (CCD). We focus on the former type, as the latter is mostly used for scientific and specialized applications, not in the consumer automotive industry.

In a simplified manner, a CMOS camera sensor works by exposing a photodetector cell area built with an array of transistor circuits for the pixels. When light hits a pixel, its circuit outputs an electric signal corresponding to the light intensity. We can model it, according to [36], as an intensity function $f(x, y)$ in terms of the spacial coordinates $x$ an $y$ axis, thus:

$$f(x, y) = i(x, y) \cdot r(x, y), \tag{3.1}$$

where $i(x, y)$ is the illumination of the image's subject and $r(x, y)$ is its reflected light component. In Equation 3.1, the terms $i(x, y)$ and $r(x, y)$ are defined in the following interval:

$$0 \leq i(x, y) \leq \infty, 0 \leq r(x, y) \leq 1. \tag{3.2}$$

As $i(x, y)$ is unbounded upwards in Equation 3.2, a maximum sensitivity must be set for a given sensor:

$$\begin{aligned} L_{min} \leq & I \leq L_{max}, \\ L_{min} = & i_{min} \cdot r_{min}, \\ L_{max} = & i_{max} \cdot r_{max}. \end{aligned} \tag{3.3}$$

If the bounds of the sensor in Equation 3.1.1 are not met, we observe either sensor saturation — in the case of too much light — or a dark image if the point is not illuminated. The captured chromatic light is characterized by three main factors: its radiance, luminance and brightness. The first expresses the total energy emitted by the light source in watts. Next, luminance accounts for the amount of energy received by the observer lumen. Brightness describes then the intensity of observed light subjectively [36].

Intensity values represent an unidimensional quantity and, by itself, is used to generate gray-scale images. In terms of color information, we can make a distinction in terms of brightness, hue and saturation. Brightness is the equivalent subjective descriptor of light intensity. Hue describes both a physical factor and a subjective perception, as it expresses the dominant wavelength of chromatic light and the predominant perceived color by the subject. Saturation hence expresses how pure the incident light is, in terms of the amount of white light present. In order to objectively describe a beam of color light, we set the chromaticity concept, combining its saturation and hue.

For achieving a color image, it is necessary to sample light in multiple channels to form the correct pixel matrix representation for each one of them. Henceforth, there are different manners of representing color spaces. The most used in digital cameras and screen is the Red-Green-Blue (RGB) representation. There are also Cyan, Magenta, Yellow (CMY), used more commonly in printers. Also there is the Hue, Saturation and Intensity (HSO), a perceptive model that separates the intensity component from the chromaticity, used in gray-scale applications.

In the RGB color space, a digital camera sensor is then equipped with filters of one of each of the basic colors. Thus, each pixel only samples light from that specific wavelength. By joining three pixels of different colors, its result is one full color point, according to the resolution available. The number of representable colors is given by the product of color levels achievable in each RGB pixel. Thus, a 24 bits color represents 8 bits of RGB, in a total of more than 16 million colors available.

As 2D arrays, digital images can be processed as matrices. Operations can range from per-element or per-matrix calculations, using linear or non-linear functions. In this work, we narrow down the possible manipulations on digital images and focus on two special sets: logical operations and geometric operations. The following discussion is based on [36].

Logical operations on images are related to Venn diagrams. In Figure 3.2 adapted from [36], $A$ and $B$ are given regions of a space $U$. The union operation $A \cup B$ represents the areas of both regions. Next, the intersection operation, denoted $A \cap B$, results in the areas where the regions overlap. Also, the complement of region $A$, represented by the operation $A^C$, denotes the areas of the space $U$ that do not overlap with $A$. Finally, the difference between the region $A - B$ is the area of $A$ that does not intersect $B$.

Furthermore, geometric operations over images intend to modify its spatial characteristics, for instance, performing rotation or inversion in relation to an axis. We next define the matrix operation for geometric manipulation. First, we define a 2D image as a row vector consisting of its spatial coordinates $x$ and $y$ and another row vector for the coordinates of the transformation

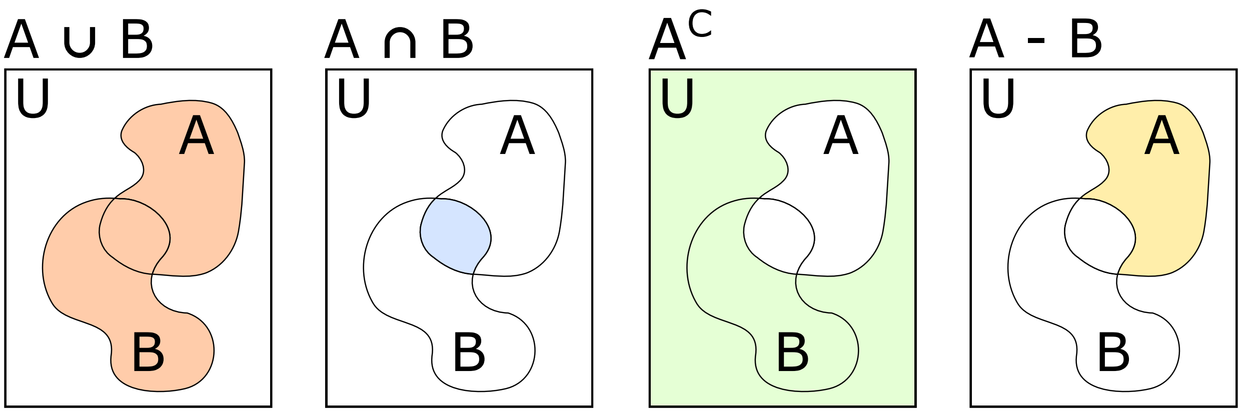


Figure 3.2: Logic operations performed on digital images.

space $v$ and $w$. We relate them by transformation matrix, also known as affine matrix $T$, hence:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} v & w & 1 \end{bmatrix} \cdot T = \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}. \tag{3.4}$$

From choosing the correct matrix $T$, we are able to perform the following transformations:

- Identity:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

- Scaling, with $c_x$ and $c_y$ the horizontal and vertical translation distances, respectively:

$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

- Rotation, with $\theta$ the angle of rotation:

$$\begin{bmatrix} cos\,\theta & sin\,\theta & 0 \\ -sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

- Translation, with $t_x$ and $t_y$ the horizontal and vertical translation distances, respectively:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \tag{3.8}$$

- Vertical shear, with $s_v$ the shear factor:

$$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

- Horizontal shear, with $s_h$ the shear factor:

$$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

### 3.1.2 LiDAR

The field of semiconductor optical gave rise to many indispensable technologies. One such application is lasers, which are narrowband coherent amplified stimulated light emissions [37]. Due to their high spacial coherence, lasers can be formed into beams and used for ranging applications. With that principle in mind, recent research has culminated into the production of the Light Detection And Ranging (LiDAR). This technology uses laser beams to measure distance from objects by measuring the reflected portion and time of flight (ToF) of said laser signals. Due to their narrow bandwidth, lasers can be detected with reduced light interference. Besides, as many laser frequencies lie away from visible spectrum, it is possible to avoid light noise and not disturb human bystanders with the laser beams.

LiDARs are considered mid-range sensors, effectively working from 3 to 150 m of distance. They output the position and intensity of light reflected in a given point, measuring both its distance and reflectance values. As a ToF based device, LiDAR measures difference between transmitted and reflected pulse of laser light (e.g. attenuation, time delay) [38]. From that, it is possible to create a 3D point cloud from readings of the surroundings of the sensor.

As an emerging technology, no standard form of construction has been consolidated for LiDAR. Thus, manufacturers have produced different builds, with ongoing evaluation in both industry and academia. In any case, two major build formats can be observed nowdays, namely mechanical or solid-state builds. Mechanical has higher beam density, as it is able to physically rotate more concentrated rays. In turn, solid state builds uses electromagnetic beam steering. This sacrifices beam density for more reliability, as this type does not depend on mechanical parts.

Regarding the type of laser used in LiDAR, the most common wavelengths are near 905 nm or 1550 nm [39]. Equipment operating in the 905nm IR-range need to limit their emission power, as IR can be absorbed by human eye, thus having harmful potential at higher power. However, these types of LiDAR are cheaper to produce and also more commercially available. On the other hand, 1550 nm devices can be more powerful, but are more expensive and may have problems with water absorption around this wavelength.

Other variations in the standard pulsing LiDAR technology concern other modes of operation for the laser beams. In flash LIDAR [40], the emitted light is output in the form of flashes instead of pulses. This results in a occupancy or free-space grid, with reduced accuracy. The other more advanced type is the Frequency-Modulated Continuous Wave (FMCW) LiDAR. As its name implies, the use of a continuous beam enables this variant device to measure Doppler signatures of objects, from with velocity readings can be obtained.

LiDAR presents many advantages in relation to other sensors present in AV. From the start, its multiple readings can measure 3D position and intensity, with some devices even measuring velocity, of objects in traffic scenarios. The use of laser beams also afford it a high spatial resolution and a wider field of vision compared to radar. Besides, it is an active sensor, making it independent of lighting conditions.

With the technology still in its early years, some shortcomings are present in LiDAR devices. In economical terms, the more advanced devices are much more expensive in comparison to other sensors, with some equipment costing up to thousands of dollars. Aside from that, not many of these devices are automotive-grade yet, with lengthy periods of certification to ensure safety of using LiDAR. In any case, some inherent limitations of LiDAR need attention. For instance, it is not capable of acquiring color information from objects, excluding it from completely substituting cameras. Also, as it is depends on light beams, some adverse weather conditions can seriously hinder its detection capabilities, when laser may scatter due to rain or snow for example. Also, some problem with beam spreading at longer distances limit its resolution in longer ranges and introduce some sparsity in data.

Now, we take a closer look into the mathematical modeling of LiDAR. According to [41], considering a target point at a distance $z$, the resulting received power $P_r$ given in watts, is:

$$P_r(z) = E_l \frac{c\rho(z)A_R}{2R^2}\tau_T\tau_R \exp(-2\int_0^z \alpha(z')dz'). \tag{3.11}$$

In Equation 3.11, the emitter is represented by the coefficient $E_l$, which is the laser pulse energy in J, by its efficiency $\tau_T$ and by the speed of light $c$ with which the beam travels. The target point characteristic is its back-scattering coefficient $\rho(z)$. The parameter $\alpha(z')$ is the scattering coefficient of the atmosphere along the path travelled by the beam, which can also account the effect of rain or snow. Finally, the receiver is characterized by its effective area $A_R$ and efficiency $\tau_R$.

By neglecting the spatial variation of the target point for simplicity and reducing sensor parameters to a single coefficient $C_s = cE_l A_R \tau_T \tau_R / 2$, we arrive at a simplified version of Equation 3.11, then:

$$P_r(z) = \frac{C_S \rho}{z^2} e^{-2\alpha z}. \tag{3.12}$$

Another consideration can be made, as $C_s$ is a constant for a particular sensor. From Equation 3.12, we can express it in terms of the relative sensor power $P_n = P_r / C_s$, given by:

$$P_n(z) = \frac{\rho}{z^2} e^{-2\alpha z}. \tag{3.13}$$

### 3.1.3 Radar

Radar is originally an acronym for Radio Detection And Ranging, given the fact that it works by measuring the ToF of a radio wave. In its inception, radar devices were created to detect metal vehicles in war scenarios, such as airplanes or ships. Its main strengths are then tracking of moving and static objects, especially the metallic ones. Its particularities include the capability of simultaneous measure velocity and position based on the Doppler Effect. In comparison to other sensors considered, radar typically has lower resolution. However, recently development imaging radar substantially increase this resolution [42]. Applications in the automotive industry range from blind-spot detection, lane-change assist and front/rear cross-traffic alert [43].

Radar devices can be built as continuous wave equipment or using pulsed signals. In automotive applications, radars operate in either 24 GHz or 77 GHz. In principle, the former frequency band is employed in short-range devices — working up to 70 m of distance—, whilst the latter is utilized in long-range detection, functioning up to 250 m. However, due to regulation and standard changes by the European Telecommunications Standards Institute (ETSI) and the Federal Communications Commission (FCC), the 24 GHz industrial, scientific and medical (ISM) band will be phased out for radar devices by 2022 [43]. Thus, moving forward, the automotive industry is increasing traction of radars using the 77 GHz band range, as regulated by the International Telecommunication Union (ITU) in [44]. Both short and long-range devices are available in this frequency. Still according to [43], some of its advantages include wider bandwidth — used for long range detection —, higher range resolution and better velocity measurement precision.

As a lower cost device, radar includes significant features of interest. Firstly, the measuring velocity and position has proven to be useful, as radar is in already in wide production and adoption, lowering costs. Besides, as is works based on radio waves, it is a robust alternative to situations with difficult weather and lighting conditions.

Some limitations are observed when using radars, as they are not commonly high-resolution and have difficulties telling static objects apart. Also, simpler equipment does not have 3D capabilities, only providing planar information. Finally, radars can be negatively affected by debris or ice build-up over the sensor, requiring human intervention to clean up.

### 3.1.4 Sonar

The sonar working principle is very similar to radar, instead measuring the ToF of a low-frequency wave. In juxtaposition to the diverse sensors, sonar is the one with the lowest cost. Already used in vehicles for parking assist, they are automotive-grade and well establish in industrial applications.

Yet, sonar has the lowest spatial resolution and shorter range of sensors considered, working from just 15 cm to maximum 5.5 m in more advanced implementations. In any case, their very low-cost and reliability on short range make them a favorable choice for parking assist even in more advanced automotive systems.

## 3.2 SENSOR FUSION

In this section, we present concepts of sensor fusion, a technique in which data from different sources are coordinated to improve or augment individual readings. A number of different schemes can be used to setup sensor fusion applications, we thus provide a panorama of the possibilities and focus on the paradigm employed in this work.

Sensor fusion can be defined as the process of merging data from multiple sensors to reduce the uncertainty in navigation and detection tasks. Configurations of sensor fusion are structures in three fundamental manners, according to [45]:

- **Redundant or Competitive sensors**: All sensors provide the same measurement or information;
- **Complementary sensors**: The sensors produce independent or disjoint types of information about the surroundings;
- **Cooperative sensors**: The sensors contribute with sequential information about the surroundings.

Each configuration yield different advantages and are intended for distinct applications. For instance, as shown in [46], redundant sensor can be used to confirm the detection of an object, while complementary sensors can generate different views to overcome obstructions. The structures are illustrated in Figure 3.3 adapted from [47], showing how each configuration interacts with the environments, which sensors are exploited and what is intended with each composition.

In order to establish communication between different sensors, it is necessary to establish a protocol. A number of schemes to achieve such communication are described in [48]:

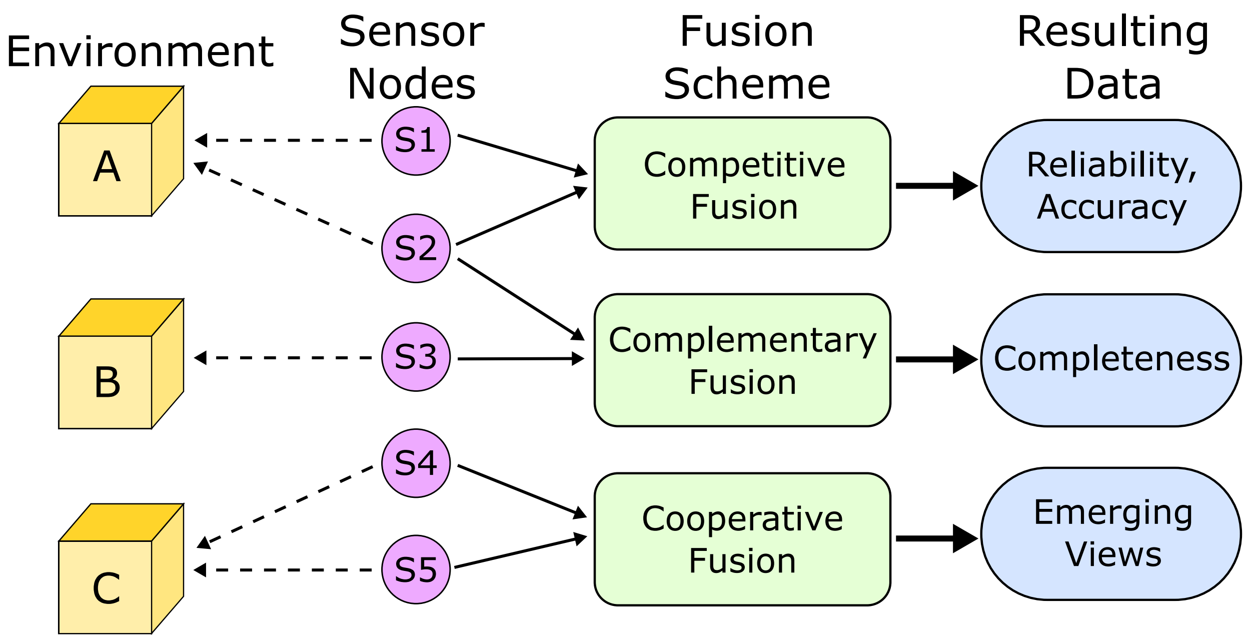


Figure 3.3: Diagram of different sensor fusion implementations.

- **Decentralized communication**: No communication happens between the sensor nodes;
- **Centralized communication**: All nodes provide readings to a central unit;
- **Distributed communication**: The nodes exchange information at a specified rate (e.g., once every five measurements).

Adoption of a centralized scheme allow all sensor data to be fused simultaneously, but may require higher processing resources and power. In contrast, a decentralized approach offloads processing to smaller parts of the sensor array, that can then act independently in case of emergency or malfunctions. However, the lack of communications between nodes can impede the full potential of multiple sensor readings. A compromise can be provided by a distributed procedure, in which sensor communicate in a reduced rate, diminishing the total data load. Nonetheless, this approach need to be validated for each specific application, as in can obstruct performance-critical real-time data processing. A block diagram illustrating this possible structure of sensor fusion communication is presented in Figure 3.4 adapted from [48].

Sensor fusion applications can also be classified in terms of the level in which composition occurs, namely at data level, feature level or decision level. The first strategy fuses data early on, when the raw data is produced in each device. This approach is thus also called early fusion, and is represented in diagram of Figure 3.5, adapted from [49]. Following, feature-level fusion happens after processing raw data into a feature-extraction algorithm. As a higher level approach, it enables a more precise and accurate feature detection, as shown in Figure 3.6 [49]. Lastly, as illustrated in Figure 3.7 [49], decision-making level is performed as a last instance of sensor fusion. Also referred to as late fusion, this strategy takes processed sensor information form sensors and adjudicate the available readings to decide on what action to take.

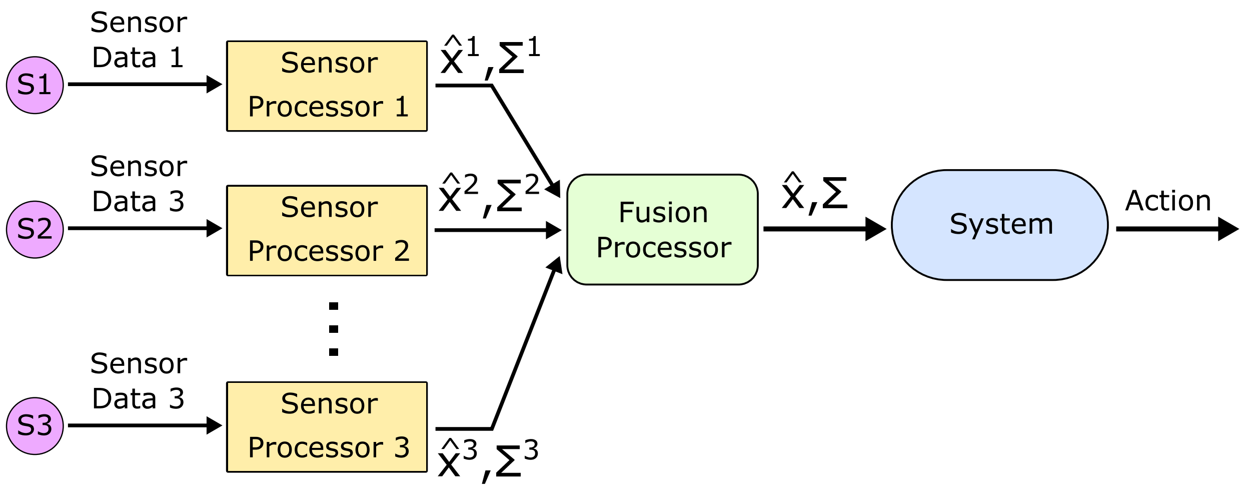


Figure 3.4: Schemes of distributed communication in sensor fusion applications.

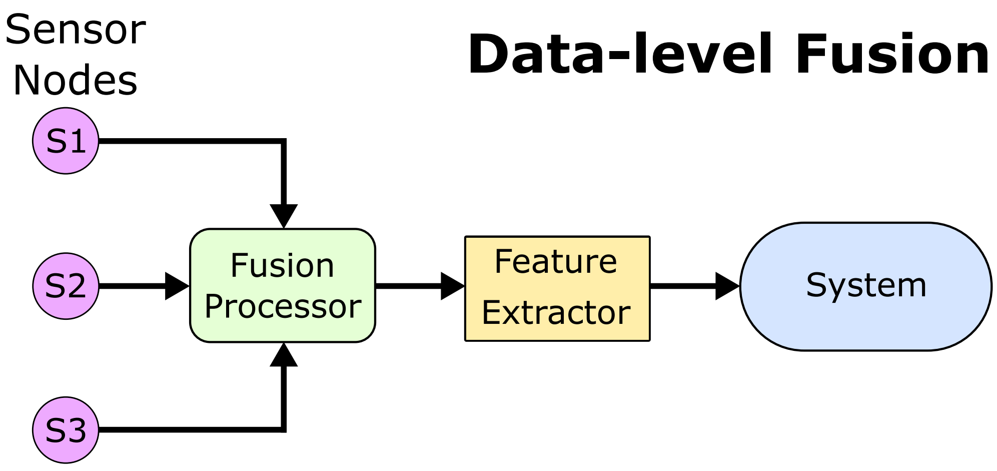


Figure 3.5: Sensor fusion scheme performed at data-level.

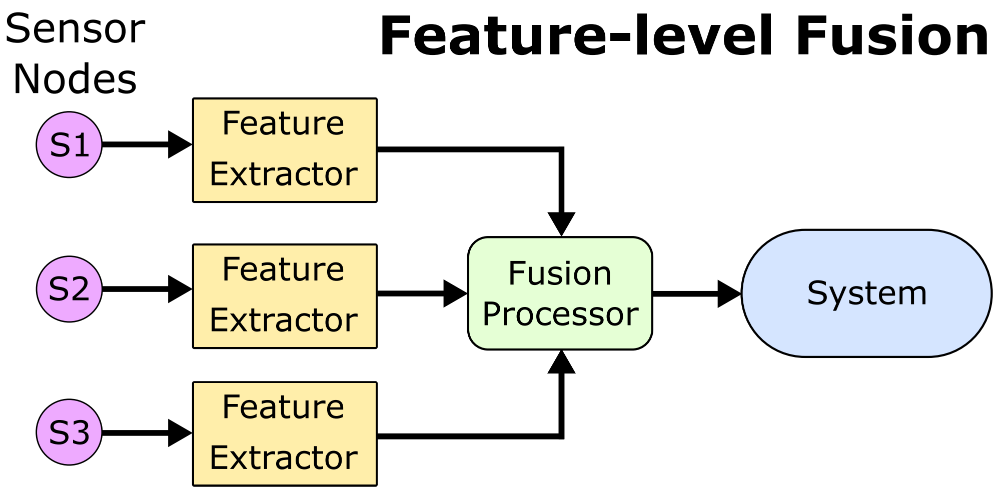


Figure 3.6: Sensor fusion scheme performed at feature-level.

With theses classifications considered, it is possible to setup a wide variety of sensor fusion structures. In this work, we propose an approach that performs early fusion using raw data from a camera and a LiDAR device. This data level strategy is then compared to feature extraction using

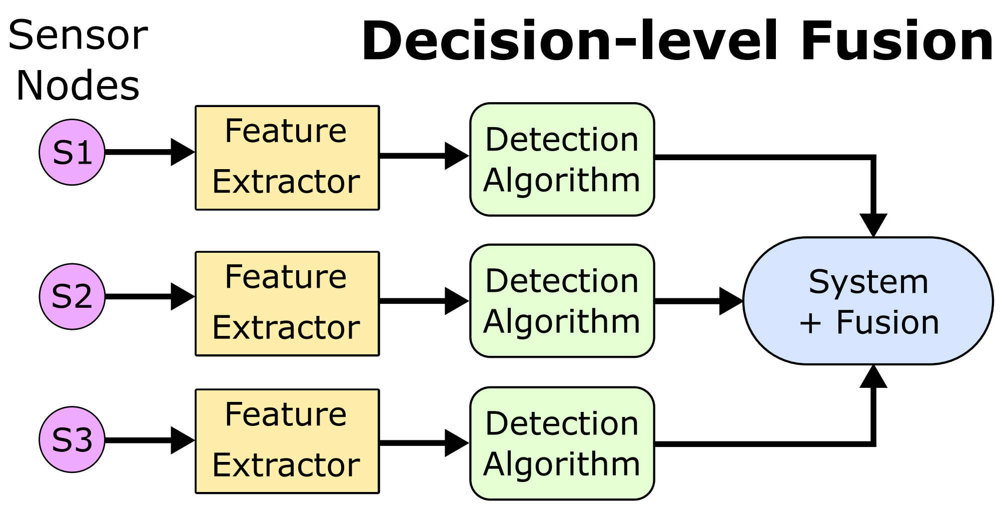


Figure 3.7: Sensor fusion scheme performed at decision-level.

individual sensor readings and a delayed features-level fusion approach.

Some specific challenges arise due to the multinode structure of sensor fusion. We now discuss some concerns that need to be taken into account when projection such applications, as in [35]. Initially, due to the increased data load coming from various sensors, synchronization is crucial for guaranteeing a reasonable performance. As different sensors produce data points as disjoint periods, they also experience processing delay and latency during transmission. This may cause miscommunication or time-blocked processing if not accounted for.

Another open problem is in regards to sensor calibration. As physical devices, the location of the sensor mounted over a vehicle has to be scrutinized. Not taking into consideration a sensor's position in relation to the vehicle and in relation to other sensors used for fusion can render the coordination useless. Thus, no reliable views can be generated without proper calibration.

Further objections are in relation of weighting the value of readings from different sensors. When redundancy is desired, it is necessary to arbitrate sometimes differing measurements of the nodes. Hence, it is not trivial to judge how reliable a given sensor reading is, as it might be malfunctioning or influenced by other detrimental factors. Some voting algorithms are proposed to alleviate this situation, but, given more extreme scenarios, additional weighting policies may prove necessary.

## 3.3 DETECTION ALGORITHMS

In this section, we present the techniques employed to perform the proposed detection tasks in the datasets shown in Section 2.2. For each undertaking, we select the more appropriate techniques according to the state-of-the-art review presented in Section 2.1.

Initially, we consider the pedestrian detection algorithm. A Machine Learning (ML) approach from a Histogram of Oriented Gradients (HOG) feature extractor is chosen. For classification, the extracted features are used to train and test a Support Vector Machine (SVM) classifier. We also evaluate the use of a boosted decision-tree method using the XGBoost framework.

For traffic sign recognition, we deemed that a Convolutional Neural Network (CNN) is better fit for sequential image recognition. A baseline non-optimized neural network is built to identify image samples in the GTSRB. In order to gauge possible gains in performance, we apply optimization techniques in a number of different CNN instances.

In the sensor fusion research area, we propose a framework to enable raw camera and LiDAR data to be fused at data-level, thus an early fusion. For that, we are tasked with making the different readings from the sensors match in dimensions and mapping. This is expected to increase the algorithm's performance over the KITTI dataset in comparison to the other implementations.

### 3.3.1 Histogram of Oriented Gradients

The authors of [31] propose the approach to characterize objects by looking at the distribution of local intensity gradients or edge directions. This approach consists in dividing a image into smaller spatial regions, called cells. From then, an unidimensional histogram of gradients and edge orientations is accumulated over each cell's pixels and normalization is performed over larger spatial regions, called blocks. This process results in the Histogram of Oriented Gradient (HOG) descriptors of a given image. As an example, the HOG of an image from the PSU dataset used in our algorithm, described in Chapter 4, is illustrated in Figure 3.8.

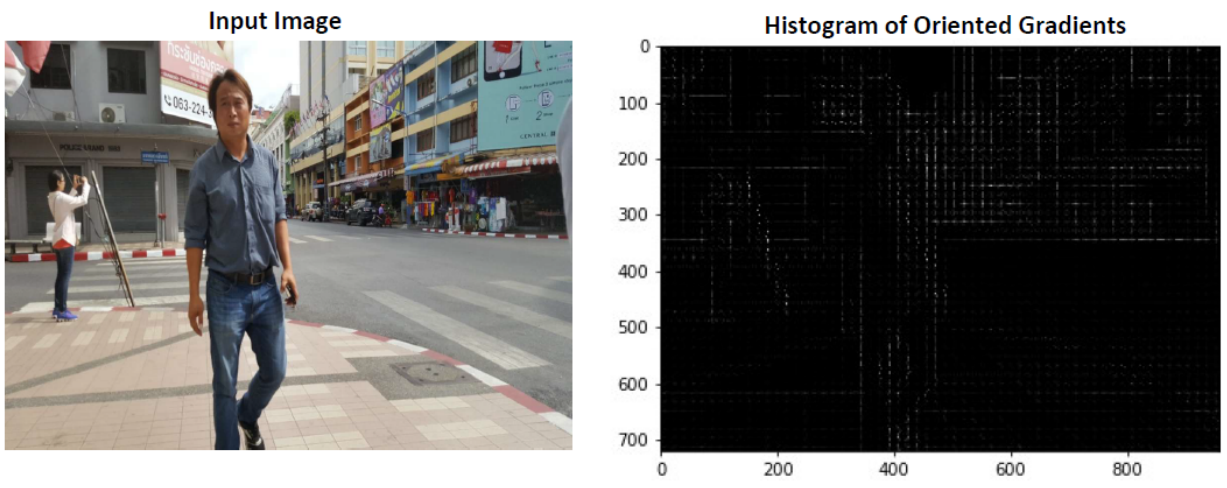


Figure 3.8: Sample input image from the PSU dataset and its resulting HOG.

Based on [50], we describe the mathematical modeling of a gradient. We first define the window $I$ from the input gray-level image. Given two components $I_x$ and $I_y$ of the gradient of $I$, we approximate them using central differences:

$$I_x(r,c) = I(r,c+1) - I(r,c-1) \qquad \text{and} \qquad I_y(r,c) = I(r-1,c) - I(r+1,c), \tag{3.14}$$

where $r$ is the vertical length and $c$ is horizontal length of the calculation interval. From Equation

3.14, we convert the components onto polar coordinates, restricting angles from 0 to 180 degrees in order to identify points in opposite directions. Thus:

$$\mu = \sqrt{I_x^2 + I_y^2} \qquad \text{and} \qquad \theta = \frac{180}{\pi}(tan_2^{-1}(I_x, I_y)\text{mod}\pi), \tag{3.15}$$

where $tan_2^{-1}$ is the four-quadrant inverse tangent, guaranteeing values between $-\pi$ and $\pi$. Next, the window $I$ is divided into adjacent cells of size $C \times C$ pixels. In each one of the cells, the gradients of Equation 3.15 are calculated forming a histogram with $B$ bins. Each of the bins is given a width of $w = 180/B$, such that each $B_i$ bin has its boundaries as $[w_i, w(i+1))$ and its center at $c_i = w(1 + 1/2)$. Thus, a pixel with magnitude $\mu$ and orientation $\theta$ contributes to two different bins in depending on its position in relation to its boundaries, as shown in [50].

Next, the algorithm performs a block normalization step. Cells are grouped in pairs, forming larger $2C \times 2C$ overlapping blocks. Then, the histograms of consecutive horizontal or vertical blocks are concatenated into a block feature $b$. The feature $b$ is then normalized by its norm, as:

$$b \leftarrow \frac{b}{\sqrt{||b||^2 + \epsilon}}, \tag{3.16}$$

where the very small constant $\epsilon$ prevents division by zero in null-valued gradient blocks. Block normalization looks to reduce effects of changes in contrast in the same objects whilst sacrificing some information of the overall gradient magnitude.

Following from Equation 3.16, the normalized block features of all blocks are concatenated. This forms a HOG feature vector $h$, also normalized as follows:

$$\begin{aligned} h &\leftarrow \frac{h}{\sqrt{||h||^2 + \epsilon}}, \\ h &\leftarrow \min(h_n, \tau), \\ h &\leftarrow \frac{h}{\sqrt{||h||^2 + \epsilon}}. \end{aligned} \tag{3.17}$$

In Equation 3.17, the term $\tau$ represents a set positive threshold that removes $h$ features greater than it. This is done to prevent that very large features overwhelm the rest of the image, degrading information. After this step, the result is the corresponding HOG, with features independent of overall image contrast.

### 3.3.2 Support Vector Machines

Support Vector Machines (SVM) are a statistical model used to separate clusters of points of different classes. According to [51], we now provide a brief mathematical overview, focusing on the kernel SVM simplification. The desired optimal hyperplane and the employed support vectors are illustrated in Figure 3.9.

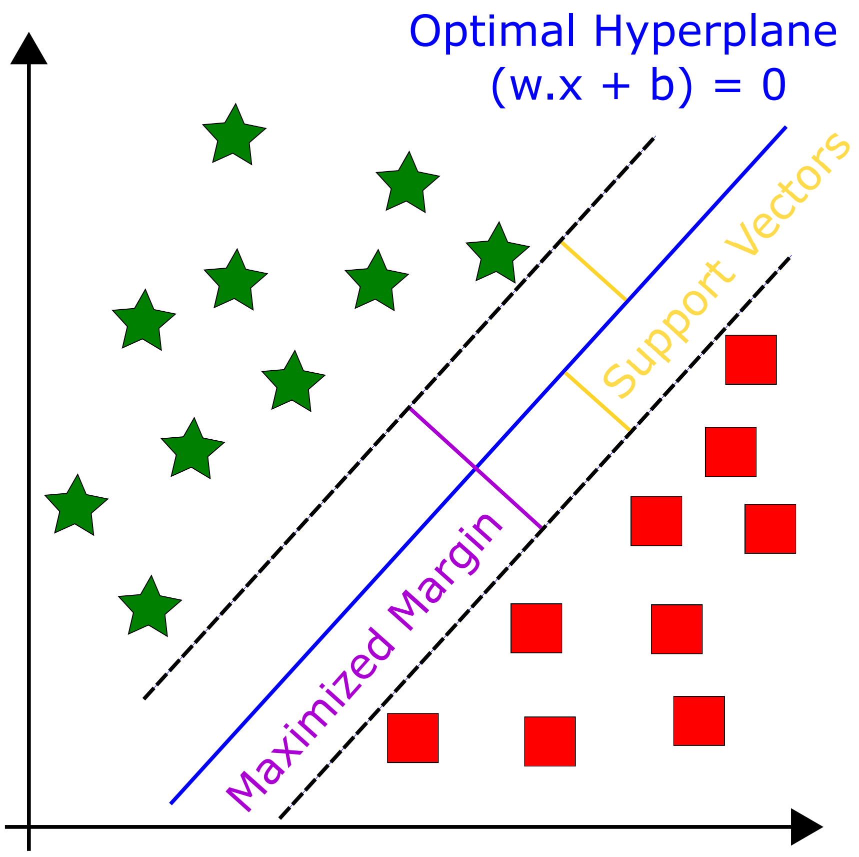


Figure 3.9: Illustration of the hyperplane optimization result and the support vectors used in the process.

In the context of classification tasks, [51] describes the task as estimating a function $f : \mathbb{R}^N \to \{\pm 1\}$ in an $N$-dimensional set of $x_i$ data points with label $y_i$, such that:

$$(x_1, y_1), ..., (x_l, y_l) \in \mathbb{R}^N \times \{\pm 1\}, \tag{3.18}$$

so the classification function is able to correctly identify $y$ from $x$, that is $f(x) = y$. From the training data, it is established a given probability distribution $P(x, y)$. By not restricting the class of functions $f$ can assume, we cannot predict new classes outside the training data. In this scenario, no learning function is achieved. The authors state that restricting the class of functions implemented is essential to reach machine learning, with due capacity for the data amount.

It is proposed in [51] that the hyperplanes class of functions are used, defined as:

$$(w \cdot x) + b = 0, w \in \mathbb{R}^N, b \in \mathbb{R}, \tag{3.19}$$

which will correspond to the classification function:

$$f(x) = \text{sign}((w \cdot x) + b), \tag{3.20}$$

where $x$ is the data point under decision. By then optimizing the hyperplane in Equation 3.19, we can reach a decision function that better separates two classes of objects.

The optimized class separation is then reached by solving a constrained quadratic optimization problem, starting from expanding $w = \sum_i v_i x_i$, from Equation 3.20, in terms of a subset of training patterns described in further detail in [51]. Thus, the name "support vectors" is given to these training patterns in the context of classification. Thus a Support Vector Machine is tasked with mapping input the data into a different dot product space $F$ — named the feature space — via a nonlinear map:

$$\Phi : \mathbb{R}^N \to F, \tag{3.21}$$

and afterwards perform the optimization problem described in Equation 3.20 over $F$. This optimization algorithm depends only on dot products, reducing the classification function, equivalent to Equation 3.20, in the new space, using the assignment operator :=, in such manner:

$$k(x, y) := (\Phi(x), \Phi(y)). \tag{3.22}$$

As the left side of Equation 3.22 may be costly to compute in higher dimensionality, we can determine a kernel $k$ that provides a simpler computation. Such kernel can be used in different forms, such as:

1. Polynomial Kernel function:
$$k(x, y) = [x.y + 1]^d;$$
2. Radial basis kernel function:
$$k(x, y) = exp|x - y|^2/\sigma^2;$$
3. Sigmoid kernel function:
$$k(x, y) = tanh(\alpha x^T y + c).$$

By employing a determined kernel $k$ in Equation 3.22, we are able to construct a classifier algorithm using the SVM. This is achieved by taking each training data point $x_i$, and using its correspondent projection in the feature space $\Phi(x)$ and perform hyperplane optimization over the space $F$. Thus, the use of kernels and decomposition of the hyperplane parameter $w$, leads the corresponding nonlinear classification function in such form:

$$f(x) = \text{sign}(\sum_{i=1}^{1} v_i k(x, x_i) + b), \tag{3.23}$$

in which parameters $v_i$ are calculated according to the quadratic problem stated in [51]. In the input space, the hyperplane is equivalent to Equation 3.23. Its form is the determined by the kernel chosen.

Thus, by choosing the optimized hyperplane for a given kernel, SVM is able to separate the input data clusters into two different classes. It is then trivial to label an unknown test data input, as it reduces to locating it in relation to the hyperplane and attributing it to the nearest class.

### 3.3.3 XGBoost

Based on the tree boosting ML algorithm, XGBoost is proposed in [52]. According to authors, the technique is intended to be a scalable solution for classification problems and further data science applications. The novelties brought by the method include a tree learning algorithm for treating sparse data, a quantile sketch procedure for efficient calculations with weights in tree learning, an algorithm for parallel tree learning that is aware of the sparsity of data and further optimizations that lead to better calculation performance.

The authors in [52] state a regularized learning task using the tree structure. From a given group of input data with $n$ samples and $m$ features, such that $\mathcal{D}\{(x_i, y_i)\}(|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$, a tree ensemble model utilizes $K$ additive functions $f_k$ in order to predict the corresponding label $y_i$. Thus:

$$\hat{y}_i = \Phi(x_i) = \sum_{k=1}^{K} f_k \in \mathcal{F}, \tag{3.24}$$

where $\mathcal{F}\{f(x) = w_{q_x}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ is called the space of regression trees. In turn, from the expression defining $\mathcal{F}$, the parameter $q$ expresses the structure of each tree, responsible for mapping an input into a corresponding leaf of a total of $T$ leaves in the tree. Thus, each function $f_k$ corresponds to a different tree structure $q$ with leaf weights $w$. The remainder of the problem is exposed in more detail in [52].

From this problem statement, XGBoost comes as a ensemble learning technique that looks to reduce lookup times into the generated trees by using a technique called Gradient Boosting, according to [53]. In a simplified manner, XGBoosts exploits both the Gradient Descent process and Gradient Boosting. The former is based on a determined cost function, that looks to represent how well predicted the values in a learning tree are. Since the predicted values must be as close to the actual values as possible, we look to diminish the cost function. The Gradient Descent techniques thus looks to set the weight parameters $w$ in the regression trees such that the cost function is minimized. By iterating over the parameters from initial weights, the algorithm adjusts and optimized the weights so as to improve the classification of a given training dataset.

The other technique exploited is Gradient Boosting. It consists of creating an ensemble of weak learners that "boost" the data points misclassified by the learning tree, attributing to them greater $w$ weight values. Thus, by later combining these weak learners, a strong learner is formed. In XGBoost models, the learning trees are built sequentially. Thus, by then averaging the weighted sum of values achieved by each tree instance, achieving a better classifier. This is possible by using Gradient Descent — responsible for adjusting the weights in an iterative manner — together with a booster algorithm with well-adjusted parameters.

## 3.4 CONVOLUTIONAL NEURAL NETWORKS

For the following description, we assume knowledge of a basic neural network structure, as detailed in [54]. In a general description, a neural network is composed of neuron units that produce a sequence of real value activation. The activation can arise from external input such as sensor data. By connecting each neuron, the network form is capable of propagating the activation, causing actions. The potential learning action of the network is to stipulate weights to each of the network's inputs so that the overall system results in the desired behaviour.

In this context, for the subsequent task of traffic sign recognition, we reckon that using Convolutional Neural Networks (CNN) is more appropriate to the application at hand. In this section, we discuss the basic structure of a CNN for image recognition, including its layers and relevant parameters. This deep learning model is based around three layers in which the recognition is performed, namely: convolutional layers, pooling layers and fully connected layers. The parameters pertaining to the general algorithm are also present in this section. The optimization-specific parameters are postponed for the methodology in Chapter 4. The following conceptual and mathematical description is provided on the context of image recognition, based on [55], [56] and [57].

### 3.4.1 Convolutional Layer

From the description of a digital image given in Subsection 3.1.1, represented by a $n_1 \times n_2$ matrix of intensity, we now describe the two dimensional convolution. The process of convolution consists of merging information of two input functions together, achieving filtering for example. In the context of CNN, performing 2D convolution over an image produces a given feature map, describing how well the input is described by the set filter.

We now explore the filtering functions in CNN. Given an input image, denoted $I$, it is filtered by filter $K$ so that $K \in \mathbb{R}^{2h_1^{(l)}+1 \times 2h_2^{(l)}+1}$, resulting in:

$$(I * K)_{r,s} := \sum_{u=-h1}^{h1} \sum_{u=-h2}^{h2} K_{u,v} I_{r-u,s-v}. \tag{3.25}$$

The filter $K$ is defined as the matrix:

$$K = \begin{bmatrix} K_{-h_1,-h_2} & \cdots & K_{-h_1,h_2} \\ \vdots & K_{0,0} & \vdots \\ K_{h_1,-h_2} & \cdots & K_{h_1,h_2} \end{bmatrix}. \tag{3.26}$$

Then, the convolutional layer is tasked with detecting local combinations of features coming from previous layers through 2D convolution. They then map such features in an appropriate output. The action in convolutional layers is achieved by dividing the input image into perceptrons. These regions are then compared to a local filter of size $m_2 \times m_3$ and the results are put into a feature map by compressing the perceptrons. These local spatial filters are thus trained according to the position in the image where they are employed.

Each convolutional layer has a bank of $m_1$ filters. The number of filters applied on a given layer is proportional to the depth of the the volume of output feature maps [57]. As each filter detects one specific feature on the input image, the output $Y_i^{(l)}$ of a given layer $l$ is the corresponding $m_1^{(l)}$ feature maps with size $m_2^{(l)} \times m_3^{(l)}$. The $i^{\text{th}}$ feature map, denoted $Y_i^{(l)}$, is given by:

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} * Y_j^{(l-1)}, \tag{3.27}$$

where $B_i^{(l)}$ is the bias matrix and $K_{i,j}^{(l)}$ is the filter of size $2h_1^{(l)} + 1 \times 2h_2^{(l)} + 1$ that connects the $j^{\text{th}}$ feature map in layer $(l-1)$ to the $i^{\text{th}}$ feature map in layer $l$.

### 3.4.2 Pooling Layer

Another key part of the CNN is the pooling layer. This layer is responsible for decreasing spatial dimensions in between convolutional layers. This is done in order to reduce parameter amount, simplify computation and avoid over-fitting.

Given a pooling layer $l$, we consider its main hyperparameters as the spatial extent of the filter $F(l)$ and its stride $S(l)$. From the previous layer's feature map, with volume size $m_1^{(l-1)} \times m_2^{(l-1)} \times m_3^{(l-1)}$, the pooling layer outputs a corresponding map of size $m_1^{(l)} \times m_2^{(l)} \times m_3^{(l)}$ in which:

$$\begin{aligned} m_1^{(l)} &= m_1^{(l-1)}, \\ m_2^{(l)} &= (m_2^{(l-1)} - F_{(l)})/S_{(l)} + 1, \qquad (3.28) \\ m_3^{(l)} &= (m_3^{(l-1)} - F_{(l)})/S_{(l)} + 1. \qquad (3.29) \end{aligned}$$

The focus of this layer is to reduce spatial information in a window of size $F_{(l)} \times F_{(l)}$ to one value. Thus, by sequentially moving the filter window by $S(l)$ positions in the array for each step, the entire input is successfully spatially reduced.

### 3.4.3 Fully Connected Layer

The resulting network feature maps from the given layer architecture are fed into the fully connected layer, described as a multilayer perceptron. In turn, this layer is responsible for turning the $m_1^{(l)} \times m_2^{(l)} \times m_3^{(l)}$ feature map into a class probability distribution. Therefore, the output of said layer consists of $m_1^{(l-i)}$ neurons, in which $i$ represents the number of layers in the multilayer perceptron.

Given a $l-1$ fully connected layer, the result from an $m_1^{(l)} \times m_2^{(l)} \times m_3^{(l)}$ feature map is denoted as:

$$y_i^{(l)} = f(z_i^{(l)}) \quad \text{with} \quad z_i^{(l)} = \sum_{j-1}^{m_1^{(l-1)}} w_{i,j}^{(l)} \cdot y_i^{(l-1)}. \tag{3.30}$$

Otherwise:

$$y_i^{(l)} = f(z_i^{(l)}) \quad \text{with} \quad z_i^{(l)} = \sum_{j-1}^{m_1^{(l-1)}} \sum_{r-1}^{m_2^{(l-1)}} \sum_{s-1}^{m_3^{(l-1)}} w_{i,j,r,s}^{(l)} \cdot (Y_i^{(l-1)})_{r,s}. \tag{3.31}$$

Hence, the main assignment is to adjust the weight parameters $w_{i,j}^{(l)}$ and $w_{i,j,r,s}^{(l)}$, from Equations 3.30 and 3.31, respectively. By tuning the weights appropriately, the fully connected layer creates a statistical distribution for each of the prediction classes based on the activation maps generated through the network's layers.

### 3.4.4 Activation Function

The use of a nonlinear activation function is deemed crucial to correctly classify objects when multiple classes are present. Some of the most commonly used activation functions are the sigmoid function and hyperbolic tangent function. Rectified Linear Units (ReLU) is a nonlinear activation function that also combines rectification, as introduced in [58]. Thus, it is able to efficiently propagate the gradient, even in deep networks. As a simple implementation, the ReLU activation is also widespread in CNNs.

### 3.4.5 Loss Function

By selecting the correct loss function of the network, we are able to express the cost of predictions made by the network. This way, the learning model look to minimize this loss by increasing the correct prediction rate. For our application, we choose to calculate the softmax loss function to evaluate the performance classification in our network, in accordance with [59], given by:

$$L_{softmax} = -\frac{1}{N}\sum_i \log(L_i) = -\frac{1}{N}\sum_i \log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_{y_i}}}\right), \tag{3.32}$$

where $f_j$ is the $j^{\text{th}}$ element of the class output vector of the final fully connected layer, and $N$ is the number of training samples. A higher recognition rate results from reducing the value of $L_{softmax}$, in the .

### 3.4.6 Learning Rate

To obtain the optimal parameters in the process of CNN learning feature, an adaptive learning rate is utilized to minimize the chosen loss function as in Equation 3.32. This optimization problem is tackled by tuning local network parameters of weight $w$ and bias $b$. According to [60] the normalized update process for these parameters is thus:

$$w_{t+1} = w_t - \eta \frac{\overline{m_t}}{\sqrt{\overline{v_t}} + \epsilon}, \tag{3.33}$$

$$b_{t+1} = b_t - \eta \frac{\overline{m_t}}{\sqrt{\overline{v_t}} + \epsilon}, \tag{3.34}$$

in which $\eta$ is a learning rate, which describes the updated amplitude of $w$ and $b$, affecting its convergence rate, and $\epsilon$ is a small constant to avoid dividing by zero. Further, hyperparameters are also considered in order to achieve a reasonable learning convergence in the network.

In a number of classifiers, some of the extracted features during training may not be appropriate for the classification of specific targets. In the CNN model, the feature maps enable training of the network from a large amount of data. This process is illustrated in Figure 3.10. It consists of a forward propagation task, used to calculate the networks weights and loss functions. Next, the back propagation stage adjusts the parameters in order to reduce the network's loss.

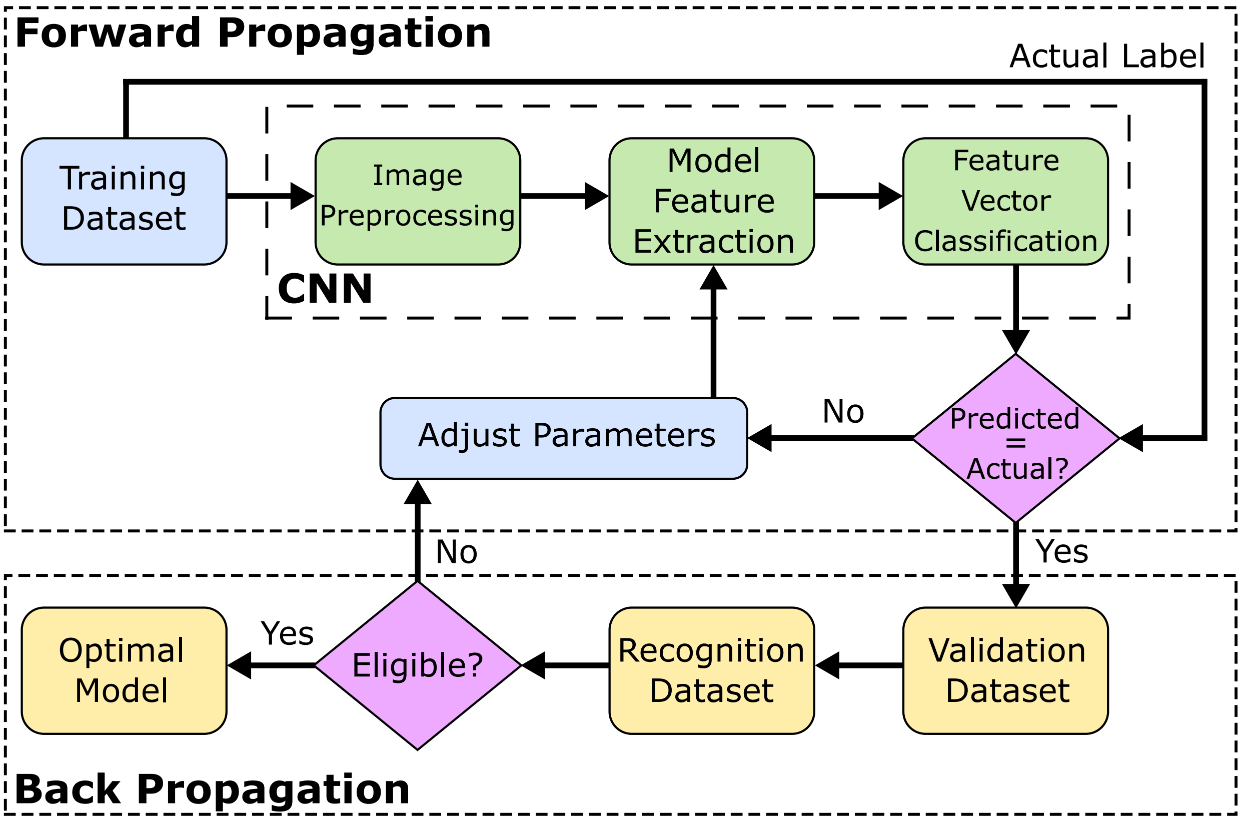


Figure 3.10: Flowchart of a CNN training process.

## 3.5 YOLO OBJECT DETECTION ALGORITHM

By using the basis of CNN presented in the previous section, it is possible to extract features by train a neural network using a given dataset. CNN-based object detection algorithms can be divided into two types: region-based and single regression. In region-based algorithms, a Region of Interest (ROI) is proposed, and ROI objects are classified using CNNs. In single regression, region proposal and classification are combined into one stage.

Aimed at excelling in real-time detection scenarios, the You Only Look Once (YOLO) algorithm is under active development. By approaching detection problems in terms of regression, YOLO aims at simplifying the task at hand. Unlike region-based and sliding window-based techniques, YOLO considers the whole image during training and test stages. It is thus able to encode information regarding classes and their appearance contextually.

### 3.5.1 YOLO V3

Used in this work, YOLO V3 [61] is able to perform detection tasks in various scales, in a robust feature extractor network. The model of the algorithm is composed by a feature extraction — also known as backbone — and either a classification or detection module — known as head. The YOLO feature extractor, known as Darknet-53, comprises 53 layers with residual or skip connection to support activation of neurons in order to propagate the gradient without descent throughout the deeper network layers.

The detector module consists of an $S \times S$ grid cells of a given size, depending on the image's scale. For a given object, if its center falls inside a grid cell, it is then tasked with detecting teh corresponding object. Each grid cell has to predict three bounding boxes and its respective confidence scores. These scores represent how certain the algorithm is if the resulting box contains an object. Mathematically, it is defined as:

$$\mathrm{Confidence} = \mathrm{Pr(Object)} \cdot \mathrm{IoU}_{\mathrm{truthpred}}, \tag{3.35}$$

where $Pr(Object)$ is the probability that the cell contains an object and $IoU_{\text{truthpred}}$ is the intersection over union (IoU) of the predicted and ground truth boxes. Graphically, the IoU operation can be seen as a logic operation over the image and the bounding box, as shown in Subsection 3.1.1.

Each grid cell is also assigned three anchor boxes. Each bounding box consists of its coordinates $x$ and $y$, the box dimension width $w$ and height $h$ and the detection confidence. Since a large variance of scale and aspect ratio of boxes is often observed, anchor boxes are used in a predefined fixed box aspect ratio. A total of 9 anchors boxes are available in YOLO, 3 for each scale, from small to large. Its coordinates ($t_x$, $t_y$) map the box's center relative to the center of the anchor boxes. As they are fixed in place, the centroid of each of the anchor boxes correspond to the same cell. The grid cell are used to predict the conditional class probabilities for a number of

$C$ classes, denoted $\mathrm{Pr(Class}_{i=1,\ldots,C}|\mathrm{Object)}$. Each of the probabilities are conditioned to the cell predicted to contain the detected object. Thus, only one probability value is calculated for each lass, regardless of the number of $B$ boxes present. From the resulting probabilities, we calculate the box confidence score as follows:

$$\mathrm{Pr(Class}_{i=1,\ldots,C}|\mathrm{Object)} \cdot \mathrm{Pr(Object)} \cdot \mathrm{IoU}_{truthpred} = \mathrm{Pr(Class}_i) \cdot \mathrm{IoU}_{truthpred}, \tag{3.36}$$

resulting in the class-specific confidence values. This score expresses both the probability that a given object class is present in the detection box and how confident the system is that said object belongs to that class.

Regarding the loss function, YOLO implements a custom expression. It consists of the total sum of four losses: $xy$ loss, $wh$ loss, loss of the object and loss of the class. It thus looks to minimize errors both in location of predictions and in the detection task. YOLO also performs postprocessing over the results using the non-max suppression technique. This is performed by removing the boxes that do not have the highest prediction scores. Thus, it avoids duplicate predictions by keeping only the most probable detection boxes for each class.

# 4 METHODOLOGY

In this chapter, we present the algorithms proposed for this work. First, we pose a framework to improve pedestrian detection in occluded or obstructed scenarios consisting of a Histogram of Oriented Gradients (HOG) system combined with a Support Vector Machines (SVM) learning model. An eXtreme Gradient Boosting (XGBoost) learning model is also proposed for comparison. A part of this proposal is published in [62] with participation of the author.

Following, we explore optimization techniques presenting a Convolution Neural Network (CNN) for traffic sign recognition. By performing improved image preprocessing, feature extraction and classification, we look to improve on the accuracy and loss rate of the base CNN model.

Concluding the chapter, we demonstrate how early sensor fusion techniques can be applied on camera and LiDAR data to improve precision of YOLO-based object detection. By using a three-channel color camera and projected LiDAR points cloud, we apply a YOLO v3 detection algorithm to evaluate precision gains over individual sensors and a feature-level fusion implementation. Likewise, this framework is published in [63] with participation of the author.

## 4.1 PEDESTRIAN DETECTION USING HOG SVM AND XGBOOST

In order to recognize the presence of pedestrians in input images, we propose two architectures, both utilizing the HOG feature extractor. In the first propose model, we pair the HOG features with a Kernel SVM. Next, for the second architecture, we use an XGBoost classifier together with HOG.

On the PSU dataset described in Subsection 2.2.1, we apply a 20/80 training-test split. Thus, we use 80% of the available images along with the respective labels representing the presence or absence of a pedestrian in the scene for training each model. By extracting the image features using HOG, we feed this into each learning algorithm, forming two trained classifiers. We then test the resulting models by feeding the remaining images without the labels into the HOG extractor and into each one of the classifiers. By comparing the predicted labels to their true values, we are able to calculate performance metrics and each respective confusion matrix. The overall structure of the generation process for the SVM and XGBoost classifiers is illustrated in Figure 4.1.

### 4.1.1 Histogram of Oriented Gradients with Support Vector Machine

First, we propose the use of HOG feature extractor with a Kernel SVM Learning algorithm for the task at hand. By correctly choosing among the available kernels, we are able to improve the

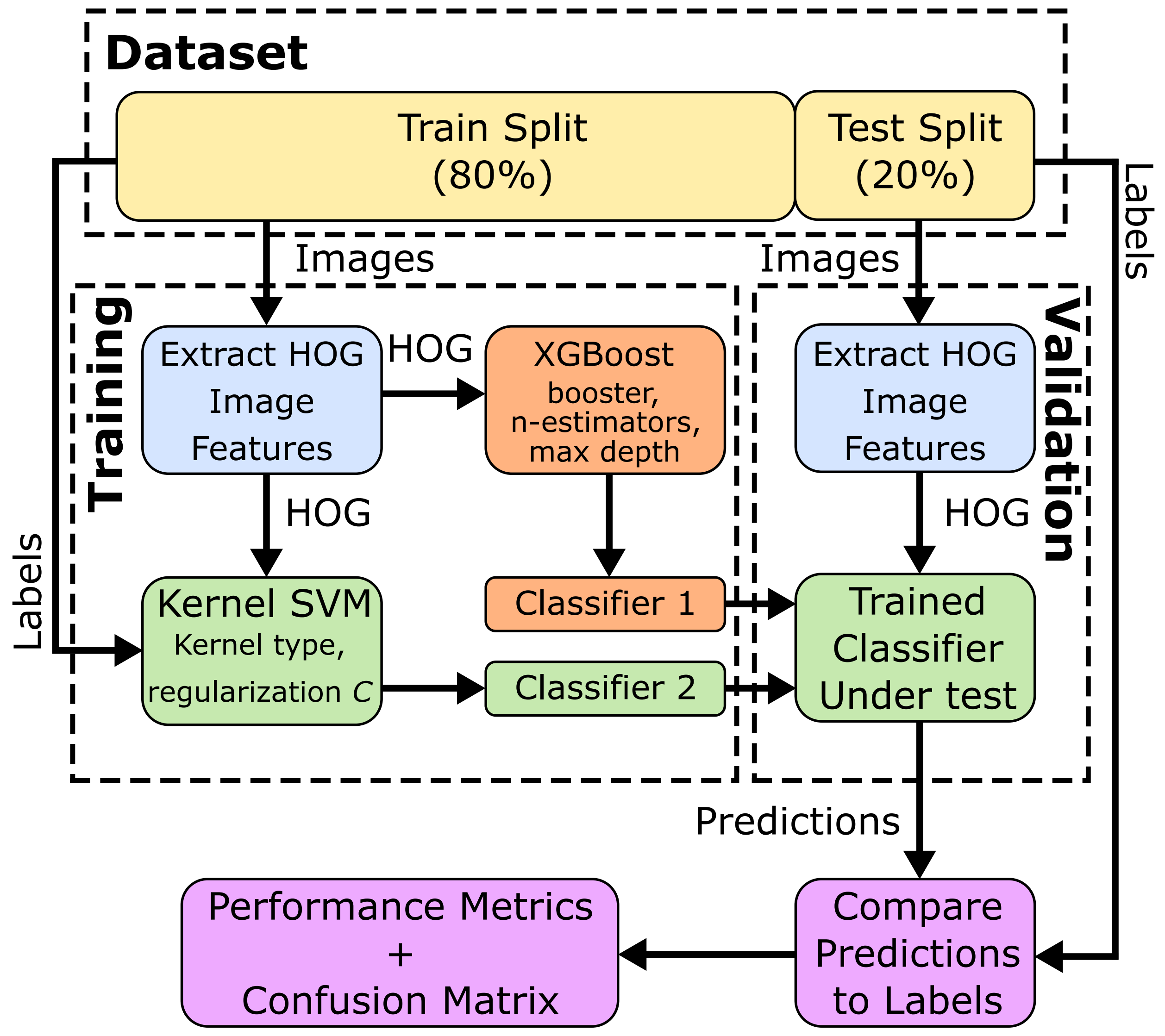

Dataset
Train Split (80%)
Test Split (20%)
Labels
Images
Images
Training
Validation
Extract HOG Image Features
HOG
XGBoost booster, n-estimators, max depth
Extract HOG Image Features
HOG
HOG
Labels
Kernel SVM Kernel type, regularization C
Classifier 1
Classifier 2
Trained Classifier Under test
Predictions
Performance Metrics + Confusion Matrix
Compare Predictions to Labels

Figure 4.1: Flowchart for training and validation of the proposed pedestrian detection classifiers.

results from the SVM algorithm. We experimented with different kernel SVM implementations, namely the polynomial, radial basis function (RBF), and sigmoid. However, after deliberating this question, we decided to use a linear kernel due to superior performance in relation to the other options.

Kernel SVM enables us to optimize the hyperplane division between pedestrian and no pedestrian classes. Looking for further optimization, we tune the hyperparameters of the SVM model by applying the grid search technique. This technique enabled us to improve algorithm performance by refining the regularization parameter $C$. The strength of the regularization is inversely proportional to $C$ and it must be strictly positive.

### 4.1.2 Histogram of Oriented Gradients with XGBoost

Next, the following proposed architecture pair the HOG feature extractor with an XGBoost learning algorithm. Again, we perform tuning of hyperparameters through grid search. We adjusted the *booster*, learning rate, *n*-estimators, and maximum depth of the decision tree implemented.

The tuned parameter *booster* indicates the type of booster used in the model. Options range using a version of regression tree as a weak learner, a linear regression function or using the dropout method over the boosted regression trees. We also set the maximum depth of a tree, so as to avoid overfiting to occur in the model. Finally, *n*-estimators determines the number of trees that should be fitted to.

### 4.1.3 Algorithm Performance Metrics

For validation, we compare our results with the benchmarks achieved in [32]. Besides that, we also calculate further performance metrics, as presented in [64]. Namely, we calculate the accuracy, precision, recall and F1 score. The metrics are calculated according to the following equations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.3}$$

$$\text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4.4}$$

In Equations 4.1 to 4.4, the parameters TP, FN, FP and TN represent the the number of true positive detections, false negative detections, false positive detections and true negative detections, respectively. By calculating these metrics, we believe a better evaluation of our proposed model is provided.

## 4.2 OPTIMIZATION OF CNN-BASED ALGORITHM FOR TRAFFIC SIGN RECOGNITION

For the task of traffic sign recognition, we develop a CNN architecture based on a modified version of VGG Net-D, consisting of 16 convolutional layers, as shown in [65]. This base architecture extracts features from input images by using convolution kernels in a sliding filter approach, by reducing dimension by pooling. The loss factor the whole CNN is then computed in forward propagation. We employ the Adam optimizer technique in order to minimize loss in step of back propagation. To enhance performance, we tune the CNN parameters and activation function. The structure of our Traffic Sign Recognition (TSR) CNN is illustrated in Figure 4.2.

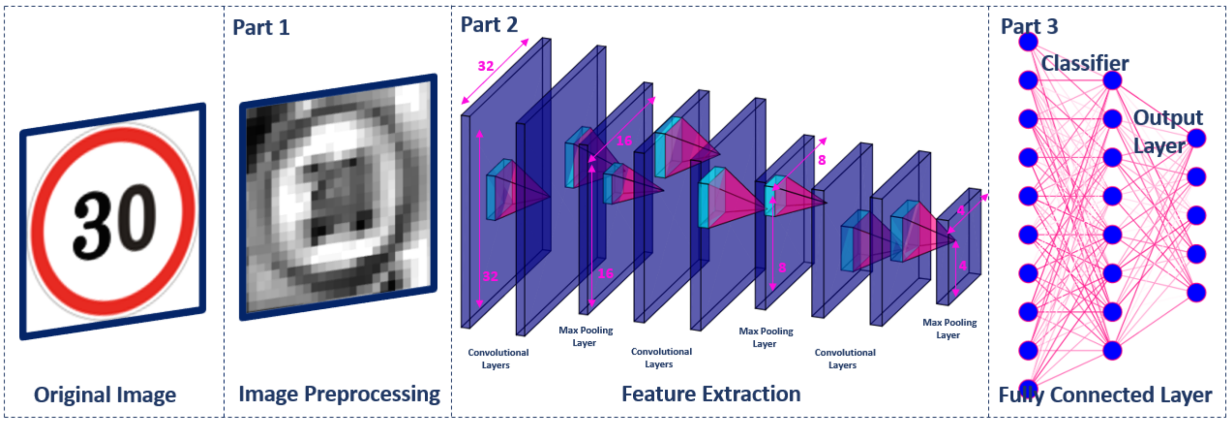


Figure 4.2: Base model stages performed over the TSR-CNN structure.

TSR-CNN includes three parts: an image preprocessing stage, a feature extractor and a classification stage.

Images are pre-processed for reducing the complexity and increasing the accuracy of the applied algorithm. The GTSRB dataset has color images with different sizes. Hence, to feed images into TSR-CNN, they need to be standardized and simplified. Images are resized to 32 by 32 pixels and converted into grayscale to reduce computational resources required. Furthermore, a histogram equalizer method is applied to improve the contrast in the image by stretching out the intensity of the range.

Following, the feature extraction process feeds the images into the TSR-CNN. This network consists of a total six convolutional layers (1-6) and three pooling layers (1-3), with corresponding filter sizes of $3 \times 3$ and $2 \times 2$, respectively. As discussed in Section 3.3, feature maps are obtained by convoluting input maps with a different convolutional kernels in each convolutional layer.

In the first convolutional layer, it is generally expected that sharp edges, angles and such can be removed. Features of the input image can be abstracted using multilayer structure through the network. By traversing the network, images are to become smaller in size. However, the resulting features are also expected to increase in significance towards better classification results. TSR-CNN contains one flatten and fully connected layer for completing the final translation of the features map into the class detection probability.

### 4.2.1 Performance Improvement Techniques

We establish a base TSR-CNN model by modifying the VGG Net-D architecture without applying any optimization. This base model serves as a referential to the subsequent networks built upon it. By measuring each their performance, and then comparing it to the baseline, we can quantify the effect of optimization techniques over the recognition accuracy and loss functions.

Next, we describe the CNN optimization approaches taken during the development of our algorithm.

#### 4.2.1.1 Dropout Regularization

By dropping random nodes out of the network, remaining nodes are lead to adapt to the changes carried out in what is called a regularization effect. Thus, this Dropout can be added to the model by appending new dropout layers. The important parameter in this technique is the dropout rate, which expresses perceptually the proportion of nodes to be dropped. Dropout rates can be fixed or variable according to the layers depth its performed. This technique is illustrated in Figure 4.3.

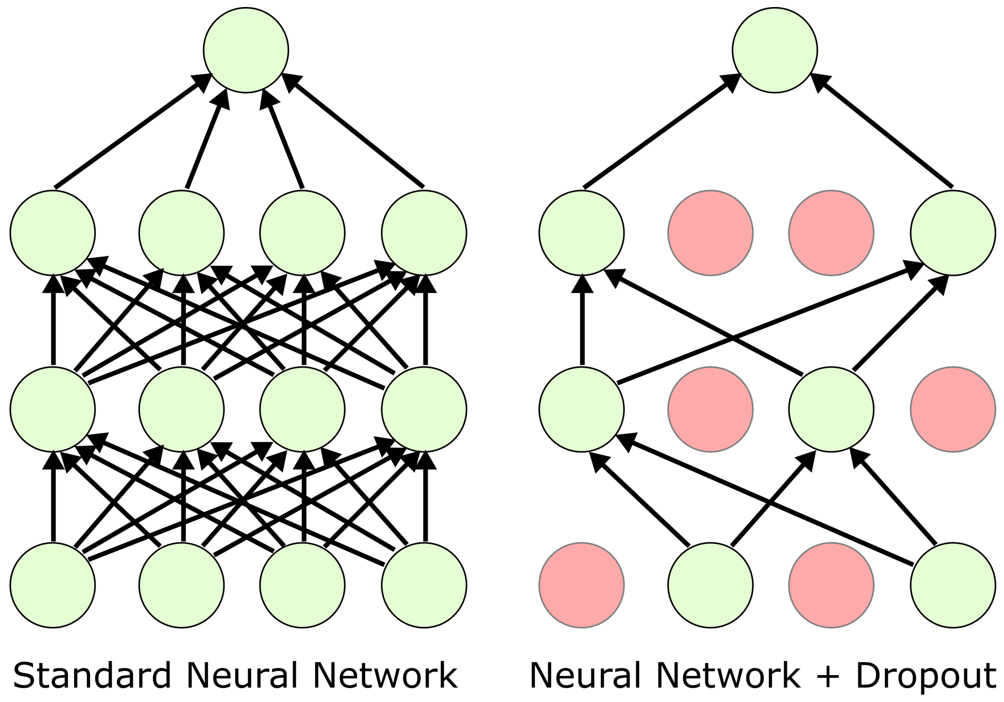


Figure 4.3: The Dropout process of eliminating random layers' nodes.

In this work, we evaluate the performance of using a fixed 20% dropout rate in each max pooling layers of the base model. Alongside that, another network implements a variable dropout rate, increasing its value proportional to the depth of the model. It is expected that the increased dropout forces stronger regularization towards the network's output than in relation to its input.

#### 4.2.1.2 Weight Decay

The weight decay, also called weight regularization, consists of updating the loss function so as to penalize the model proportionally to the size of its weight functions. The regularization effect keeps weights small as the learning algorithm is updated.

It has regularizing effect for keeping the weights small while updating learning algorithm. We can represent the respective loss function, with L2 regularization [66] as:

$$Loss = E + \frac{\lambda}{2}||w_{ij}||^2, \tag{4.5}$$

where $E$ represent loss function and $\lambda$ is the regularization parameter. To observe the effects of weight regularization, we apply a L2 regularizer in all convolutional layers and to the fully connected layers of one of the developed models with a fixed weighting value of 0.001.

#### 4.2.1.3 Batch Normalization

The use of batch normalization in a CNN model is intended to automatically standardize the inputs to each of the layers, as explained in [67]. It results in output data with zero mean and standard deviation of one. The expected effects of performing batch normalization an acceleration in the model training time along with better performance. We can express the batch normalization effect, according to [68], on a given node as follows:

$$\overline{z_i} = \gamma \frac{z_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{4.6}$$

where, $\mu = \frac{1}{n}\sum z_i$ and $\sigma^2 = \frac{1}{n}\sum (z_i - \mu)^2$ represent the mean and variance of the data, $\epsilon$ is a constant introduced to control the equation's numerical stability, $\gamma$ is a scaling factor and $\beta$ is shift factor.

We develop three test instances built upon the base model including a total of 7 batch normalization layers, which are placed after every activation layer and after the full connected layer.

### 4.2.2 Summary of TSR-CNN models

We develop a total of six different models from the base reference TSR-CNN model, applying different optimization techniques over each one. More specifically, the develop models are:

- **TSR-CNN**: no optimization introduced;
- **TSR-CNN-1**: application of a L2 regularizer in all convolutional layers and fully connected layer with weighting value equals to 0.001;
- **TSR-CNN-2**: addition of a dropout layer after each max pooling layer using a fixed dropout

rate of 20%;

- **TSR-CNN-3**: addition of a dropout layer after each max pooling layer with a dropout rate increasing from 20% to 40% proportional to the layer's depth;
- **TSR-CNN-4**: placing batch normalization layers after every activation layer and full connected layer;
- **TSR-CNN-5**: batch normalization layers, as in TSR-CNN-4, and dropout layers with fixed dropout rate, as in TSR-CNN-2;
- **TSR-CNN-6**: batch normalization layers, as in TSR-CNN-4, and dropout layers with increasing dropout rate, as in TSR-CNN-3.

From these architectures, we calculate the accuracy and loss function resulting after training and testing for a recognition task over the German Traffic Sign Recognition Benchmark dataset described in Subsection 2.2.2. Results are presented, alongside considerations over the confusion matrix resulted from the best performing model in Chapter 5.

## 4.3 EARLY DATA FUSION FOR YOLO-BASED FRAMEWORK

The YOLO algorithm has a limitation of a maximum of three channels in its image inputs. Thus, in order to allow data fusion to be used in YOLO detection, it is necessary to establish a framework to make the inputs compatible. Specifically, we propose a way of performing early sensor fusion using RGB color channels from digital cameras and a 3D point cloud from LiDAR sensors. The proposed framework is illustrated in Figure 4.4.

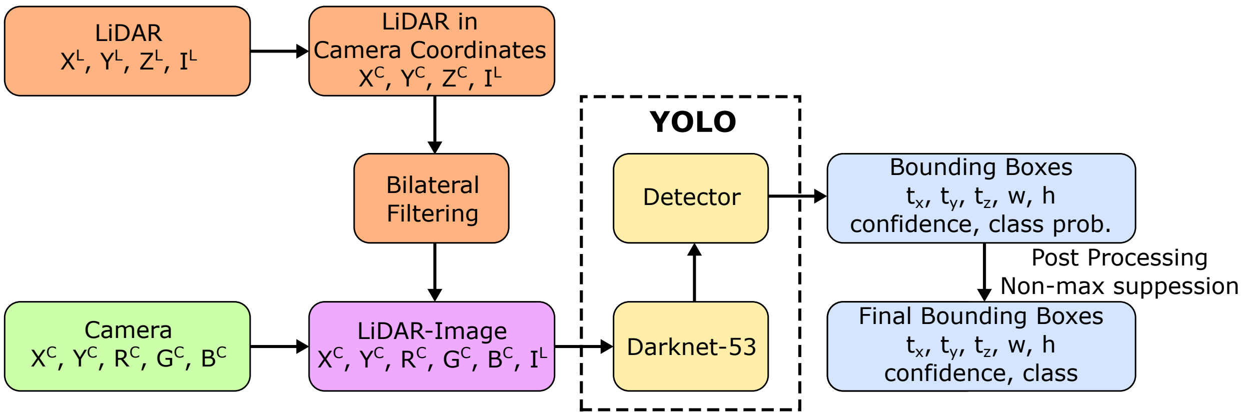


Figure 4.4: Block diagram of the proposed data-level sensor fusion YOLO based framework for raw data (YOLO-RF).

We now follow the diagram shown in 4.4 to explain our proposal. From the camera input, the original data points are the $X^L$ and $Y^L$ spatial coordinates and the three color channels $R^C$, $G^C$ and $B^C$. From the LiDAR, the 3D point cloud contributes with three spatial coordinates $X^L$, $Y^L$

and $Z^L$ along with an intensity measurement $I^L$. In order to match the inputs from the latter to the former sensor, the spatial coordinates in the LiDAR are projected into camera coordinates $X^C$, $Y^C$ and $Z^C$ according to the image format. Next, a depth and a reflectance map are generated, containing the distance and intensity data respectively. These generated maps are upscaled using bilateral filtering and finally fused together with the image data from the camera. That way, we achieve a YOLO-compatible set of parameters, noted as "LiDAR-Image".

Now onto the detection algorithm, our sensor fusion result is fed into the YOLO algorithm. By performing object detection over the samples, YOLO produces confidence scores, the bounding boxes' dimensions $t_x$, , $t_y$, $w$ and $h$, the respective anchor boxes.

After, the YOLO results are post-processed to turn the produced dimensions into absolute coordinates $x$, $y$, $w$ and $h$. A non-max suppression is also applied to maintain only the set of boxes with the highest confidence score. With both these information, we project onto the image the final bounding boxes.

### 4.3.1 LiDAR data transformation

In this subsection, we discuss how data transformation is applied to LiDAR measurements to make it compatible with digital camera images. As explained in Subsection 3.1.2, LiDAR provides reflectance and depth data, forming a total of three spatial coordinates. As cameras are 2D vectors, it is necessary to project the 3D space from LiDAR to make both input sizes match for sensor fusion.

We can represent the 3D space from LiDAR as a row vector $L_\mathrm{k} \in R^{(1\times 4)} (k \in 1, \cdots, n)$ like so:

$$L_\mathrm{k} = \begin{bmatrix} X_k^L & Y_k^L & Z_k^L & I_k^L \end{bmatrix}^\mathrm{T}, \tag{4.7}$$

where $X_k^L$, $Y_k^L$, and $Z_k^L$ are the LiDAR spatial coordinates and $I_k^L$ is the measured intensity data. From Equation 4.7, it is necessary to project the data into a 2D image plane suitable to the camera's coordinate system. We express the LiDAR coordinates transformation as:

$$\begin{bmatrix} X_k^C \\ Y_k^C \\ Z_k^C \\ 1 \end{bmatrix} = \begin{bmatrix} R_L^C & T_L^C \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_k^L \\ Y_k^L \\ Z_k^L \\ 1 \end{bmatrix}, \tag{4.8}$$

where $X_k^C$, $Y_k^C$, and $Z_k^C$ are the resulting projected coordinates of LiDAR, referenced to the camera's coordinate system. $R_L^C \in R^{(3\times 3)}$ is a rotational matrix and $T_L^C \in R^{(3\times 3)}$ is a translational matrix used for the LiDAR-to-camera projection. The KITTI dataset provides such matrices along with the camera and LiDAR data.

Next, it is necessary to make compatible the dimensions of the digital image and the projected point cloud. This is achieved by dropping the areas of the projection that do not comprise the same objects as the image, illustrated in Figure 4.5.

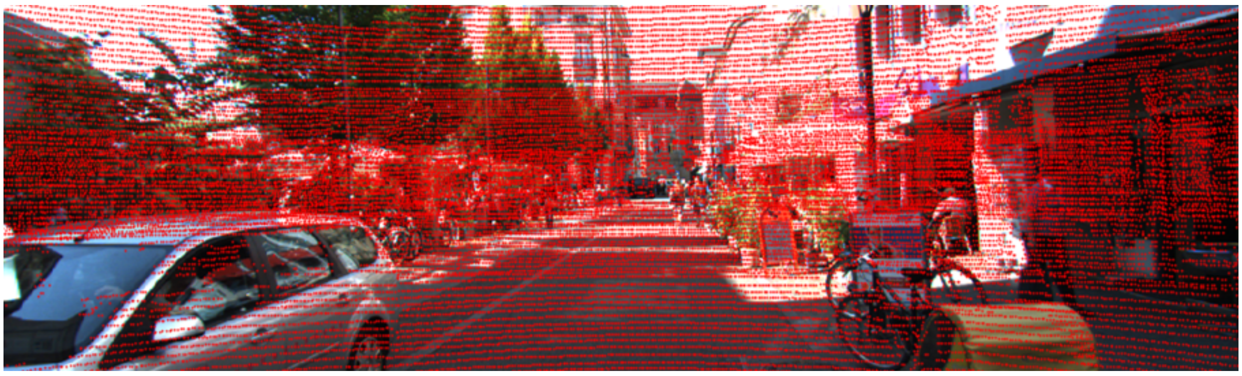


Figure 4.5: Projected sparse point cloud from LiDAR over a camera image's coordinates.

Next, we convert the projected points into a 2D array. Two different array maps are created to better exploit measurements provided by the sensors. The first is formed by the 2D spatial coordinates paired with the LiDAR intensity values, resulting in a reflectance map, illustrated on the top of Figure 4.6. The next is the depth information mapping in relation to the 2D coordinates, as shown on the bottom of Figure 4.6. Both arrays are then upscaled using a bilateral filter, turning the sparse LiDAR data into a denser plot.

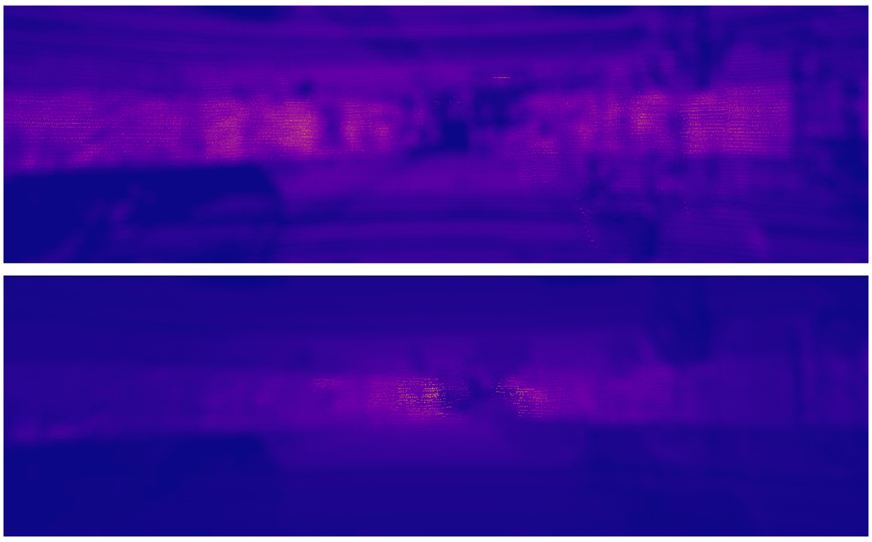


Figure 4.6: Resulting reflectance map from data-level sensor fusion, on the top, and its corresponding depth map, on the bottom.

### 4.3.2 Sensor fusion of LiDAR and camera

Now, with the reflectance and depth maps in a matching format, it is possible to fuse the processed LiDAR data with the original color image. In our proposed framework, LiDAR contributes with two additional channels — reflectance and depth — on top of the three RGB color channels from the camera. This resulting LiDAR-image is represented in Figure 4.7.

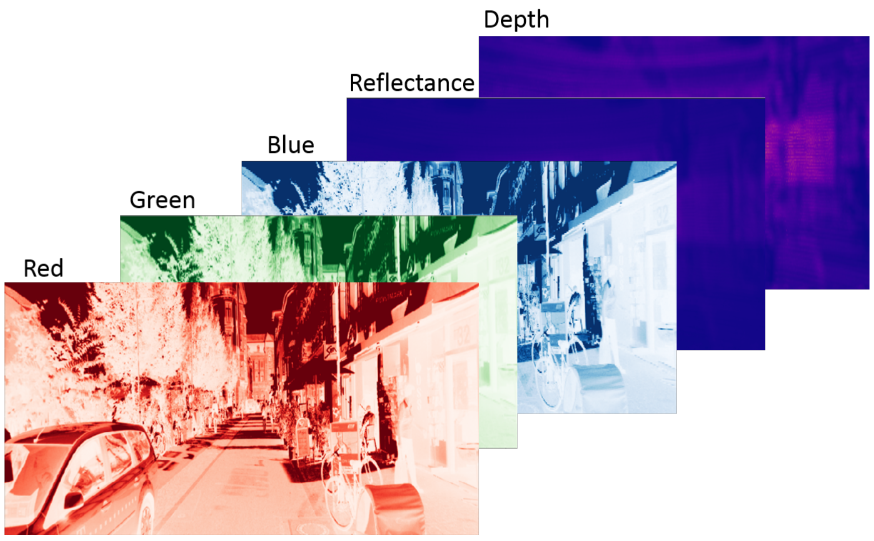


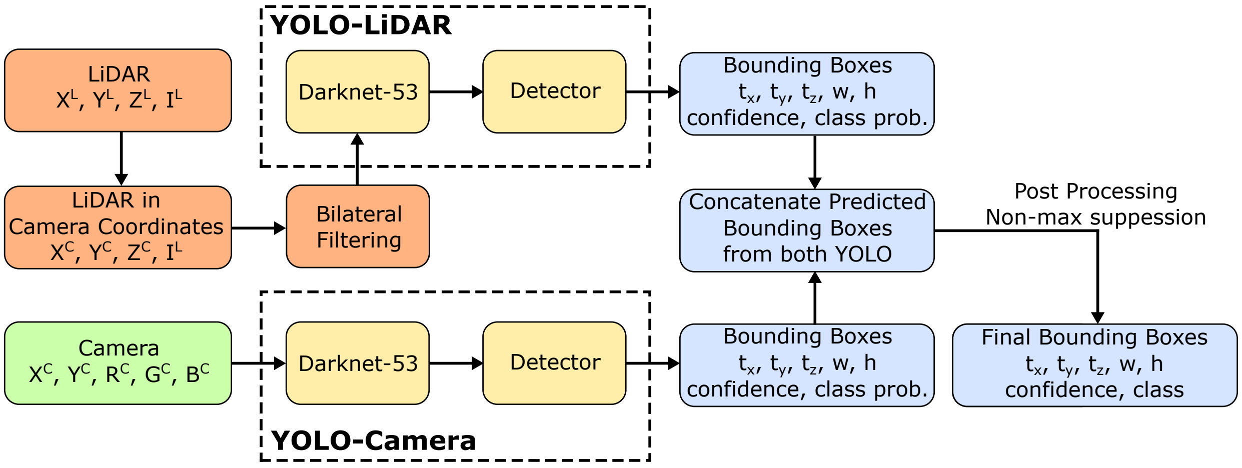
Figure 4.7: Illustration of fused sensor data channels, from front to back: red, green, blue, reflectance and depth.

### 4.3.3 Model Training

To evaluate the performance of our method, three other models of YOLO are also trained. The differences between each version are summarized in Table 4.1, according to the nomenclature used previously in this work. One of these approaches also applies sensor fusion, but at feature-level. This approach, called YOLO-DF is illustrated in Figure 4.8 and serves to validate if our proposal is more advantageous in relation to this other sensor fusion scheme.

Table 4.1: Implementations of YOLO-based models regarding input and sensor fusion type

| | **Input type** | **Type of fusion** | **Input size** |
|---|---|---|---|
| **YOLO-Camera** | Images | None | 416×416×3 |
| **YOLO-LiDAR** | Point cloud | None | 416×416×2 |
| **YOLO-DF** | Images + Point cloud | Feature-level | 416×416×5 |
| **YOLO-RF** | Images + Point cloud | Data-level | 416×416×5 |



YOLO-LiDAR
LiDAR $X^L, Y^L, Z^L, I^L$
Darknet-53
Detector
Bounding Boxes $t_x, t_y, t_z$, w, h confidence, class prob.
LiDAR in Camera Coordinates $X^C, Y^C, Z^C, I^L$
Bilateral Filtering
Concatenate Predicted Bounding Boxes from both YOLO
Post Processing Non-max suppression
Camera $X^C, Y^C, R^C, G^C, B^C$
Darknet-53
Detector
Bounding Boxes $t_x, t_y, t_z$, w, h confidence, class prob.
Final Bounding Boxes $t_x, t_y, t_z$, w, h confidence, class
YOLO-Camera

Figure 4.8: Block diagram of the feature-level sensor fusion YOLO model (YOLO-DF).

All three models are trained for 40 epochs. The KITTI dataset scene used is composed of 7481 training images, along with test and validation samples. The annotations of each data sample are converted into three tensors, one for each scale, to be recognized by the algorithm. This is achieved for each box by determining the best anchor for each ground truth based on the calculated IoU values. We reserve a portion of 15% of the test images set — that contain no annotations — for use as validation samples.

### 4.3.4 Evaluation methodology

The performance metric used for evaluating the trained models is the Mean Average Precision (mAP), as considered in the Pascal VOC 2012 competition [69]. The mAP value is calculated by taking the mean value of the average precision for each of the prediction classes. Based on the predicted classes' IoU value in comparison to the ground truth, a prediction is considered a match if the object label is equivalent and IoU $> 0.5$. To avoid double-counting, ground-truth objects used in a match are discarded afterwards. Boxes with IoU smaller than 0.5 are regarded as false positives. From these assessments, we are able to generate the precision-recall curve. Then, the average precision is calculated by taking the area under this resulting curve.

# 5 RESULTS AND DISCUSSION

This chapter presents results for the validation of all three proposals. First, our pedestrian detection algorithm is validated on both the PSU and INRIA datasets, showing substantial improvements over previous benchmarks. We evaluate the results comparing performance of the SVM model and the XGBoost, showing which is more adequate for the task at hand.

Next, we show the resulting accuracy and loss results of the TSR-CNN-6 algorithm proposed for traffic sign recognition on the GTSRB dataset. We compare the iterations made, demonstrating which optimization techniques generated higher performance gains. By comparing the resulting metrics to the base model, we analyse improvements achieved and discuss possible error reduction approaches.

Concluding the chapter, we gauge the performance of the proposed early sensor fusion technique for YOLO-based frameworks. By comparing our YOLO-RF to YOLO-Camera, YOLO-LiDAR and YOLO-DF approaches, we validate the precision advancements reached in this work. We also compare error type occurrences of each approach to give a wider perspective on each one's limitations and possibilities of further improvement.

## 5.1 PEDESTRIAN DETECTION USING HOG SVM AND XGBOOST

We now present the results for testing both pedestrian detection classification techniques over the PSU dataset and cross validation over the INRIA dataset. The metrics obtained are compared to the state-of-the-art benchmarks presented in [32]. The achieved performance metrics are summarized in Tables 5.1 for the PSU dataset and 5.2 for the INRIA dataset.

Table 5.1: Metrics achieved by the proposed frameworks applied to the PSU dataset

| | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| SVM with linear kernel | 82% | 64% | 74% | 78% |
| XGBoost | 76% | 53% | 68% | 69% |

Table 5.2: Metrics achieved by the proposed frameworks applied to the INRIA dataset

| | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| SVM with linear kernel | 86% | 92% | 87% | 89% |
| XGBoost | 82% | 94% | 80% | 86% |

For the proposed SVM model using a linear kernel we reach an accuracy of 82% on the test images of the PSU dataset. This model produced error rates of 36% false positives and 0% false negatives. As for the XGBoost model, the resulting figures are an accuracy of 76.5%, with error rates of 47% false positive and 0% false-negative results. Theses statistics are presented in Figure 5.1 along with the benchmarks. The confusion matrices presenting the error rates are in Figure 5.2.

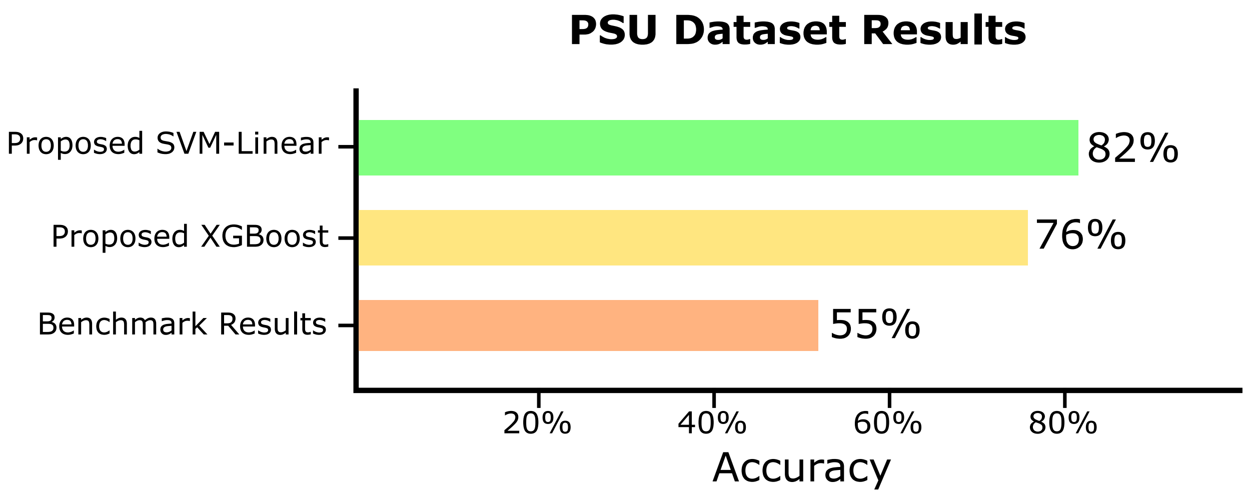


Figure 5.1: Resulting accuracy of the proposed frameworks in comparison to the benchmark of the PSU dataset.

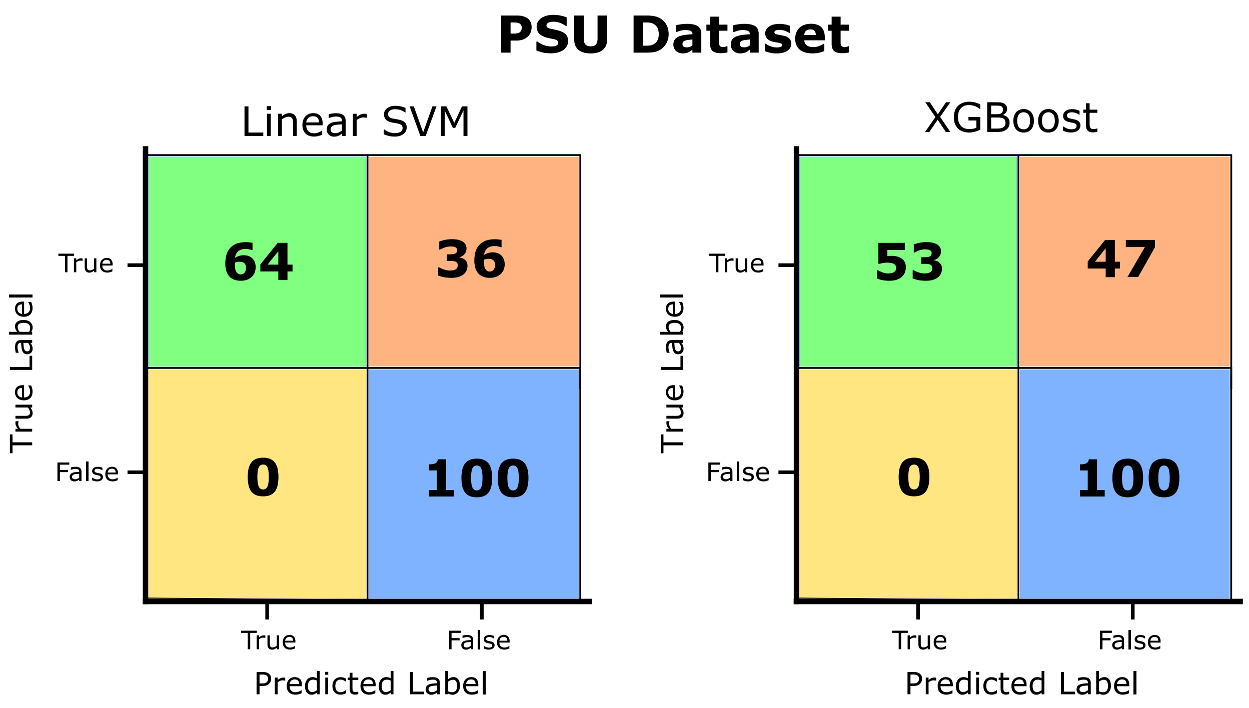


Figure 5.2: Confusion matrices from the experiment with the PSU dataset.

Additionally, we cross validate the implemented classification models over the INRIA dataset images. The linear SVM resulted in 86.3% accuracy rate, with error rates of 7.5% false positives and 23.2% false negatives. In turn, the XGBoost approach reached 81.6% accuracy, with error rates of 5.9% false positives and 37.8 % false negatives results. The benchmark comparison for both models applied to the INRIA dataset are shown in Figure 5.3, while the confusion matrices are in Figure 5.4.

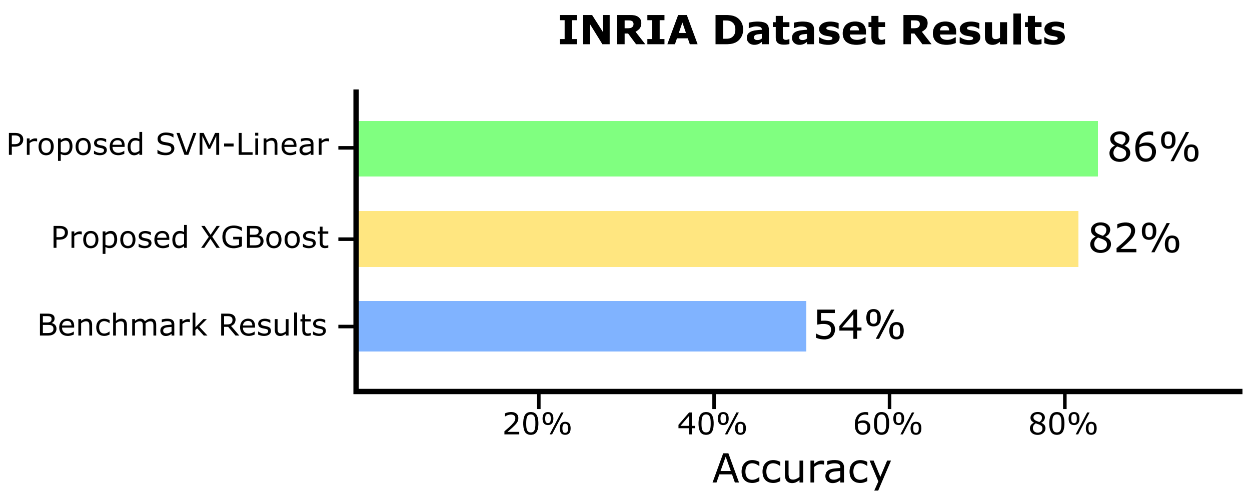


Figure 5.3: Resulting accuracy of the proposed frameworks in comparison to the benchmark of the INRIA dataset.

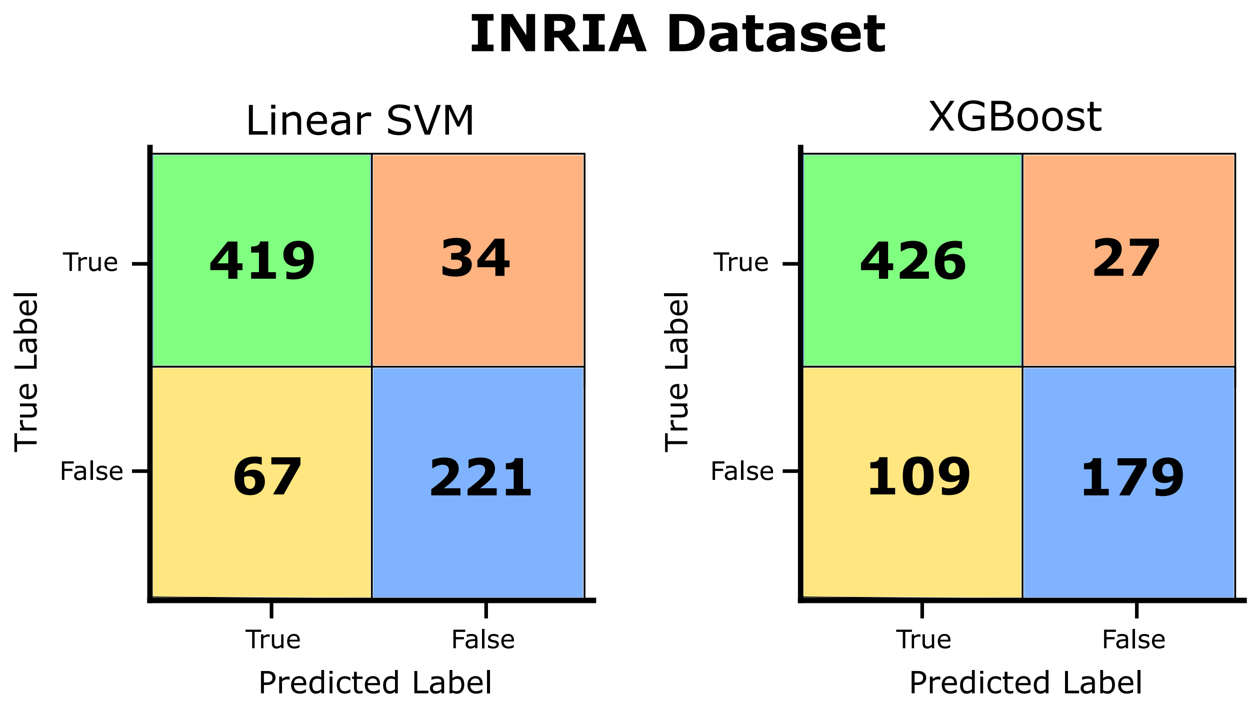


Figure 5.4: Confusion matrices from the experiment with the INRIA dataset.

The PSU dataset includes scenarios with partial occlusion and visual obstruction of pedestrians in the samples. That way, we deem it a more challenging scenario, which is corroborated by the lower accuracy results. However, we surpass the benchmarks of [32] by 28% on this dataset using Linear Kernel SVM. The XGBoost algorithm reaches a 22% improvement.

For the INRIA dataset, we observe higher accuracy as pedestrians are clearer in the scenes. The reached accuracy rates of the Kernel SVM model is of 86%, adding a 31% enhancement over the benchmark established. The XGBoost classifier also improved over the benchmark for a figure of 27%.

Regarding the error rates, it is important not just to consider the amount of errors, but their types as well. Because decisions based on different detection results can lead to erratic maneuvers in AVs, non-detection of a pedestrian when one is present can lead to unforeseen accidents. This situation corresponds to false negative results. As seen in Figures 5.2 and 5.4, we reach higher false negatives over the PSU dataset for both approaches, with XGBoost showing worse performance. Over the INRIA dataset, the false positive occurrences are more common. However, this error type may lead to unnecessary maneuvers on the AV, most likely not harming anyone and causing some discomfort.

The analysis of the presented metrics validate our improvements over existing benchmarks. The Kernel SVM algorithm is shown to perform better on the pedestrian detection task than the XGBoost approach. We assume this is because since human subjects are better classified in the hyperplane domain than in decision trees. We also provide valuable intuition regarding the datasets and the scenarios they represent. For instance, the occluded and obstructed pedestrian scenarios on the PSU dataset show a greater number of false negatives, which must be diminished to reach acceptable performance. The INRIA dataset has lead to higher false positives, maybe due to the more diversity in background scenarios of the samples. In both cases, adding redundancy in decision or performing sensor fusion may deem advantageous to improve pedestrian detection.

## 5.2 OPTIMIZATION OF CNN-BASED ALGORITHM FOR TRAFFIC SIGN RECOGNITION

In this section, we present the results of traffic sign detection tasks over the GTSRB dataset using the different implementations of the TSR-CNN algorithm. All the TSR-CNN networks are trained using Keras. We also use OpenCV 4.1.2 and TensorFlow 2.0 to carry out all the computational vision operations on Google Colab. The results are presented in Figure 5.5 for the base model and all remaining iterations.

The results from the improvements applied in comparison to the baseline TSR-CNN model show which techniques rendered the best results in the present task. The accuracy values reached are presented in Figure 5.5. Starting at the base reference of 98.03% accuracy on the TSR-CNN model, we reach an improvement of 1.73%, totaling an accuracy of 99.76% on the best-performing TSR-CNN-6. In terms of loss curves, we reach a 0.27% loss rate at TSR-CNN base model and improve it to 0.01% at TSR-CNN-6.

From the results, we can make some conclusions on the techniques considered to improve performance. The first remark is that fixed dropout rates are deemed inappropriate in the present application. The implementation of this technique in TSR-CNN-2 is the only instance that reached lower accuracy than the base model. We suspect that the fixed fixed dropout diminishes or hinders regularization effects on deeper layers by not removing enough nodes to keep a reduced complexity. On the other hand, the increasing dropout rates resulted in improved performance in both implementations, TSR-CNN-3 and TSR-CNN-6. The TSR-CNN-5 implementation, that also use fixed dropout, shows a worse performance than TSR-CNN-6 with a variable dropout.

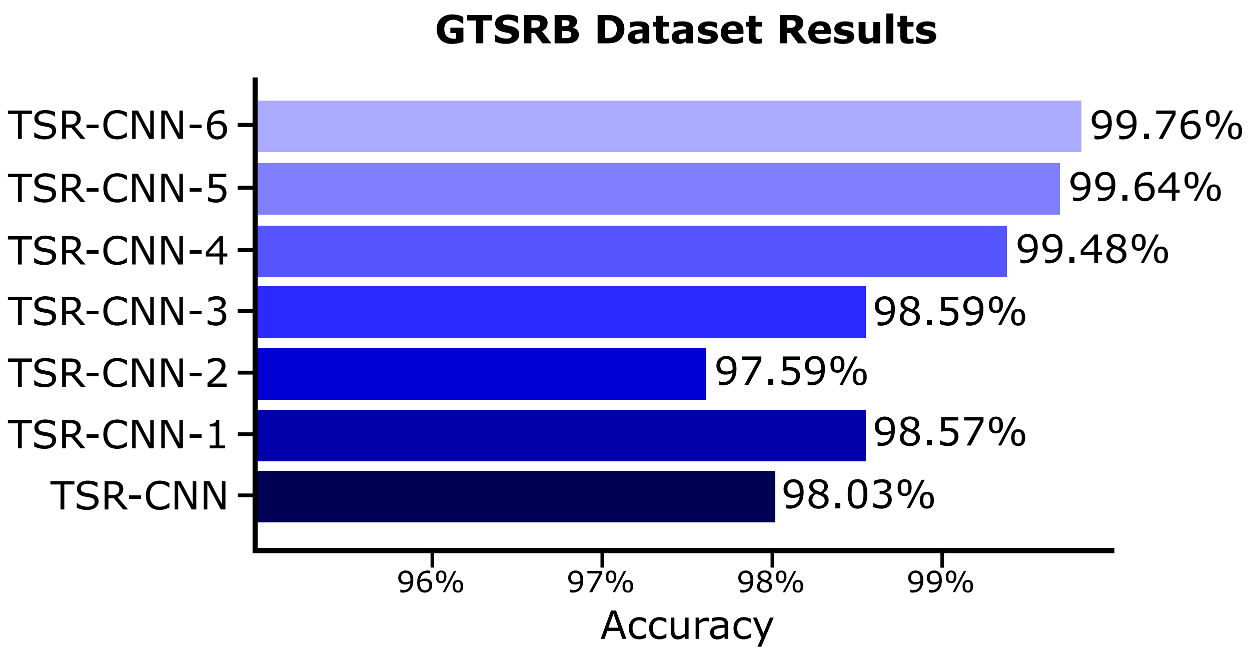


Figure 5.5: Resulting detection accuracy of CNN architectures under test with the GTSRB dataset.

The next conclusion is that using regularizers generate a performance improvement comparable to that of variable dropout rates, as seen comparing TSR-CNN-1 to TSR-CNN-3. We assume the effect introduced by regularizers is equivalent to that of dropout. However, the highest performance gains are experienced by batch-normalized implementations. As observed, TSR-CNN-4 shows better accuracy than TSR-CNN-1 through TSR-CNN-3, which implement regularization and dropout, respectively. Even TSR-CNN-5 shows enhanced performance using a fixed dropout rate, previously determined inappropriate. The most accurate version is TSR-CNN-6, achieving a total 99.76% accuracy and 0.01% loss. This instance applies batch normalization and increasing dropout rates, surpassing benchmarks presented in the literature review whilst using fewer resources.

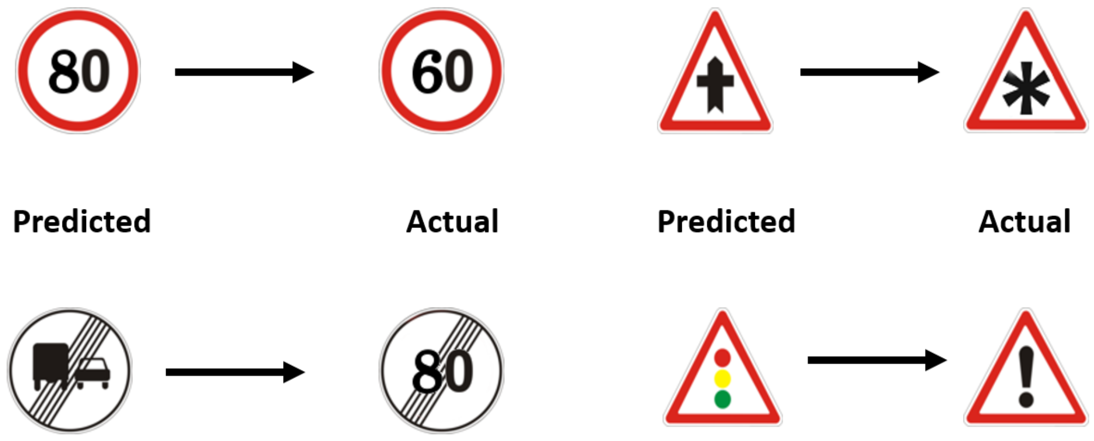


Figure 5.6: Examples of wrong traffic sign predictions and the true corresponding signs.

To give a better picture of possible improvements on the proposed models, we analyse the wrong predictions resulting in TSR-CNN-6. As exemplified in Figure 5.6, some signs were misclassified in favor of similar-looking ones. We hypothesize that some common aspects are observed in these errors, such as similar forms, visual features and indifference to color information. Then, we conclude that these errors probably arise from the image preprocessing stage performed over the samples. In this step, we convert images to gray-scale and reduce their size. Thus, we lose both color and resolution features in order to curtail the computational power required to train the CNN. By using more advanced resources, such as dedicated GPUs, we believe the proposed algorithm may reach even higher improvements.

## 5.3 DATA FUSION FOR YOLO-BASED FRAMEWORK

We presently validate the proposed sensor fusion framework over color camera and LiDAR data present in a scene from the KITTI dataset. A total of 1048 samples are used for validation, containing 4095 vehicles to be detected. The mAP of all four implemented models are presented in Figure 5.7.

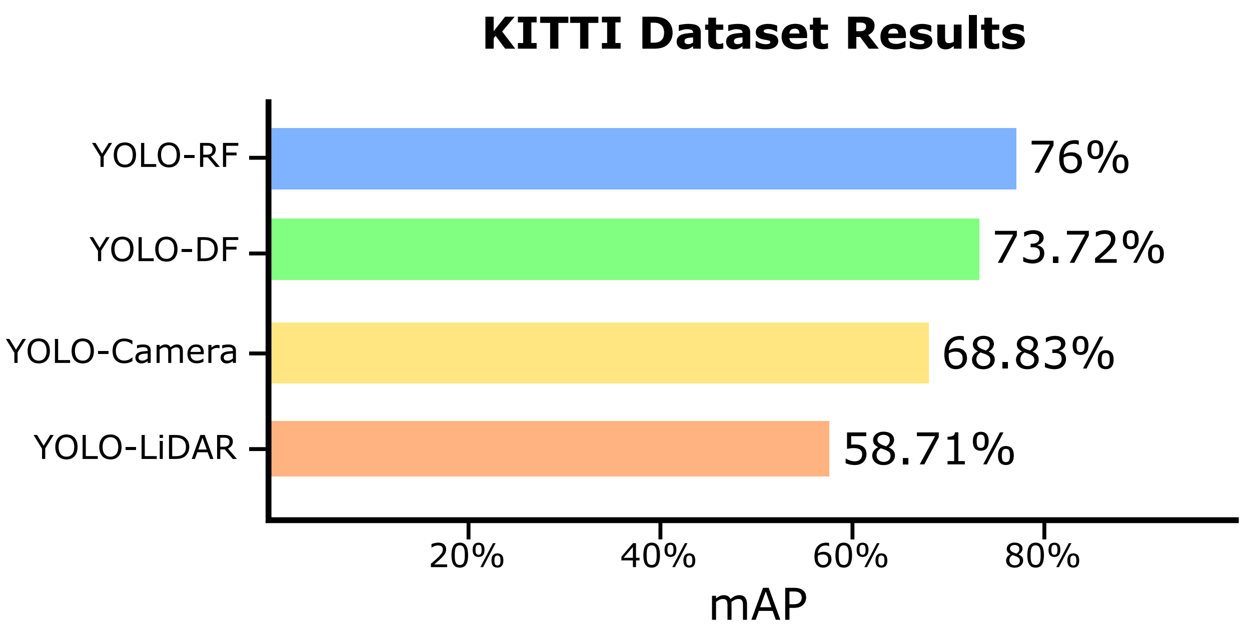


Figure 5.7: Resulting mAP values of the networks tested on the KITTI dataset.

From the results, we can see that the proposed early sensor fusion YOLO-RF has the highest mAP. Regarding the other models, we observe the lowest accuracy is observed in YOLO-LiDAR. We believe that this is due to the more sparse data of 3D point clouds in this sensor, which can lead to fewer significant features for YOLO to detect. Next, YOLO-Camera shows a lower mAP than that of either sensor fusion results. This leads to conclude that the addition of depth and reflectance information are a net positive over the detection algorithm proposed.

The sensor fusion approaches — YOLO-RF and YOLO-DF — represent data-level fusion and feature-level fusion respectively. The former approach reached a higher mAP of 76%, in comparison to 73.72% of the latter. Thus, we conclude that raw data fusion is shown to be better in performance for camera-LiDAR sensor fusion vehicle detection. Besides that, we also cogitate that data-level fusion brings additional advantages over feature-level fusion. For instance, by combining data early on, we pass the inputs in just one instance of YOLO, in comparison to two networks in YOLO-DF. Also, the method of combining the features are more complex since the context and significance of resulting bounding boxes must be considered to successfully extract augmented information from them.

The model YOLO-RF, based on the proposed framework has an mAP of 78%, a 10% improvement from Base YOLO, and a 20% improvement from YOLO-LiDAR. Next, we present the error rates of the models, summarized in Table 5.4.

Table 5.3: Comparison of true positives, false positives and false negatives of YOLO-based models

Table 5.4: Results of YOLO-based models showing the occurrences of errors

| | **True positives** | **False negatives** | **False positives** |
|---|---|---|---|
| **YOLO-Camera** | 2995 | 1100 | 754 |
| **YOLO-LiDAR** | 2531 | 1564 | 551 |
| **YOLO-DF** | 3230 | 865 | 1246 |
| **YOLO-RF** | 3323 | 772 | 691 |

From the error rates, we can observe that the sensor fusion approaches have approximate true positive results. However, the false negative and false positives, specially, are considerably higher in YOLO-DF. We can hypothesize that this behavior is due to the concatenation of bounding boxes in this model. This can result in IoU values surpassing the minimum match threshold by summing contributions of the two YOLO networks, incurring undue detection.

Regarding individual sensor approaches, YOLO-RF outperformed them in most metrics. In turn, YOLO-LiDAR shows a lower false positive rate. In any case, we believe that this is due to the low density of LiDAR data, which can lead to less detection matches overall. This is also corroborated due to the high number of false negatives in YOLO-LiDAR, showing that this method is dropping vehicle detection across the board. Based on these conclusions, we state that YOLO-RF presents the best performance overall.

For comparison reasons, we consider other models trained with the KITTI dataset, such as UNN [70] and WM-YOLO [22]. These models are trained with higher computational resources. For instance, WM-YOLO was trained for 45000 epochs in comparison to our 40 epochs. Thus, we believe that, due to the large resource disparities, it is not interesting to directly compare our results to those cited. Thus, our contribution is significant in the scale utilized, that is, requiring less processing resources and fewer epochs for training. We also note that our approach may adapt faster to models combined with reinforcement learning, as it takes less epochs for training.

# 6 CONCLUSION

During this work, we proposed a number of improvements for Autonomous Vehicles in order to increase road safety for drivers and pedestrians alike. First, we introduce an improved classification approach that increased accuracy in pedestrian detection in occluded or obstructed scenarios based on HOG and Kernel SVM. We also implement a model based on HOG and XG-Boost, comparing it to the other classifier. The Linear Kernel SVM method utilized outperformed the original benchmarked results for the datasets used in up to 31%.

Next, in order to increase traffic sign detection accuracy, we propose an approach using Convolution Neural Networks. Our method is able to reach 99.76% accuracy rate, outperforming the base model in the dataset considered. Additionally, the optimization techniques employed in each iteration are evaluated, showing which ones lead to better performance gains. We also explore forms of reducing the occurrences of detection errors, which can further improve the applicability of our method in real scenarios.

Following, we present a data-level sensor fusion framework to enable combining raw camera and LiDAR data, named YOLO-RF. Our approach is able to combine the strength of the individual YOLO-Camera algorithm — with higher recall rate — with the individual sensor YOLO-LiDAR algorithm — which has better precision. Thus, it outperforms both approaches individually, with improved results when considering occurrences of true positives, false positives, and false negatives. The proposed YOLO-RF achieves a 10% increase in mAP in comparison to to the YOLO-Camera algorithm and approximately a 20% improvement in mAP in comparison to YOLO-LiDAR. We also compare it to a delayed fusion at feature-level model named YOLO-DF, verifying that early fusion achieves a marginally improved result.

Road safety in an ongoing concern, with much space for improvement in all approaches proposed. The combination of techniques integrates and better exploits each one's strengths. At the same time, they are able to compensate for inherent imprecision among each other. They can also serve as redundancy, guaranteeing that the best decision is taken by the vehicle's automated system.

## 6.1 FUTURE WORKS

Regarding the pedestrian detection approach, it is of interest to reduce the occurrences of false negatives, thus guaranteeing that no failure in pedestrian recognition occurs in challenging scenarios. Furthermore, by employing a possible multi-camera approach, fusing data from different image views, greater detection performance is achievable in occluded and obstructed scenarios.

In the area of traffic sign recognition, a possible future work is to cross validate the achieved results with datasets from different countries. On top of that, it is important to propose a solution to the misrecognition of similar signs. We can suggest using an additional or improved image processing algorithm applying segmentation approaches to tell similar signs apart to improve this scenario.

Finally, the use of YOLO-RF may be further improved by extending the number of training epochs. Thus, by training the model with more computational resources, it is possible to achieve an a higher advantage or surpass the other models considered. Optimization of data preprocessing and fine tuning training parameters also seem fruitful. In terms of applications, the proposed YOLO-RF model has a wide potential. It can be applied to improve both other detection tasks considered in this work, since it is able to access color information for traffic sign recognition and add depth information for pedestrian detection.

# BIBLIOGRAPHY

1 U.S. DEPARTMENT OF TRANSPORTATION. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. [S.l.], fev. 2015.

2 SOCIETY OF AUTOMOTIVE ENGINEERS. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. [S.l.], 2018.

3 ERRAMI, M.; RZIZA, M. Improving pedestrian detection using support vector regression. In: IEEE. *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*. [S.l.], 2016. p. 156–160.

4 CHEN, Z.; CHEN, K.; CHEN, J. Vehicle and Pedestrian Detection Using Support Vector Machine and Histogram of Oriented Gradients Features. In: *2013 International Conference on Computer Sciences and Applications*. Wuhan, China: IEEE, 2013. p. 365–368. ISBN 978-0-7695-5125-8.

5 CHEN, Y.-Y.; JHONG, S.-Y.; LI, G.-Y.; CHEN, P.-H. Thermal-Based Pedestrian Detection Using Faster R-CNN and Region Decomposition Branch. In: *2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. Taipei, Taiwan: IEEE, 2019. p. 1–2. ISBN 978-1-72813-038-5.

6 KHARJUL, R. A.; TUNGAR, V. K.; KULKARNI, Y. P.; UPADHYAY, S. K.; SHIRSATH, R. Real-time pedestrian detection using svm and adaboost. In: IEEE. *2015 International Conference on Energy Systems and Applications*. [S.l.], 2015. p. 740–743.

7 LAN, W.; DANG, J.; WANG, Y.; WANG, S. Pedestrian Detection Based on YOLO Network Model. In: *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*. Changchun: IEEE, 2018. p. 1547–1551. ISBN 978-1-5386-6074-4 978-1-5386-6075-1.

8 ZHANG, J.; XIAO, J.; ZHOU, C.; PENG, C. A multi-class pedestrian detection network for distorted pedestrians. In: *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. Wuhan: IEEE, 2018. p. 1079–1083. ISBN 978-1-5386-3758-6.

9 THU, M.; SUVONVORN, N. Pyramidal Part-Based Model for Partial Occlusion Handling in Pedestrian Classification. *Advances in Multimedia*, v. 2020, p. 1–15, fev. 2020. ISSN 1687-5680, 1687-5699.

10 NAN, M.; LI, C.; JIANCHENG, H.; QIUNA, S.; JIAHONG, L.; GUOPING, Z. Pedestrian Detection Based on HOG Features and SVM Realizes Vehicle-Human-Environment Interaction. In: *2019 15th International Conference on Computational Intelligence and Security (CIS)*. Macao, Macao: IEEE, 2019. p. 287–291. ISBN 978-1-72816-092-4.

11 WALI, S. B.; ABDULLAH, M. A.; HANNAN, M. A.; HUSSAIN, A.; SAMAD, S. A.; KER, P. J.; MANSOR, M. B. Vision-Based Traffic Sign Detection and Recognition Systems: Current Trends and Challenges. *Sensors*, v. 19, n. 9, p. 2093, maio 2019. ISSN 1424-8220.

12 YASMINA, D.; KARIMA, R.; OUAHIBA, A. Traffic signs recognition with deep learning. In: *2018 International Conference on Applied Smart Systems (ICASS)*. Medea, Algeria: IEEE, 2018. p. 1–5. ISBN 978-1-5386-6866-5.

13 SUN, Y.; GE, P.; LIU, D. Traffic Sign Detection and Recognition Based on Convolutional Neural Network. In: *2019 Chinese Automation Congress (CAC)*. Hangzhou, China: IEEE, 2019. p. 2851–2854. ISBN 978-1-72814-094-0.

14 HALOI, M. Traffic Sign Classification Using Deep Inception Based Convolutional Networks. *arXiv:1511.02992 [cs]*, jul. 2016. Disponível em: <http://arxiv.org/abs/1511.02992>.

15 TABERNIK, D.; SKOCAJ, D. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition. *IEEE Transactions on Intelligent Transportation Systems*, v. 21, n. 4, p. 1427–1440, abr. 2020. ISSN 1524-9050, 1558-0016.

16 SHEIKH, M. A. A.; KOLE, A.; MAITY, T. Traffic sign detection and classification using colour feature and neural network. In: *2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI)*. Kolkata, India: IEEE, 2016. p. 307–311. ISBN 978-1-5090-2638-8.

17 DO, H. N.; VO, M.; LUONG, H. Q.; NGUYEN, A. H.; TRANG, K.; VU, L. T. K. Speed limit traffic sign detection and recognition based on support vector machines. In: *2017 International Conference on Advanced Technologies for Communications (ATC)*. [S.l.: s.n.], 2017. p. 274–278.

18 YANG, Y.; LUO, H.; XU, H.; WU, F. Towards Real-Time Traffic Sign Detection and Classification. *IEEE Transactions on Intelligent Transportation Systems*, v. 17, n. 7, p. 2022–2031, jul. 2016. ISSN 1524-9050, 1558-0016.

19 XU, D.; ANGUELOV, D.; JAIN, A. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA: IEEE, 2018. p. 244–253. ISBN 978-1-5386-6420-9.

20 GU, S.; ZHANG, Y.; TANG, J.; YANG, J.; KONG, H. Road Detection through CRF based LiDAR-Camera Fusion. In: *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, 2019. p. 3832–3838. ISBN 978-1-5386-6027-0.

21 HUANG, S.; XIONG, G.; ZHU, B.; GONG, J.; CHEN, H. LiDAR-Camera Fusion Based High-Resolution Network for Efficient Road Segmentation. In: *2020 3rd International Conference on Unmanned Systems (ICUS)*. Harbin: IEEE, 2020. p. 830–835. ISBN 978-1-72818-025-0.

22 KIM, J.; KIM, J.; CHO, J. An advanced object classification strategy using YOLO through camera and LiDAR sensor fusion. In: *2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS)*. Gold Coast, Australia: IEEE, 2019. p. 1–5. ISBN 978-1-72812-194-9.

23 LYU, K.; HU, J.; ZHAO, C.; HOU, X.; XU, Z. Multi-sensor fusion based obstacle localization technology. In: IEEE. *2020 IEEE 16th International Conference on Control & Automation (ICCA)*. [S.l.], 2020. p. 731–736.

24 LEE, J.-S.; PARK, T.-H. Fast Lidar - Camera Fusion for Road Detection by CNN and Spherical Coordinate Transformation. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France: IEEE, 2019. p. 1797–1802. ISBN 978-1-72810-560-4.

25 DENG, Q.; LI, X.; NI, P.; LI, H.; ZHENG, Z. Enet-CRF-Lidar: Lidar and Camera Fusion for Multi-Scale Object Recognition. *IEEE Access*, v. 7, p. 174335–174344, 2019. ISSN 2169-3536.

26 WU, T.-E.; TSAI, C.-C.; GUO, J.-I. LiDAR/camera sensor fusion technology for pedestrian detection. In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Kuala Lumpur: IEEE, 2017. p. 1675–1678. ISBN 978-1-5386-1542-3.

27 DALAL, N. *INRIA Person Dataset*. 2005. Disponível em: <http://pascal.inrialpes.fr/data/human/>.

28 DOLLÁR, P.; WOJEK, C.; SCHIELE, B.; PERONA, P. Pedestrian detection: An evaluation of the state of the art. *PAMI*, v. 34, 2012.

29 ENZWEILER, M.; EIGENSTETTER, A.; SCHIELE, B.; GAVRILA, D. M. Multi-cue pedestrian classification with partial occlusion handling. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2010. p. 990–997. ISSN 1063-6919.

30 VEHICLE, E. *CVC-14: Visible-Fir Day-Night Pedestrian Sequence Dataset*. 2016. Disponível em: <http://adas.cvc.uab.es/elektra/enigma-portfolio/cvc-14-visible-fir-day-night-pedestrian-sequence-dataset/>.

31 DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.: s.n.], 2005. v. 1, p. 886–893 vol. 1. ISSN 1063-6919.

32 THU, M.; SUVONVORN, N.; KARNJANADECHA, M. A new dataset benchmark for pedestrian detection. In: ACM. *Proceedings of the 3rd International Conference on Biomedical Signal and Image Processing*. [S.l.], 2018. p. 17–22.

33 STALLKAMP, J.; SCHLIPSING, M.; SALMEN, J.; IGEL, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In: *The 2011 International Joint Conference on Neural Networks*. [S.l.: s.n.], 2011. p. 1453–1460.

34 GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI: IEEE, 2012. p. 3354–3361. ISBN 978-1-4673-1228-8 978-1-4673-1226-4 978-1-4673-1227-1.

35 VOZAR, S. Sensors for autonomous vehicles. IEEE Learning Network, n. EDP538, 2019.

36 GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 3rd ed. ed. Upper Saddle River, N.J: Prentice Hall, 2008. ISBN 978-0-13-168728-8.

37 SALEH, B. E. A.; TEICH, M. C. *Fundamentals of Photonics*. 2nd ed. ed. Hoboken, N.J: Wiley Interscience, 2007. (Wiley Series in Pure and Applied Optics). ISBN 978-0-471-35832-9.

38 TEXAS INSTRUMENTS. *An Introduction to Automotive LIDAR*. [S.l.], dez. 2017.

39 JARVIS, A. Guide to LiDAR wavelengths. out. 2018. Disponível em: <https://velodynelidar.com/newsroom/guide-to-lidar-wavelengths/>.

40 LOHANI, B.; CHACKO, S.; GHOSH, S.; SASIDHARAN, S. Surveillance system based on flash LiDAR. *Indian Cartographer*, dez. 2013.

41 GOODIN, C.; CARRUTH, D.; DOUDE, M.; HUDSON, C. R. Predicting the influence of rain on LIDAR in ADAS. In: . [S.l.: s.n.], 2019.

42 BRISKEN, S.; RUF, F.; HÖHNE, F. Recent evolution of automotive imaging radar and its information content. *IET Radar, Sonar Navigation*, v. 12, n. 10, p. 1078–1081, 2018.

43 RAMASUBRAMANIAN, K.; RAMAIAH, K. Moving from Legacy 24 GHz to State-of-the-Art 77-GHz Radar. v. 13, n. 3, p. 46–49. ISSN 2192-9092.

44 INTERNATIONAL TELECOMMUNICATION UNION. Systems characteristics of automotive radars operating in the frequency band 76-81 GHz for intelligent transport systems applications. p. 12.

45 GALAR, D.; KUMAR, U. Chapter 1 - sensors and data acquisition. In: GALAR, D.; KUMAR, U. (Ed.). *eMaintenance*. [S.l.]: Academic Press, 2017. p. 1–72. ISBN 978-0-12-811153-6.

46 MITCHELL, H. B. *Multi-Sensor Data Fusion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. ISBN 978-3-540-71463-7 978-3-540-71559-7.

47 GALAR, D.; KUMAR, U. Sensors and Data Acquisition. In: *eMaintenance*. [S.l.]: Elsevier, 2017. p. 1–72. ISBN 978-0-12-811153-6.

48 TZAFESTAS, S. G. Mobile Robot Localization and Mapping. In: *Introduction to Mobile Robot Control*. [S.l.]: Elsevier, 2014. p. 479–531. ISBN 978-0-12-417049-0.

49 CHEN, X.; MA, H.; WAN, J.; LI, B.; XIA, T. Multi-view 3D Object Detection Network for Autonomous Driving. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, 2017. p. 6526–6534. ISBN 978-1-5386-0457-1.

50 TOMASI, C. *Histograms of Oriented Gradients*. [S.l.]: Duke University, 2015.

51 HEARST, M.; DUMAIS, S.; OSUNA, E.; PLATT, J.; SCHOLKOPF, B. Support vector machines. *IEEE Intelligent Systems and their Applications*, v. 13, n. 4, p. 18–28, jul. 1998. ISSN 1094-7167.

52 CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 785–794, ago. 2016.

53 SHAW, R. *XGBoost: A Concise Technical Overview*. 2017. Disponível em: <https://www.kdnuggets.com/2017/10/xgboost-concise-technical-overview.html>.

54 SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, p. 85–117, jan. 2015. ISSN 08936080.

55 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

56 GÉRON, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. [S.l.: s.n.], 2019. ISBN 978-1-4920-3264-9.

57 LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. Paris, France: IEEE, 2010. p. 253–256. ISBN 978-1-4244-5308-5.

58 IDE, H.; KURITA, T. Improvement of learning for CNN with ReLU activation by sparse regularization. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. Anchorage, AK, USA: IEEE, 2017. p. 2684–2691. ISBN 978-1-5090-6182-2.

59 ZHU, Q.; ZHANG, P.; WANG, Z.; YE, X. A New Loss Function for CNN Classifier Based on Predefined Evenly-Distributed Class Centroids. *IEEE Access*, v. 8, p. 10888–10895, 2020. ISSN 2169-3536.

60 GAD, A. *Is Learning Rate Useful in Artificial Neural Networks?* Disponível em: <https://www.kdnuggets.com/is-learning-rate-useful-in-artificial-neural-networks.html/>.

61 REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, 2016. p. 779–788. ISBN 978-1-4673-8851-1.

62 CASTELINO, R. M.; PINHEIRO, G. P. M.; PRACIANO, B. J. G.; SANTOS, G. A.; WEICHENBERGER, L.; JUNIOR, R. T. D. S. Improving the Accuracy of Pedestrian Detection in Partially Occluded or Obstructed Scenarios. In: *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*. Deggendorf, Germany: IEEE, 2020. p. 834–838. ISBN 978-1-72816-759-6 978-1-72816-760-2.

63 DANAPAL, G.; SANTOS, G. A.; da Costa, J. P. C. L.; PRACIANO, B. J. G.; PINHEIRO, G. P. M. Sensor fusion of camera and LiDAR raw data for vehicle detection. In: *2020 Workshop on Communication Networks and Power Systems (WCNPS)*. Brasilia, Brazil: IEEE, 2020. p. 1–6. ISBN 978-1-72818-791-4.

64 PRACIANO, B. J. G.; da Costa, J. P. C. L.; MARANHÃO, J. P. A.; de Mendonça, F. L. L.; de Sousa Júnior, R. T.; PRETTZ, J. B. Spatio-temporal trend analysis of the brazilian elections based on twitter data. In: IEEE. *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. [S.l.], 2018. p. 1355–1360.

65 SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, abr. 2015.

66 NAGPAL, A. *L1 and L2 Regularization Methods*. Disponível em: <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>.

67 BROWNLEE, J. *How to Accelerate Learning of Deep Neural Networks With Batch Normalization*. Disponível em: <https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/>.

68 IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, mar. 2015. Disponível em: <http://arxiv.org/abs/1502.03167>.

69 EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, jun. 2010. ISSN 0920-5691, 1573-1405.

70 NAGHAVI, S. H.; AVAZNIA, C.; TALEBI, H. Integrated real-time object detection for self-driving vehicles. In: *2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP)*. Isfahan, Iran: IEEE, 2017. p. 154–158. ISBN 978-1-5386-4405-8.

71 OSSMANN, M. *HackRF One - Great Scott Gadgets*. Disponível em: <https://greatscottgadgets.com/hackrf/one/>.

72 OSSMANN, M. *HackRF Wiki*. Disponível em: <https://github.com/mossmann/hackrf>.

73 RFFC5071/5072 WIDEBAND SYNTHESIZER/VCO WITH INTEGRATED 6GHz MIXER. [S.l.]: RF Micro Devices, Inc., 2014.

74 MAX2837 2.3GHz to 2.7GHz Wireless Broadband RF Transceiver. [S.l.]: Maxim Integrated Products, Inc., 2015.

75 MAX5864 - Ultra-Low-Power, High-DynamicPerformance, 22Msps Analog Front End. [S.l.]: Maxim Integrated Products, Inc., 2003.

76 COOLRUNNER-II CPLD Family. [S.l.]: Xilinx, Inc., 2008.

77 GNU Radio. *GNU Radio - The Free and Open Source Radio Ecosystem*. Disponível em: <https://www.gnuradio.org/>.

# APPENDICES

# I. SDR EXPERIMENTS

In this appendix, we present the results from test and simulation scenarios for Software Define Radio (SDR) experiments. We perform a characterization test run in a parking lot using a HackRF One device and apprehend useful data for achieving desired range requirements using an external amplifier. We also check the need for equalization in real-life scenarios using uncoded Binary Phase-Shift Keying (BPSK).

## I.1 HACKRF ONE

The HackRF One is an SDR peripheral developed by Great Scott Gadgets [71], capable of RF transmission or reception in frequencies from 1 MHz to 6 GHz. The device — shown in Figure I.1 — is an open source hardware platform aimed towards development and testing of radio applications. It comprises a half-duplex transceiver, hence it is only capable of transmitting (TX) or receiving (RX) RF signals during operation. Much of the hardware architecture is share between TX and RX chain. We then explore the hardware architecture of the HackRF One in the next paragraphs based on the information provided by the manufacturer in [72].

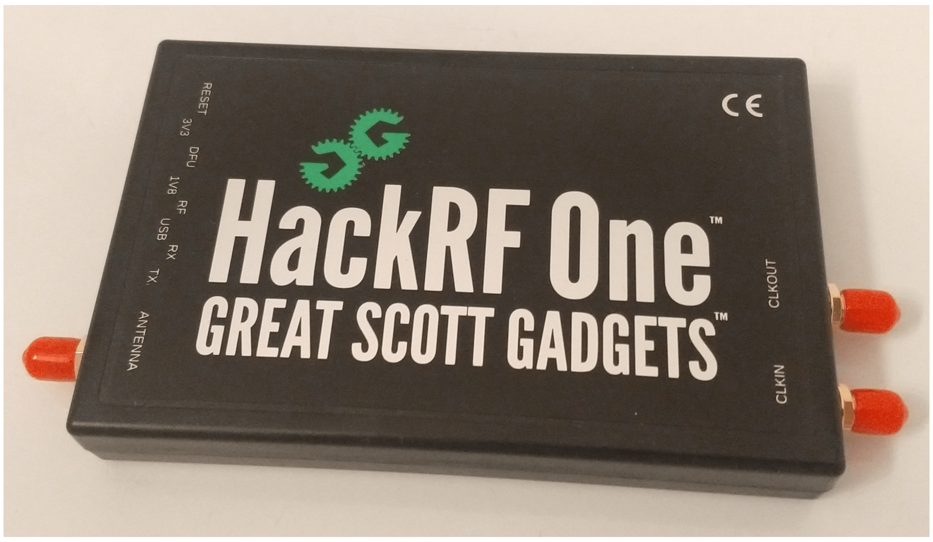


Figure I.1: HackRF One device used.

To better understand this device and its capabilities, we illustrate its functional block diagram in Figure I.2, according to the available specification and project files.

Starting from **1**, the device has a SubMiniature version A (SMA) connector with 50 Ω impedance for using an external antenna using a coaxial cable. The antenna port power is software controlled and is rated to a maximum of 50 mA operating at 3.3 V. The antenna employed in this work is a MR77S Antenna, with a specified operation frequency ranging from 144 MHz to 430 MHz, gain

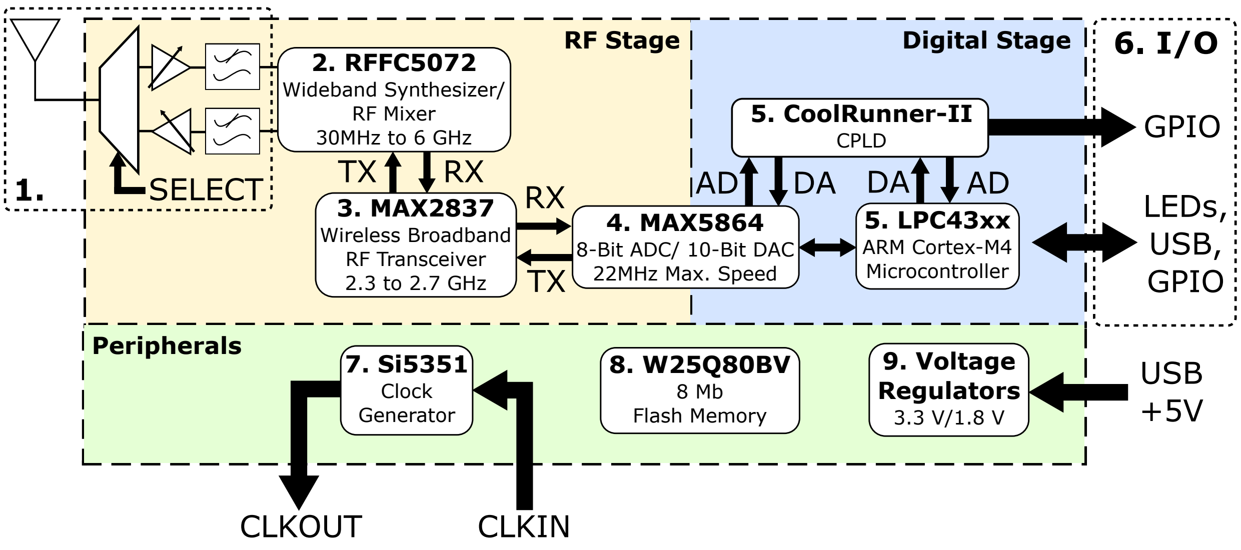


Figure I.2: HackRF One block diagram

of 3.4 dBi at 430 MHz, length of 0.5 m, input impedance of 50 Ω and coaxial cable length of 4 m.

Next, the RF Frontend in **2** the RFFC5072 [73] mixer and wideband synthesizer. This integrated circuit (IC) is constituted of a reconfigurable frequency conversion device, a fractional-N Phase Locked Loop (PLL) synthesizer, a low-noise voltage controlled oscillator (VCO) and a mixer with high linearity. The frequency conversion range is from 30 MHz to 6000 MHz. Other components in the RF front-end include a Low-Noise Amplifier (LNA), additional RF filters and a input multiplexer that selects controls if the device is transmitting or receiving.

The block **3** is the device's MAX2837 RF transceiver [74]. It comprises all the necessary circuitry, including control interfaces, a VCO for IF conversion, a RF-to-baseband receive path, a baseband-to-RF transmit path, and additional detection and filtering circuits. Further, in block **4** is the the MAX5864 ADC/DAC [75]. It is responsible for analog baseband processing and includes filters, dual 8-bit ADCs and dual 10-bit DACs. The specifications relevant to this block is the supported sample rates of 2 Msps to 20 Msps (quadrature) and the set 8-bits resolution.

The digital stage is centered around the LPC43xx Series ARM Cortex-M4 microcontroller (MCU) and CoolRunner-II CPLD [76], both designated as **5**. The MCU performs task such as DSP, input/output operations and interfacing with the analog baseband block trough the MAX5864 ADC/DAC. It also manages the USB 2.0 interface with the host PC. The CPLD is included to assist the MCU, as it is capable of processing digital signals from the ADC/DAC in parallel according to its programmable logic.

The I/O in **6** include external buttons allow users to reset the device, reprogram it and update the firmware. Additionally, external LED indicators display information on the state of power source voltages, USB usage and TX/RX operations. Some GPIOs and a debug interface are available in the device's circuit board, but are not accessible while using its enclosure. The HackRF one uses an USB 2.0 communication interface that enables host PCs to communicate with the device and reprogram it.

Clock generation is performed by an Si5351 IC in **7**, sourcing the timing signals necessary for operation of the SDR system. The HackRF One provides a CLKOUT port using an SMA connector. This port produces a 10 MHz, square wave, from 0 V to 3 V, clock signal intended for high impedance loads. Alongside, a CLKIN port is available for synchronizing the device using an external clock input. Users can then configure clock synchronization to reference the internal oscillator or an external clock source trough the CLKIN port. An 8 Mb flash memory unit **8** is available to store firmware from the MCU and further data. Power is sourced in **9** to the HackRF One through an USB Micro-B connector. The incoming 5 V is regulated to 3.3 V and 1.8 V, according to each component's power requirements.

## I.2 TEST SETUP

For characterization of the SDR equipment and the GNURadio interface [77], we set up two nodes using the HackRF One devices. The first equipment will be positioned in a fixed point acting as the transmitter. Simultaneously, the other device will record transmitted data in different points of interest in a parking lot, as illustrated in Figure I.3. The points of interest are near metallic objects of different sizes, namely a large aluminum fence and a public trash, denoted in Figure I.3 and will be referred as Point 1 and Point 2, respectively, from now on. This allows us to check the power and range capabilities of the HackRF One equipment and verify the effect of multipath in the transmission due to wave reflection on the metallic objects.

The modulation scheme utilized is uncoded Differential BPSK (DBPSK) streaming a test string continuously at the center frequency 433.07 MHz. Both antennas are identical, rated for 3.4 dBi at 430 MHz, length of 0.5 m, cable length of 4 m with SMA connector and input impedance of 50 $\Omega$. Both communication nodes are connected to laptop computers and no external clock synchronization is performed.

## I.3 CHARACTERIZATION STAGE

As described previously, the TX and RX nodes were positioned in a parking lot and used to stream uncoded DBPSK signals carrying a test payload. The former node is in a stationary position and is used as the origin for distance measurements to points of interest. The latter node was put in two different points, with distance of approximately 70 and 18 meters from the stationary node.

For each point, a different power setting was used. Point 1 employed the maximum power settings available in the HackRF one, while Point 2 employed only BB and IF amplification on typical test values, as shown in Table I.1. Next, we present the FFT plots for the raw signals recorded in each of the points of interest in Figures I.4 and I.6, respectively. For improved signal quality, we then apply a DC block filter, center the DBPSK lobe and equalize the signal using a

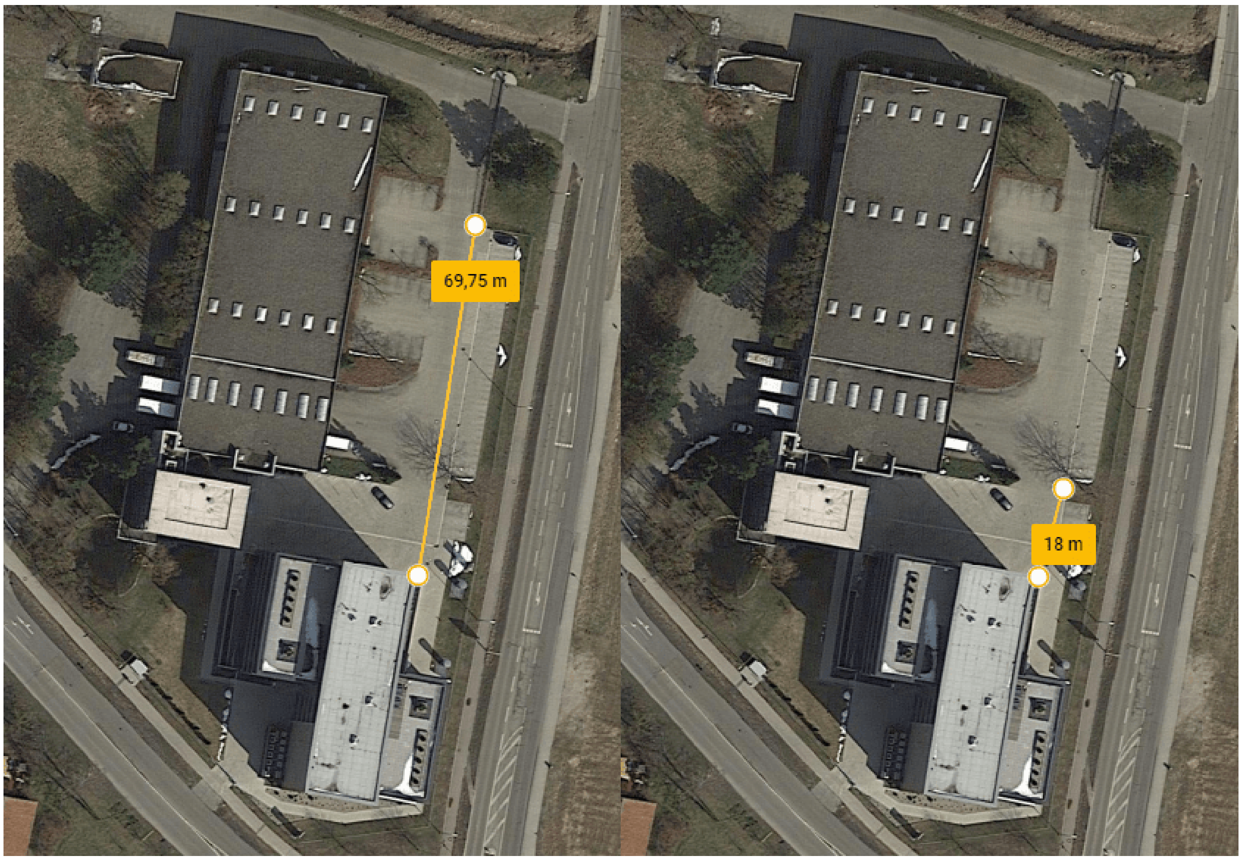


Figure I.3: Distances to the test points from the transmitter's reference.

CMA Equalizer, all in GNURadio. The processed signals' FFT plots are illustrated in Figures I.5 and I.7, respectively. Along with each captured signal, we also present a waterfall plot to illustrate the power variation over a period of time.

Table I.1: Power settings on the HackRF One for TX and RX nodes in each point of interest.

| | **Point 1** | **Point 2** |
|---|---|---|
| **RF** | 0 dB | 0 dB |
| **IF** | 20 dB | 20 dB |
| **BB** | 20 dB | 20 dB |

| | **Point 1** | **Point 2** |
|---|---|---|
| **RF** | 14 dB | 0 dB |
| **IF** | 47 dB | 20 dB |
| **BB** | - | - |

From the information made available by the manufacturer in [72], we obtain the absolute maximum TX power in terms of operating frequency range, as show in Table I.2.

| **Frequency Range** | **Maximum TX power** |
|---|---|
| 1 MHz to 10 MHz | 5 dBm to 15 dBm, generally increasing as frequency increases |
| **10 MHz to 2150 MHz** | **5 dBm to 15 dBm, generally decreasing as frequency increases** |
| 2150 MHz to 2750 MHz | 13 dBm to 15 dBm |
| 2750 MHz to 4000 MHz | 0 dBm to 5 dBm, decreasing as frequency increases |
| 4000 MHz to 6000 MHz | -10 dBm to 0 dBm, generally decreasing as frequency increases |

Table I.2: Maximum HackRF One transmission power in each specified frequency range

We operate in the 433.07 MHz ISM band for this test scenario, we have a maximum transmis-

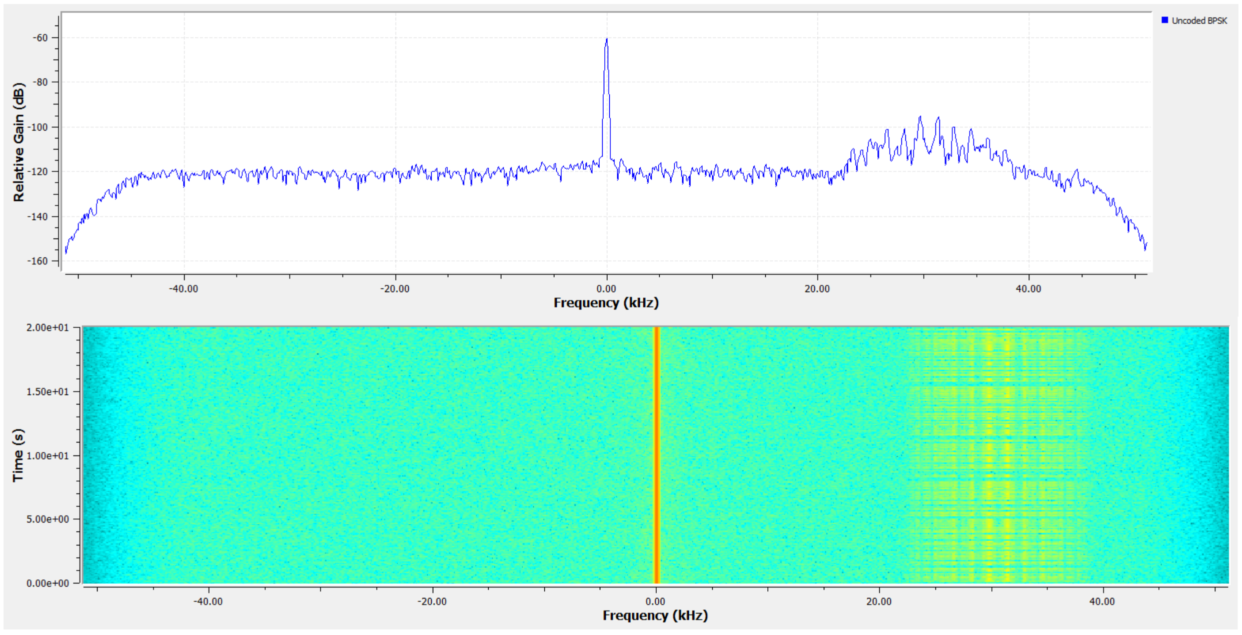


Figure I.4: Raw RF signal captured at Point 1.

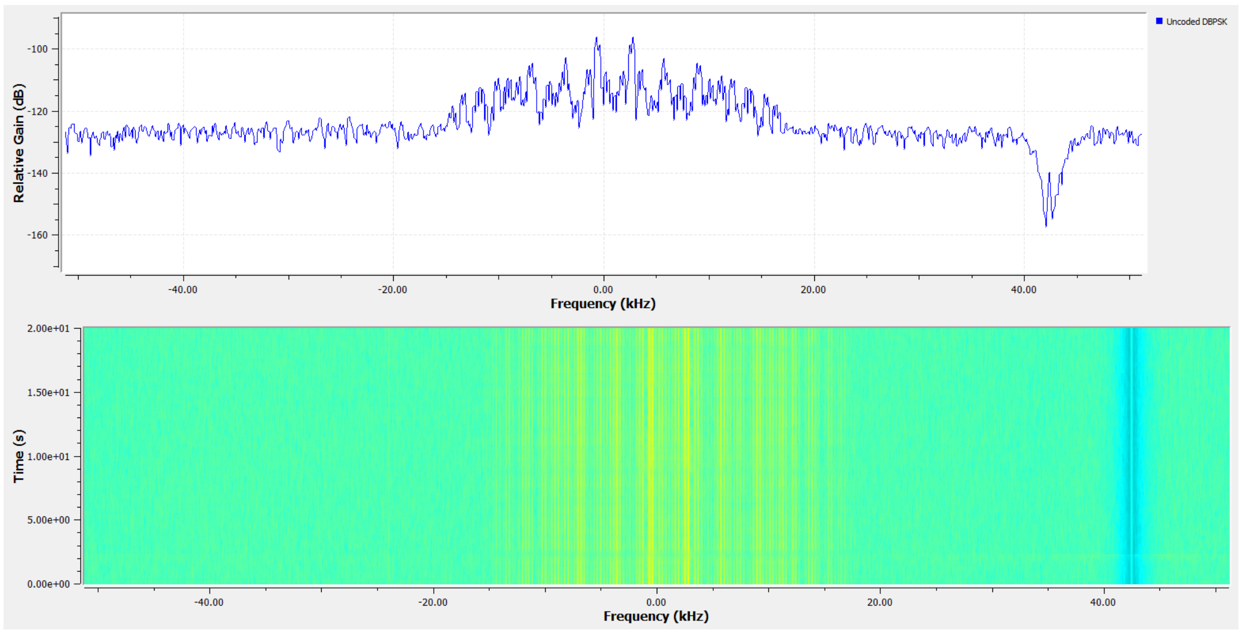


Figure I.5: Processed results from the capture at Point 1.

sion power of 15 dBm. However, the provided TX power data states that the rated value might decrease with increasing frequency. Anyhow, we assume the best case scenario for transmission, considering the transmit power is sufficiently close to 15 dBm at the operating frequency, according to Table I.2.

Another factor to consider is the maximum RX power, which the manufacturer of the HackRF One rates it at -5 dBm in [72]. This power rating is imperative to consider given that exceeding this threshold may result in permanent damage to the equipment. The manufacturer also states that the HackRF One can handle up to 10 dBm of power safely, as long as the frontend RX amplifier

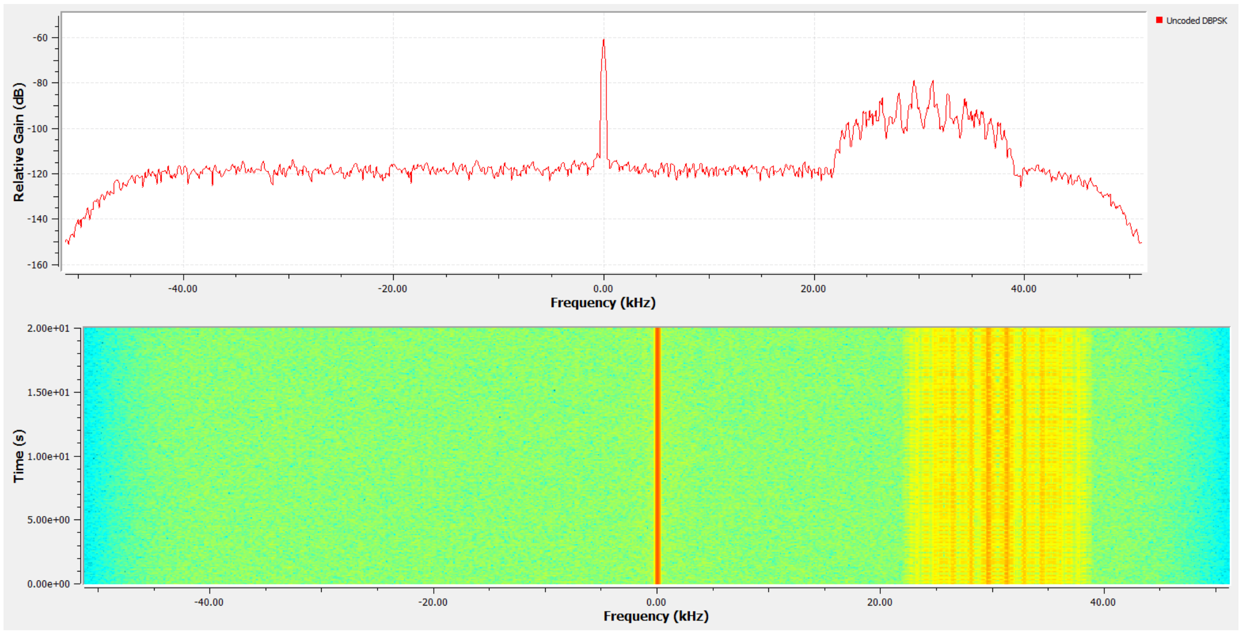


Figure I.6: Raw RF signal captured at Point 2.

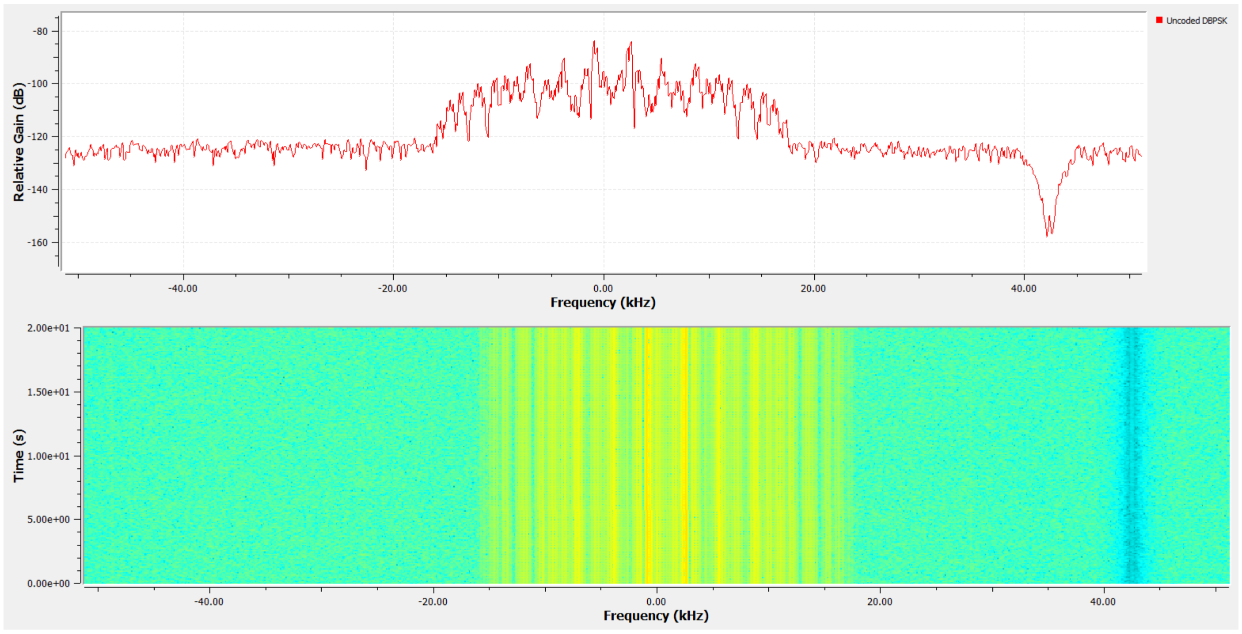


Figure I.7: Processed results from the capture at Point 2.

is disabled. However, since this RF amplification is enabled via software, a misconfiguration in this parameter during testing can render the device unusable. In order to maintain power withing this safe range, the manufacturer recommends the use of external attenuators to mitigate this risk.

The antenna model used during tests have a gain rating of 3.4 dBi at a frequency of 430 MHz. As the transmission in performed at 433.07 MHz, we assume both TX and RX gains are approximately the manufacturer's rated value, as both antennas are of identical model.

With safety in mind, we initially calculate a relation between the external amplification gain

$G_A$ and the RF link range $R$ in order to maintain the maximum RX power just described. Given the RX power $P_R = 5dBm$, transmitting with power $P_T = 15dBm$ at frequency $f = 433.07$ MHz using antennas with gains $G_T = G_R = 3.4dBi$ we use the Friis Equation:

$$P_R|_{dB} = P_T|_{dB} + G_T|_{dBi} + G_R|_{dBi} + 2 \cdot \left. \frac{c}{4\pi f} \right|_{dB} + G_A|_{dB}. \tag{I.1}$$

From Equation I.1, we calculate the following relation:

$$-5dBm = 15dBm+3.4dBi+3.4dBi+2\cdot 10\cdot \log_{10}\left(\frac{c}{4\cdot\pi\cdot f}\right)-2\cdot 10\cdot\log_{10}(R)+G_A|_{dB}, \tag{I.2}$$

$$G_A|_{dB} = 20 \cdot \log_{10}(R) - 1.621dB. \tag{I.3}$$

The relation expressed in I.3 describes the minimum distance between the TX node and all RX nodes must be separated to guarantee safe operation given an external amplifier with gain $G_A|_{dB}$. As the TX node is considered a stationary base station, positioning it in a location far enough from the test track or using it on a tower may achieve the desired separation. Besides, the use of attenuators at RX nodes or using backoff techniques at TX can also contribute to achieve a safe operation power.

Next, we estimate the amplification necessary to achieve the arbitrary operation range of $R = 6km$ from the results contained in Figures I.4 to I.7. The noise floor we observe in the tests performed with the HackRF One is approximately at $-120dB$. The signal detected has a maximum power varying from $-80dB$ — in a close range transmission with no amplification — to $-100dB$ in a longer range amplified transmission, both using BPSK. In this modulation scheme, it is necessary for the signal to have a positive SNR, thus, staying above the noise floor. We must then set a minimum threshold for signal detection above $-120dB$. Assuming the transmission to Point 1 gives a reasonable approximation of the necessary power for correct transmission, we set the reception power to be $P_R = -100dBm$. Again, considering transmission power $P_T = 15dBm$ at frequency $f = 433.07$ MHz using antennas with gains $G_T = G_R = 3.4dBi$ to achieve a range of 6 km, the Friis Equation gives us the external amplification gain necessary $G_A$:

$$-100dBm = 15dBm + 3.4dBi + 3.4dBi + 20 \cdot \log_{10}\left(\frac{c}{4\cdot\pi\cdot 433.07e6 \cdot 6e3}\right) + G_A|_{dB}, \tag{I.4}$$

$$G_A = 21.06dB. \tag{I.5}$$

Considering the amplification is 21.06 dBW, this leads to an equipment of approximately 128 W of power to achieve the desired range. Considering Equation I.3, we have the minimum safe distance between the RF nodes equal to:

$$G_A = 21.06dB = 20 \cdot \log_{10}(R) - 1.621dB. \tag{I.6}$$

$$R = 13m. \tag{I.7}$$

Again, given the TX node is stationary, this separation can be introduced both as vertical — using the base station in a tower — or horizontal distance — positioning the base station far from the test area. This amplification rating gives us an estimation of the power necessary to detect uncoded BPSK signals in the operational parameters described.