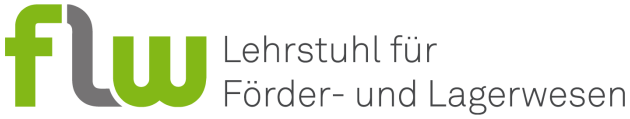
TU DORTMUND UNIVERSITY, DORTMUND, GERMANY

Faculty of Mechanical Engineering



Chair of Material Handling and Warehousing

Master Thesis

Towards 6G-Driven Sensing: Development of Machine Learning-based Radar Object Detection for Logistics Entities

First Examiner: Prof.'in Dr.-Ing. Alice Kirchheim

Second Examiner: Irfan Fachrudin Priyanta, M.Sc.

Presented by: Sahil Sanjay Rajpurkar

Matriculation number: 242976

Course: Master of Science in Automation and Robotics

Issue date: 13. 08. 2024

Submission date: 13. 02. 2025

Dortmund, Germany

"This thesis is dedicated to my hardworking father, and beloved family. Their unwavering love, encouragement, and belief in me have been my constant inspiration throughout my journey. I would also like to extend my deepest appreciation to my esteemed supervisors, Irfan Fachrudin Priyanta and Prof.'in Dr. Alice Kirchheim, whose exceptional mentorship and invaluable insights have been pivotal in shaping the trajectory of my academic pursuits. Furthermore, this work is a tribute to all aspiring robotics engineers worldwide, whose passion for knowledge and relentless dedication continue to drive innovation and progress in this dynamic field."

Abstract

The intra-logistics industry is continuously advancing toward automation technologies to enhance efficiency, accuracy, and reliability in handling goods. Conventional camera and Light Detection and Ranging (LiDAR)-based object detection systems face significant drawbacks, including high costs, environmental sensitivity, privacy concerns, and occlusion issues in logistics warehouses. To overcome these challenges, this thesis explores the potential of millimeter-wave (mmWave) RAdio Detection And Ranging (RADAR) sensors, particularly in the context of emerging 6G networks, to improve object detection in logistics environments. The research focuses on the development of a machine learning-based object detection system using the Texas Instruments IWR6843ISK mmWave radar sensor. The study begins with the integration of the radar sensor into a logistics setup, followed by the collection and preprocessing of radar data for various logistic scenarios. Machine learning models, including YOLOv7, Detectron2, and OpenPCdet, are trained on the preprocessed dataset to evaluate their effectiveness in detecting logistics entities such as forklifts, logistic robots, and small load carriers (Small Load Carriers (Kleinladungsträger in German) (KLT)s). The methodology involves offline and online testing of the models to assess accuracy, precision, and robustness in real-time logistic operations. The results demonstrate that mmWave radar technology, combined with machine learning methods, significantly enhances object detection capabilities, particularly in dynamic and cluttered environments. The research also highlights the advantages of mmWave radar sensors, including high resolution, high-speed data transfer, resilience to environmental changes, and the ability to function in low-visibility conditions. The findings suggest that the limitations of conventional camera and LiDAR-based systems can be addressed by integrating mmWave radar sensors with machine learning, paving the way for more efficient and reliable logistics automation. This research contributes to the development of 6G-driven sensing technologies by providing a scalable and cost-effective solution for modern logistics operations. Future work could explore the use of additional radar parameters, multiple sensors, and advanced machine learning techniques to further improve detection accuracy and expand the system's applicability in industrial Internet of Things (IoT) and smart logistics.

Acknowledgements

I would like to express my sincere gratitude to the exceptional individuals who have played a significant role in my thesis journey. I am immensely grateful to my supervisor, Prof.'in Dr. Alice Kirchheim and Irfan Fachrudin Priyanta M.Sc., for their invaluable guidance, insightful feedback, and unwavering support throughout the development of this thesis. Their influence not only helped me achieve my thesis objectives but also inspired me to strive for excellence and surpass predefined goals.

I am also deeply thankful to my friends and family members for their unwavering support and understanding, which kept my spirits high during the thesis. My heartfelt gratitude goes to my co-workers who were always there when I needed assistance. I would also like to acknowledge the contributions of researchers in this field whose published studies provided crucial insights and references that were instrumental in shaping the framework of this thesis. Their pioneering work served as a strong foundation for my research and greatly enriched its quality.

Lastly, I would like to extend my thanks to the Chair of Material Handling & Warehousing (FLW) at TU Dortmund University for granting me this incredible opportunity. This supportive environment fostered my growth as a researcher and provided me with the resources needed to complete my thesis successfully.

Contents

1 Introduction **1**
- 1.1 Problem Definition . . . 1
- 1.2 Objective . . . 3
- 1.3 Approach . . . 3
- 1.4 Overview . . . 4

2 Fundamentals **5**
- 2.1 mmWave Radar Sensor . . . 5
  - 2.1.1 Frequency-Modulated Continuous Wave Radar . . . 5
  - 2.1.2 Range, Velocity, and Angle Measurement . . . 6
  - 2.1.3 Sensor Description . . . 9
- 2.2 Data Acquisition and Representation . . . 10
  - 2.2.1 Data Acquisition . . . 11
  - 2.2.2 Data Representation . . . 12
- 2.3 Correlation of Radar with 6G . . . 13
  - 2.3.1 Joint Communication and Sensing (JCAS) . . . 14
  - 2.3.2 Impact of 6G on Radar Systems . . . 14
- 2.4 Radar Object Detection Concept . . . 15
  - 2.4.1 Machine Learning for Object Detection . . . 15
  - 2.4.2 Sensor Fusion for Object Detection . . . 16

3 State of the Art **18**
- 3.1 Intralogistics: Current Sensing and Localization Technologies . . . 18
- 3.2 The Role of Radar . . . 19
- 3.3 State-of-the-Art Radar Applications . . . 19
- 3.4 Machine Learning Models for Object Detection . . . 20

4 Methodology **23**
- 4.1 Model Selection . . . 23
  - 4.1.1 Yolov7 for Object Detection . . . 24
  - 4.1.2 Detectron2 Framework . . . 25
- 4.2 Design Pipeline . . . 27
- 4.3 Dataset Preparation . . . 28
  - 4.3.1 Data Annotation . . . 28
  - 4.3.2 Data Augmentation . . . 29
  - 4.3.3 Data Splitting . . . 29
- 4.4 Model Training . . . 29
  - 4.4.1 Training Yolov7 . . . 30
  - 4.4.2 Training Detectron2 Models . . . 30

4.5 Sensor Integration . . . 31
4.5.1 Flashing the IWR6843ISK Evaluation Module . . . 31
4.5.2 Hardware Setup . . . 32
4.5.3 Sensor Configuration . . . 33
4.5.4 Available Code Repositories . . . 33

5 Experiments and Results **35**
5.1 Dataset Collection . . . 35
5.1.1 Introduction to Objects and Environment . . . 35
5.1.2 Setup for Dataset Collection . . . 36
5.1.3 Dataset Collection Details . . . 37
5.1.4 Example of Logistic Scenarios . . . 38
5.2 Preprocessing of Collected Dataset . . . 38
5.2.1 Range-Azimuth 2D Histogram . . . 38
5.2.2 3D Point Cloud . . . 39
5.2.3 Filtering Multi-Paths and Boundary Constraints . . . 42
5.2.4 Data annotation . . . 43
5.2.5 Data splitting . . . 45
5.3 Object Detection with Range-Azimuth . . . 46
5.3.1 Offline Model Testing . . . 46
5.3.2 Analysis and Results for Offline Model Testing: Radar Azimuth . . . 47
5.3.3 Comparative Analysis of Models . . . 52
5.3.4 Online Model Testing . . . 53
5.4 Object Detection with 3D Point Cloud . . . 56
5.4.1 Offline Model Testing . . . 56
5.4.2 Analysis and Results for Offline Model Testing: 3D Point Cloud . . . 57
5.4.3 Result Comparison with existing motion detection system . . . 61
5.5 Algorithms and Datasets . . . 62

6 Conclusion and Future Work **63**
6.1 Conclusion . . . 63
6.2 Future Work . . . 64

1 Introduction

Logistics warehouse play a pivotal role in the supply chain by serving as hubs for the reception, storage, and dispatch of goods. The efficiency of these facilities is a cornerstone of modern logistics, influencing the speed, accuracy, and reliability of the supply chain. With the consistent increase in the volume of goods being handled, traditional manual and semi-automated methods prove inadequate in meeting the demands of modern logistics operations. To address these challenges, the industry rapidly moving toward automation, integrating technologies such as automated guided vehicles (Automated Guided Vehicle (AGV)s), robotics, and automated storage and retrieval systems (AS/RS) [1, 2]. These innovations not only improve operations, but they help address labor shortages, save operating expenses, and eliminate human error. Despite these advantages, obtaining peak efficiency in automated warehouses is hampered by the limits of traditional object identification systems, which rely mainly on camera-based technology. These systems are frequently costly, constrained by environmental circumstances, and prone to errors in dynamic and complicated warehouse environments [3].

To mitigate these limitations, radar-based sensing technologies plays crucial role, particularly mmWave radar sensor, valued as robust alternative. mmWave radar sensor operates in the millimeter-wave spectrum (30 GHz to 300 GHz), offers high-resolution sensing capabilities and demonstrates durability to environmental variations. It allows for precise long-range detection, classification, and tracking of logistics entities, such as pallets, forklift and KLT(Kleinladungsträger), even in cluttered or obstructed scenarios [4]. Furthermore, principle improvements in mmWave Frequency Modulated Continuous Wave (Frequency Modulated Continuous Wave (FMCW)) radar enhance its effectiveness by providing additional measurements like range, velocity, and direction, irrespective of adverse weather or lighting conditions [5]. These features make mmWave radar a impactful part of modern logistics automation. Furthermore, this shift toward higher frequency bands, including mmWave and terahertz (THz), within emerging 6G networks is expected to further boost detection accuracy and environmental mapping capabilities, enabling innovative applications in industrial IoT and smart logistics [6, 7]. Against this backdrop, the study suggests the creation of a machine learning-based radar object identification system that takes advantage of mmWave radar's distinct capabilities to improve detection accuracy, efficiency, and reliability in logistics warehouses. This methodology seeks to overcome the major issues of traditional detection methods, providing smooth and efficient warehouse operations in an increasingly automated environment.

1.1 Problem Definition

The logistics industry faces multiple challenges with traditional camera-based object detection systems, including high costs, limited fields of view, and susceptibility to blockage by objects

that are either stationary or moving. These systems are further hindered by their dependence on illumination conditions and privacy concerns. Low-resolution cameras struggle to record micro motions [8], reducing their performance in dynamic logistics conditions. While alternate devices such as LiDAR sensors and wireless sensing systems offer certain advantages, but also comes with limitations. LiDAR is capable of detecting micro-motions, but it has narrow sensing range and dependencies on weather conditions [9]. Wireless technologies such as Wi-Fi, Ultra-Wideband (UWB), and LoRa offer other choices but come with their own problems. Wi-Fi, while cost-effective and widely available, suffers from lower spatial resolution and potential interference [10]. UWB provides high precision and robustness to occlusion but is limited in range and requires dedicated infrastructure [11]. LoRa offers extensive coverage and low power consumption but is constrained by low data rates and limited accuracy in motion detection [12]. Solving these issues is key to making autonomous logistics systems more efficient and accurate.

Radar delivers long range for object detection and classification across various environmental condition with high-resolution sensing, and robust data processing capabilities, making it highly effective in environment like indoor logistic hubs [13]. The use of high-frequency radar sensors, specifically operating within 60 GHz and 300 GHz, is well-suited for real-time processing [14]. These benefits overcome the limitations of camera and LiDAR-based systems by offering precise velocity and range measurements, reliable performance, and resistance to harsh environments. It is a sensible option for logistics because of its capacity to operate without hiccups in cluttered areas and with poor visibility. Radar systems are also scalable and reasonably priced, making them flexible enough to meet a range of industrial demands. These features make radar an ideal solution for modern logistics environments.

As industries moving towards 6G technology, there is a notable gap in the application of radar-based sensing within intralogistics [15]. When combined with 6G networks, radar technology presents a substantial opportunity to close this gap. The high-frequency radar systems enabled by 6G can provide precise object detection, tracking, and environmental mapping in complex warehouse environments. Using machine learning (ML)-based algorithms is crucial to maximizing these capabilities. ML algorithms improve intralogistics efficiency by optimizing real-time decision-making and autonomous navigation. The combination of radar technology and machine learning (ML) has the potential to transform intralogistics by offering flexible, high-resolution solutions that can adapt to the changing needs of contemporary supply chains, especially since 6G is anticipated to support AI-driven frameworks [16].

Therefore, the problems to be addressed in this thesis are:

1. How can mmWave radar technology help intralogistics overcome the limitations of camera and LIDAR-based object detection systems?
2. How can high-frequency operated radar sensors impact real-time processing capabilities in logistic application?
3. How can the combination of mmWave technology and machine learning improve the development of 6G-driver sensing for autonomous logistic and other industrial applications?

In the age of 6G technology, The aim of these research questions is to improve the reliability and efficiency of autonomous logistics systems. The proposed research focus on developing a machine learning-based radar object detection system using mmWave radar technology, ensuring it meets the accuracy, efficiency, and real-time processing requirements of modern logistics operations.

1.2 Objective

The Master thesis is divided into the following objectives:

1. Integrate the Texas Instruments (TI's) IWR6843ISK, a 60GHz mmWave radar sensor, into the setup for accurate radar data collection.
2. Collect mmWave radar sensor data for various logistic scenarios and pre-process the collected data to prepare it for machine learning model training.
3. Train machine learning models on the pre-processed radar dataset, evaluating backbone algorithms such as Convolutional Neural Network (CNN)s and Faster Region-based Convolutional Neural Network (RCNN) to identify the most effective method for object detection in intralogistics environments.
4. Evaluate the accuracy, precision, and efficiency of the selected machine learning model for radar-based detection.
5. Test and validate the model in offline and online environments to assess its robustness and performance in real-time processes.

1.3 Approach

The proposed approach of this research is focusing on leveraging of mmWave radar technology and machine learning methods to overcome the limitation of traditional object detection system in modern autonomous logistic. The research begin with the integration of the Texas Instruments(TI's) IWR6843ISK, a 60 GHz mmWave radar sensor, into the experiment. In a variety of logistics scenarios, this sensor's high-resolution range and velocity readings allow for the collection of reliable radar data. After the radar signals are gathered, they are processed to produce extensive datasets. These datasets are then preprocessed to extract important features needed for object detection. Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) are two examples of machine learning techniques used to examine these datasets. The research identifies the best algorithm for accurate and efficient radar-based object detection through repeated testing and parameter adjustments. The accuracy, precision, and robustness of the chosen model are tested in the offline evaluation phase under various environmental circumstances and logistics use cases. By tackling issues like occlusion, poor visibility, and cluttered environments, these initiatives guarantee that the radar detection system functions reliably in dynamic warehouse environments [17].

The study also looks into possible developments to improve the system's functionality. This involve testing the model online in similar and different environment to check the real-time performance and adaptability under operational conditions. The bridge between mmWave radar technology and machine learning enhance with state-of-art framework like You Only Look Once (YOLO)v7 [18] and Detectron2 [19], which provide object detection pipelines suitable for radar dataset. The transition towards 6G technologies, which will facilitate AI-driven analytics and high-frequency radar systems for sophisticated sensing and decision-making. The proposed approach toward 6G sensing using advancement of the radar technologies, helping to improve autonomous logistics and drive future advancements in industrial IoT and smart logistics.

1.4 Overview

This thesis consist of six chapter, focusing on the development and evaluation of machine learning-based radar object detection system designed for logistics automation. Chapter 1 - Introduction, Elaborate the traditional object detection method use in modern logistic, limitation of these existing systems and outline proposed approach to overcome this limitation. Chapter 2 - Fundamentals, covers the principle of mmWave radar, including operating mechanism, data acquition, and traditional radar-based object detection methodologies. Also highlights the potential of the benefits of using artificial neural networks to improve detection performance. Key gaps addressed in this research are identified in Chapter 3 - State of the Art, which reviews recent developments in radar-based sensing and neural network integration.

Chapter 4 - Methodology, outlines the model selection, dataset preparation, experimental setup and pipeline for implementing radar-based object detection system. Chapter 5 -Experiments and Results, the proposed models implementation by conducting offline and online tests, comparing their performance to baseline methods, and tackling challenges related to real-time implementation. Finally, Chapter 6, Conclusion and Future Work, highlights the importance of radar-based automation in contemporary logistics by summarizing the research findings, talking about limitations, and suggesting directions for further investigation.

2 Fundamentals

This chapter explores the fundamental aspects of object detection using mmWave radar sensors, with a particular emphasis on integrating machine learning methods. The operating principles behind mmWave radar technology are examined, including Frequency Modulated Continuous Wave (FMCW) radar and joint communication and sensing (Joint Communication and Sensing (JCAS)). Additionally, the machine learning techniques that facilitate object detection are discussed. A review of traditional object detection methods is provided to establish the foundation for understanding the complexities of modern approaches. These classical methods pave the way for the development of modern deep-learning techniques. Building on this foundation, subsequent chapters delve deeper into these concepts, focusing on their application to detecting objects in logistics environments using mmWave radar sensors.

2.1 mmWave Radar Sensor

Radio Detection and Ranging (RADAR) is a system that uses electromagnetic waves to determine the range, velocity, and angle of objects in the surrounding environment. A RADAR system transmits an electromagnetic wave signal that reflects off objects in its path. By analyzing the reflected signal through signal processing techniques, RADAR systems can precisely determine key parameters such as range, velocity, and angle. These properties make RADAR an essential sensing technology, often referred to as RADAR sensors [20]. Millimeter-wave (mmWave) radar sensors operate in the millimeter-wave band, which refers to electromagnetic waves with frequencies ranging from 30 to 300 GHz and wavelengths between 1 mm and 10 mm. Due to the high propagation speed of these waves (equivalent to the speed of light in air), mmWave radar sensors are highly accurate, particularly for detecting objects at long distances. Additionally, mmWave radar sensors are robust in adverse environmental conditions such as fog, dust, and varying light levels, making them suitable for industrial and autonomous systems [21]. Compared to other sensing technologies, such as laser sensors, these sensors offer similar accuracy but at significantly lower costs. Furthermore, they are highly versatile and flexible, with applications in radar systems, logistic systems, and autonomous vehicles [22].

2.1.1 Frequency-Modulated Continuous Wave Radar

Frequency-Modulated Continuous Wave (FMCW) radar is a specialized type of radar sensor that continuously transmits and receives signals, unlike traditional pulsed-radar systems that emit short bursts of energy at intervals. In FMCW radar, the transmitted (Tx) signal is a linearly frequency-modulated wave that periodically sweeps over a bandwidth $B = (f_{\text{stop}} - f_{\text{start}})$ within

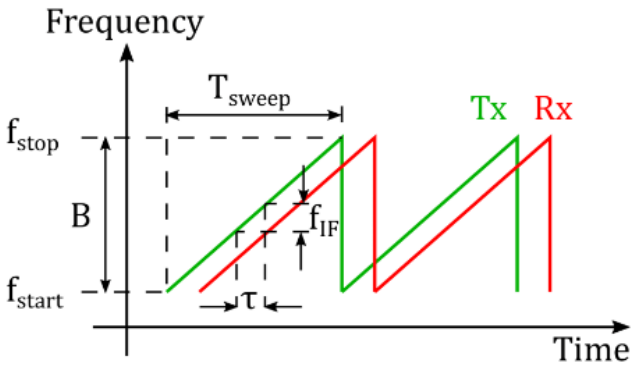


Figure 2.1: Principle of Frequency-Modulated Continuous Wave (FMCW) radar [22].

a time period T_{sweep}. The received (Rx) signal is a time-delayed version of the Tx signal, with the delay $\tau = \frac{2R}{c_0}$ corresponding to the round-trip time of the radar signal to and from the target, where R is the target distance and c_0 is the speed of light [23]. The difference between the Tx and Rx signals produces an intermediate frequency (IF) signal, which is used to calculate the target distance. The relationship is given by Equation 2.1. where B, T_{sweep}, and c_0 are fixed parameters for a given radar system. The accuracy of distance measurement in an FMCW radar depends on how precisely the value of f_{IF} is estimated. Enhanced accuracy can be achieved using advanced frequency and phase-estimation methods to analyze the IF signal samples [24, 25].

$$f_{\text{IF}} = \frac{2RB}{c_0 T_{\text{sweep}}} \tag{2.1}$$

2.1.2 Range, Velocity, and Angle Measurement

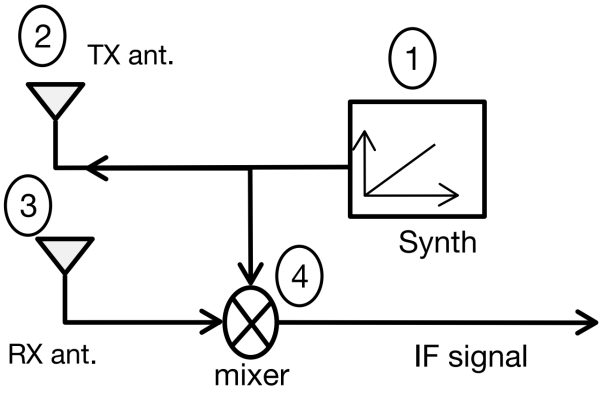


Figure 2.2: Simplified block diagram of an FMCW radar system [16].

The radar transmits a chirp signal, which, upon reflection from an object, is captured by the receiving antenna (RX). Figure 2.2 illustrates the core components of an FMCW radar system. First, a synthesizer generates the chirp signal, which is then transmitted through the TX antenna. After the chirp reflects off an object, the reflected signal is received by the RX antenna. The transmitted and received signals are subsequently mixed, generating an intermediate frequency (IF) signal, which is used for further processing to determine the range, velocity, and angle of the object.

Range Measurement: The range to an object is determined by analyzing the frequency content

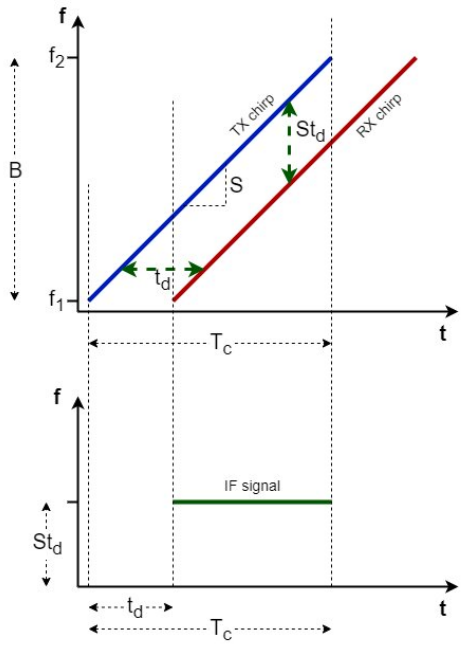


Figure 2.3: Transmitted and received chirps for range measurement [26].

of the intermediate frequency (IF) signal. As shown in Figure 2.3, the transmitted and received chirps are plotted as a function of time for a single detected object. From the figure, it is evident that the received chirp is a time-delayed version of the transmitted chirp, with a delay denoted as τ. This time delay, τ, is related to the distance R to the detected object and the speed of light C by the equation:

$$\tau = \frac{2d}{C} \tag{2.2}$$

The initial phase of the IF signal, denoted as ϕ_0, is related to the distance by:

$$\phi_0 = \frac{4\pi d}{\lambda} \tag{2.3}$$

The actual range d is then computed using the formula:

$$d = \frac{C \times f_{\text{IF}}}{2S} \tag{2.4}$$

Velocity Measurement: To measure velocity, an FMCW radar transmits two chirps separated by a time interval T_c. The reflected chirps are processed using range-FFT, which detects the range of the object. The range-FFT results in peaks at the same location for both chirps, but with different phases due to the object's motion. This phase difference corresponds to a velocity of vT_c, as shown in Figure 2.4. The phase difference is given by Equation as:

$$\Delta\Phi = \frac{4\pi T_c}{\lambda} \tag{2.5}$$

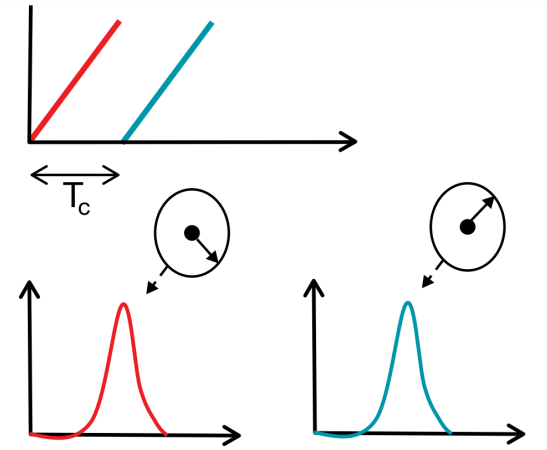


Figure 2.4: Illustration of velocity measurement with two chirps [16].

From this phase difference, velocity is calculated using:

$$v = \frac{\lambda \Delta \Phi}{4\pi T_c} \tag{2.6}$$

Angle Measurement: For angle estimation, multiple antennas or an antenna array are essential. As shown in Figure 2.5, the system consists of one transmitting (TX) antenna and two receiving (RX) antennas. The transmitted chirp echoes back to both RX antennas, but the distance it traverses to each receiver differs by a distance Δd. Suppose the distance between the two antennas is l. By using trigonometric functions, Δd is expressed as $\Delta d = l \sin(\theta)$, where θ represents the angle of arrival (AoA). This change in distance causes a phase shift in the received signal at each antenna. These characteristics are analyzed and utilized for angle estimation. Let $\Delta d = l \sin(\theta)$

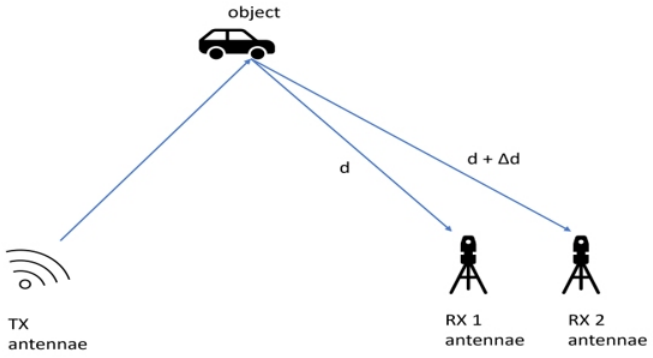


Figure 2.5: Angle estimation in FMCW radar using multiple antennas [20].

be replaced with the phase shift relation Equation 2.3. Rearranging the equation to solve for θ, the angle of arrival (AoA) is obtained.

$$\theta = \sin^{-1}\left(\frac{\lambda \phi}{2\pi l}\right) \tag{2.7}$$

This demonstrates that the change in angle θ exhibits a nonlinear relationship with the phase change ϕ. However, for small angles, $\sin(\theta) \approx \theta$ is a valid approximation, which linearizes the relationship and makes it suitable for accurate angle estimation. The maximum angular field of

view, $\theta_{\max}$, is derived from the condition $\frac{\lambda\phi}{2\pi l} = 1$. The theoretical maximum angular field of view is 180° or π radians. When $\phi = \pi$, it follows that $l = \frac{\lambda}{2}$. This defines the optimal spacing between antennas for achieving maximum angular resolution [20].

Millimeter-wave (mmWave) sensor parameters vary by model and manufacturer. Key parameters include range resolution, which measures the radar's ability to distinguish targets at different distances but the same azimuth, and velocity resolution, which is influenced by the signal-to-noise ratio. The field of view (Field of View (FOV)) defines the radar's angular coverage in azimuth and elevation planes. Angular resolution helps to differentiate targets at the same distance but different azimuth angles. Table 2.1 compares 24GHz, 60GHz, and 77GHz sensors in terms of operating frequency, resolution, antenna size, and applications, with 24GHz suited for basic scenarios, 60GHz for industrial and security uses, and 77GHz mainly for transportation and automotive fields [21].

mmWave Sensor Model	**24GHz**	**60GHz**	**77GHz**
Operating Frequency	24GHz	60GHz–64GHz	76GHz–81GHz
Distance Resolution (cm)	Minimum 60	Minimum 3.75	Minimum 3.75
Speed Resolution	Low	High (2.5 times that of 24G radar)	High (2.5 times that of 24G radar)
Antenna Size	Large	Smaller (about 6 times smaller than 24G antenna size)	Smaller (about 6 times smaller than 24G antenna size)

Table 2.1: Comparison of Millimeter-Wave Sensor Models [21]

2.1.3 Sensor Description

The IWR6843 ISK (Industrial Sensor Kit) Evaluation Module by Texas Instruments is designed to evaluate the performance of the IWR6843 mmWave sensor in industrial applications. This evaluation module (EVM) integrates the IWR6843 radar sensor with antennas and peripheral interfaces, providing a comprehensive platform for testing applications like motion detection, object tracking, and proximity sensing[27]. The IWR6843 ISK integrates the IWR6843 mmWave radar sensor along with an antenna array and peripheral interfaces, making it easy to evaluate various applications. The onboard antenna array features 3 transmit (TX) and 4 receive (RX) antennas, which support accurate angle measurement and object tracking across a wide field of view.

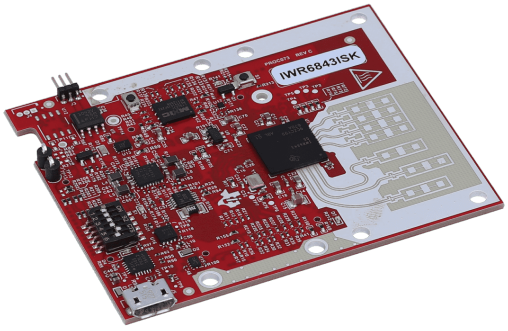


Figure 2.6: Texas Instruments IWR6843 ISK Evaluation Module[27].

Feature	Specification
Frequency Range	60-64 GHz
Antenna Configuration	3 Transmit, 4 Receive Channels
Maximum Range	Up to 100 meters
Power Supply	5V DC via Micro USB
Interfaces	UART, SPI, I2C, JTAG, CAN-FD
Processor	Arm Cortex-R4F, DSP (C674x)
Memory	1.75 MB
Functional Safety Compliance	SIL-2 Certified
Sampling Rate	25 Msps
Intermediate Frequency	10 MHz
Operating Temperature	-40°C to 85°C

Table 2.2: IWR6843 ISK Evaluation Module Specifications[27]

The EVM provides multiple communication interfaces, including UART, SPI, I2C, and CAN-FD, allowing it to be integrated into a variety of systems for real-time radar data processing. Additionally, the IWR6843 ISK includes built-in hardware accelerators for fast Fourier transform (FFT), constant false alarm rate (Constant False Alarm Rate (CFAR)) detection, and other advanced signal processing tasks. The evaluation module supports autonomous operation by allowing user applications to run directly from the onboard QSPI flash. This makes the IWR6843 ISK suitable for deployment in standalone or embedded systems without the need for an external host.

The MMWAVEICBOOST is an add-on carrier board designed to be used with the IWR6843 ISK and other xWR6843 family devices. It enables additional interfaces, debugging capabilities, and supports data capture via the DCA1000EVM. The MMWAVEICBOOST facilitates development and evaluation by offering expanded connectivity and debugging interfaces[28] .

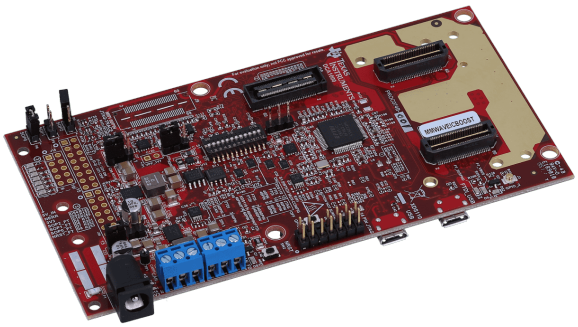


Figure 2.7: Texas Instruments MMWAVEICBOOST Board[28] .

2.2 Data Acquisition and Representation

Data acquisition in FMCW radar involves capturing the transmitted and reflected signals to extract information about the object's distance, velocity, and angle. This data is then represented in various forms, such as range-Doppler and 3D point clouds, to provide insights for different applications. These representations enable radar systems to deliver accurate and real-time information, essential for tasks like object detection and tracking in dynamic environments.

2.2.1 Data Acquisition

The basic data acquisition technique used by an FMCW radar sensor involves three key components: RF (Radio Frequency), analog, and digital electronic blocks. These components work together to acquire and process the data from the sensor. In the RF block, the radar sensor includes a transmit antenna (TX) and a receive antenna (RX). The TX antenna transmits electromagnetic waves, and the RX antenna receives the reflected waves from objects in the environment. The time delay between the transmitted and received signals corresponds to the distance the signal has traveled. This information is passed to the next stage—the analog block. In the analog block, the received signal (referred to as the intermediate frequency (IF) signal) passes through a low-pass filter (Low Pass Filter (LPF)) to eliminate unwanted high-frequency noise. The filtered signal is then digitized by an analog-to-digital converter (Analog-to-Digital Converter (ADC)), converting it into digital form. Once digitized, the data is sent to the digital processing block for further analysis. In the digital block, the captured data undergoes signal

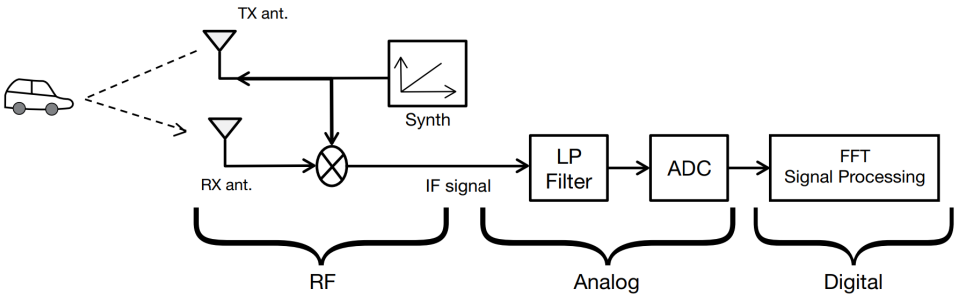


Figure 2.8: Block diagram of an FMCW sensor[16].

processing, typically through Fast Fourier Transform (Fast Fourier Transform (FFT)) operations, as illustrated in Figure 2.8. This data processing reveals information about the object's range, velocity, and angle. To explain the data acquisition process in more detail, consider the frequency difference between the transmitted and received signals. This difference corresponds to the signal's propagation time, which is used to determine the object's distance. The transmitted signal, $S_{Tx}(t)$, and the received signal, $S_{Rx}(t)$, are expressed as:

$$S_{Tx}(t) = \exp[j(2\pi f_c t + \pi K t^2)]$$

$$S_{Rx}(t) = \alpha S_{Tx}(t - \frac{2R(t)}{c})$$

where α represents the path loss, f_c is the chirp start frequency, and K is the chirp slope. The distance between the radar and the object is denoted by $R(t)$, and c is the speed of light. By mixing the transmitted and received signals, the beat frequency signal $s(t)$ is obtained:

$$s(t) = S_{Tx}(t)S_{Rx}(t)^* \approx \alpha \exp\left[j\frac{4\pi(f_c + Kt)R(t)}{c}\right]$$

The phase of the beat signal $s(t)$ reveals the object's distance $R(t)$. To separate the signals reflected from different ranges, a Range-FFT operation is applied to the samples of $s(t)$, producing a range spectrum[8], as shown in Figure 2.9.

After determining the object's range, the radar performs a Doppler-FFT on the signals within the selected range bin to estimate the object's velocity, resulting in a range-Doppler spectrum. The

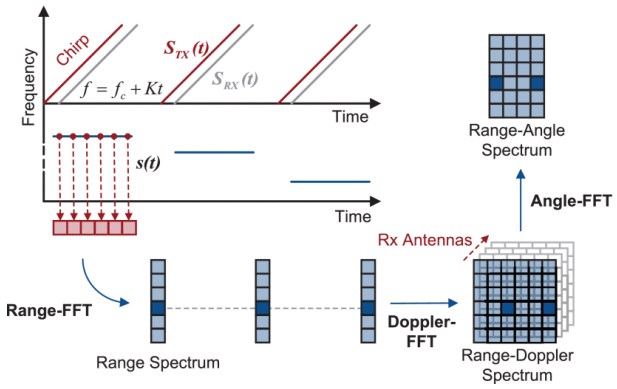


Figure 2.9: Illustration of data capture based on FMCW radar[8].

range-Doppler spectrum provides both distance and velocity information about the object[29]. Finally, the position of the object is determined using multiple receiver antennas and performing an Angle-FFT on the received signals, generating the range-angle spectrum. This step allows the radar to estimate the angle of arrival (AoA) and determine the object's position in space. For improved angular resolution, beamforming techniques can be applied instead of Angle-FFT, concentrating the signal energy in the desired direction.

2.2.2 Data Representation

The data captured by the mmWave radar sensor can be represented in various forms depending on the type of information being analyzed. These representations include range-Doppler, range-angle, phase waveform, and point cloud data, each offering unique insights for different sensing tasks. This study focuses on the range-angle and 3D point cloud representations.

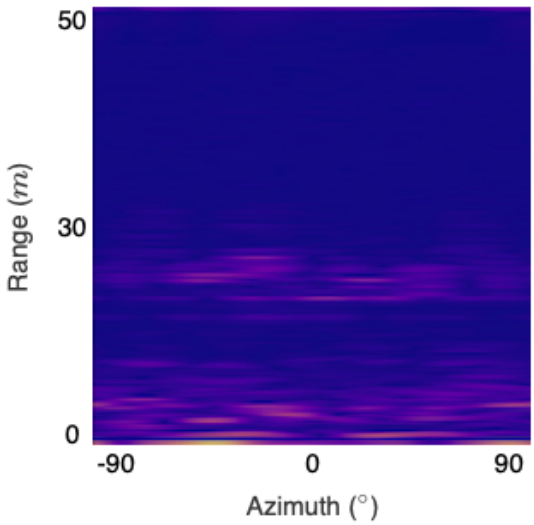


Figure 2.10: Range-Angle spectrum example showing distance and angular position of objects [30].

Range-Azimuth: The range-azimuth spectrum, is used to determine the angular position of objects. By employing multiple receiver antennas, the radar can detect the phase differences between received signals, allowing it to compute the angle of arrival (AoA). An angle-FFT is

applied to derive the range-angle spectrum, which gives the spatial location of objects in relation to the radar. Figure 2.10 shows a range-angle spectra, which are crucial for tasks like detecting and tracking moving objects in space.

3D point Cloud: A point cloud is a 3D representation of an object's surface, much like the output from LiDAR or 3D imaging systems. In mmWave radar, a point cloud is generated by performing operations such as range-FFT, Moving Target Indication (MTI), and beamforming. The point cloud consists of numerous data points, each representing a reflection point on the object's surface. This output is particularly useful for tasks like human imaging, gesture recognition, and 3D mapping. To generate a point cloud, several steps are taken, including FFT processing and advanced signal filtering. Figure 2.11 illustrates the typical flowchart for point cloud generation using mmWave radar[31]

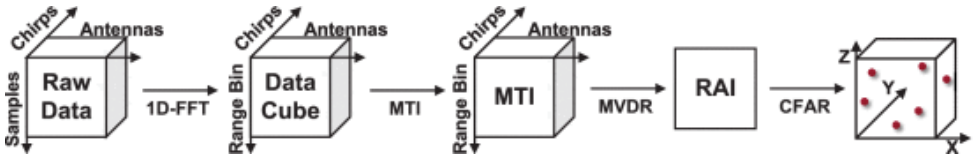
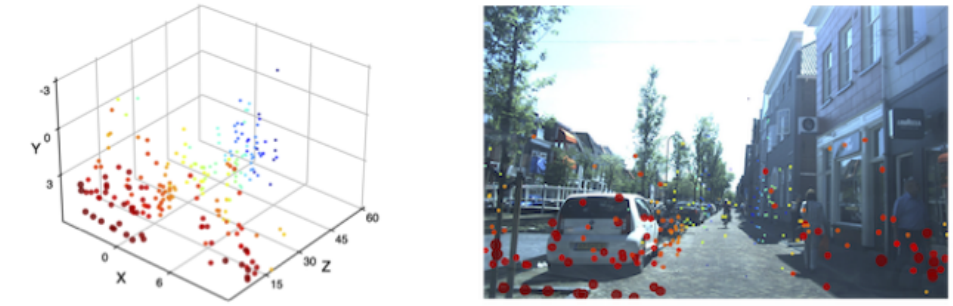


Figure 2.11: Flowchart of point cloud generation using mmWave radar[8].



(a) Point clouds in 3D coordinates (b) Point clouds on image plane

Figure 2.12: Example of a point cloud generated by mmWave radar[8].

2.3 Correlation of Radar with 6G

Radar and communication technologies have played significant roles in modern life since the early 21st century. Integrating radar technology with 6G networks opens up new possibilities for wireless communication and sensing. In particular, millimeter-wave (mmWave) radars are being actively investigated for their potential to complement the capabilities of future 6G networks. These radars offer high resolution, low latency, and strong resilience to environmental factors, making them ideal for applications in dynamic and challenging environments. As 6G aims to deliver ultra-reliable, low-latency communications, radar systems are poised to enhance situational awareness, object detection, and real-time sensing across various industries, including logistics, autonomous vehicles, and smart cities.

2.3.1 Joint Communication and Sensing (JCAS)

Joint Communication and Sensing (JCAS) is a pivotal concept in 6G networks, combining the functionalities of communication and radar sensing within a unified framework. This approach enables the seamless integration of communication signals with radar waveforms, optimizing spectrum utilization and improving the performance of both systems. Radar and communication applications typically demand more spectrum, power, and hardware resources to boost their performance, leading to potential allocation conflicts. JCAS addresses this challenge by merging both technologies through shared hardware, signal processing schemes, and bandwidth. The integration of Reconfigurable Intelligent Surfaces (RIS) into JCAS systems, referred to as RIS-JCAS, presents both opportunities and challenges, including the need to manage massive unit cells and low phase resolution, which are critical for the advancement of radar systems in the 6G era [32]. The rising issue of spectrum scarcity has made JCAS an area of growing interest. As many radio systems compete for spectrum access, such as telephony, television, radiophony, radiolocation, satellite systems, radio navigation, and radio astronomy, JCAS mitigates this problem by using the same waveform for both communication and sensing. This dual functionality not only optimizes spectrum usage but also reduces energy consumption, making it an environmentally friendly technology [33]. By leveraging mmWave radar sensors in conjunction with communication technologies, JCAS facilitates real-time environmental sensing, which is crucial for applications such as autonomous driving, industrial automation, and smart healthcare systems. Furthermore, JCAS enables the exchange of valuable sensing data while maintaining high-speed communication, offering a more efficient and scalable solution compared to traditional, separate systems.

2.3.2 Impact of 6G on Radar Systems

The exploration of higher frequency bands is expected to be a key trend in the evolution of Sixth Generation (6G) networks. The ongoing development of high-frequency bands has increasingly aligned the characteristics of radar and communication systems, including hardware structure, channel behavior, and signal processing techniques. This convergence has driven the emergence of Integrated Sensing and Communication (ISAC) [34]. The high-frequency spectrum available in 6G, especially in the mmWave and terahertz bands, will enable radar systems to achieve greater bandwidth and resolution, enhancing their ability to detect and track objects in highly dynamic and cluttered environments. This advancement will be particularly beneficial in sectors such as autonomous driving, logistics automation, and healthcare, where real-time decision-making is essential. With the rapid technological advancements in 6G, the sensing capabilities are significantly enhanced due to the expanded spectrum, leading to an exponential increase in the volume and velocity of sensing data. These developments impose stringent requirements on the precision and real-time performance of data processing [35]. Microwave photonics and 6G technologies significantly improve radar performance, delivering high accuracy (measurement errors within ±4 cm), high resolution (25 × 24.7 mm), and the capability to integrate communication and sensing functions seamlessly [36]. As radar technologies continue to evolve, their integration with 6G will unlock new opportunities for creating intelligent, connected environments, thereby enhancing safety, security, and automation across a variety of industries.

2.4 Radar Object Detection Concept

Radar sensors, known for their robust sensing capabilities even in challenging environmental conditions, play a pivotal role in object detection applications. These sensors are often combined with other sensing technologies in a process called **sensor fusion**, significantly enhancing their performance across various domains, including autonomous systems. This synergy enables robust perception and reliable operation under diverse scenarios.

2.4.1 Machine Learning for Object Detection

Machine learning (ML) has become a pivotal component in advancing the capabilities of radar object detection systems, enabling them to handle complex and dynamic environments. In the realm of object detection, machine learning algorithms play an essential role in identifying objects and classifying them accurately using sensor data. These algorithms can adapt and improve their decision-making process as they are trained on more data, allowing them to outperform traditional rule-based systems. While deep learning is a subset of machine learning, machine learning in general serves as the backbone of many modern detection systems. It empowers systems to learn from historical data and recognize patterns that would be challenging to program explicitly. Radar sensor data typically contains significant noise and uncertainty, which could be attributed to environmental disturbances, sensor limitations, or external factors such as weather conditions. Machine learning models, especially those based on deep learning architectures like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), are particularly adept at handling these noisy datasets by extracting meaningful features. Through sensor fusion, where data from multiple sensors, including radar and cameras, is combined, machine learning models improve their accuracy and reliability. This fusion technique is central to modern radar object detection systems, as it enhances the ability to detect objects under challenging circumstances like poor visibility or sensor occlusion [37].

Object detection, as a task in machine learning, generally follows a pipeline comprising three core stages: region selection (localization), feature extraction, and classification. In the region selection stage, the algorithm identifies areas within the sensor data that are likely to contain objects of interest. Feature extraction then analyzes these regions to identify patterns or attributes that can help distinguish objects from the background. Finally, in the classification stage, the algorithm assigns labels to the identified regions, categorizing the objects accordingly. A simple neural network diagram depicting this concept is shown in Figure 2.13.

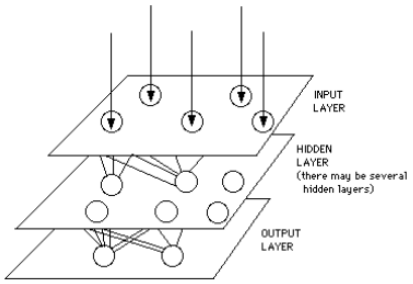


Figure 2.13: A Simple Neural Network Diagram [38]

Machine learning algorithms for object detection are classified into two main categories: one-stage detectors and two-stage detectors. One-stage detectors directly perform detection on a

dense grid of potential object locations, skipping the region proposal step. Examples of one-stage detectors include RetinaNet [37], YOLO (You Only Look Once) [39], and SSD (Single Shot MultiBox Detector) [40]. These methods are faster, making them suitable for real-time applications, though they may sometimes sacrifice accuracy in detecting smaller or overlapping objects. On the other hand, two-stage detectors, such as Faster R-CNN [41] and R-FCN (Region-based Fully Convolutional Networks) [42], first propose regions of interest and then classify them, providing higher detection accuracy at the cost of slower processing. The concept of a simple network with feedback and competition is illustrated in Figure 2.14.

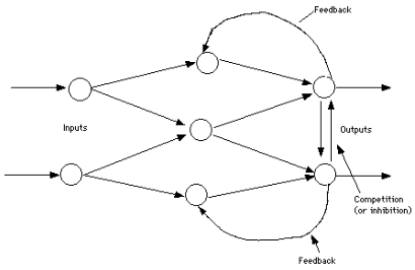


Figure 2.14: Simple Network with Feedback and Competition [38]

The accuracy of these detection algorithms is typically measured using a metric known as mean average precision (Mean Average Precision (mAP)). mAP combines both precision and recall, calculating the average precision across all classes in the dataset. Precision refers to the fraction of true positive detections among all the detections made by the algorithm, while recall represents the fraction of true positive detections among all the actual objects present in the dataset. A detection is considered a true positive if the predicted bounding box overlaps sufficiently with the ground truth, usually defined by an Intersection over Union (IoU) greater than 0.5. This metric is essential for evaluating the effectiveness of object detection systems, ensuring they provide both accurate and reliable results [43]. Furthermore, the concepts of supervised and unsupervised learning are fundamental to machine learning. In this study, the focus is primarily on supervised learning, where the model is trained on labeled data. The differences between these learning paradigms are visualized in Figure 2.15.

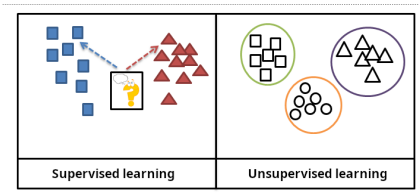


Figure 2.15: Supervised and Unsupervised Learning [44]

Machine learning-based object detection is transforming radar systems, enabling them to be more intelligent and robust in dynamic environments. With advancements in sensor fusion and deep learning, these systems can offer improved performance across various domains, such as autonomous driving, industrial applications, and surveillance, among others.

2.4.2 Sensor Fusion for Object Detection

Sensor fusion integrates data from multiple sensors to improve detection and perception capabilities. This approach has proven highly effective in object detection applications. Commonly

used sensors in fusion systems include RADAR, LIDAR, and cameras, each bringing unique strengths to the table while mitigating individual weaknesses. For example, RADAR excels in determining distances and velocities but is less precise in angle measurements. Conversely, cameras offer high-resolution object recognition and classification but may struggle in low-light or adverse weather conditions [45]. The combination of these sensors ensures robust object detection in autonomous systems. RADAR operates by emitting electromagnetic waves, receiving the echoes reflected from objects, and analyzing these echoes to estimate parameters such as distance, velocity, and angle. In contrast, cameras capture detailed visual data in the form of high-resolution images or videos, enabling accurate object recognition and classification. By combining data from RADAR and cameras, sensor fusion systems achieve enhanced detection accuracy and provide a more comprehensive understanding of the environment [46]. The workflow of sensor fusion, integrating data from RADAR and cameras, is illustrated in Figure 2.16. In autonomous driving scenarios, radar-camera fusion systems provide valuable depth information for all detected objects, eliminating the need for computationally expensive 3D object detection using LIDAR point clouds [47].

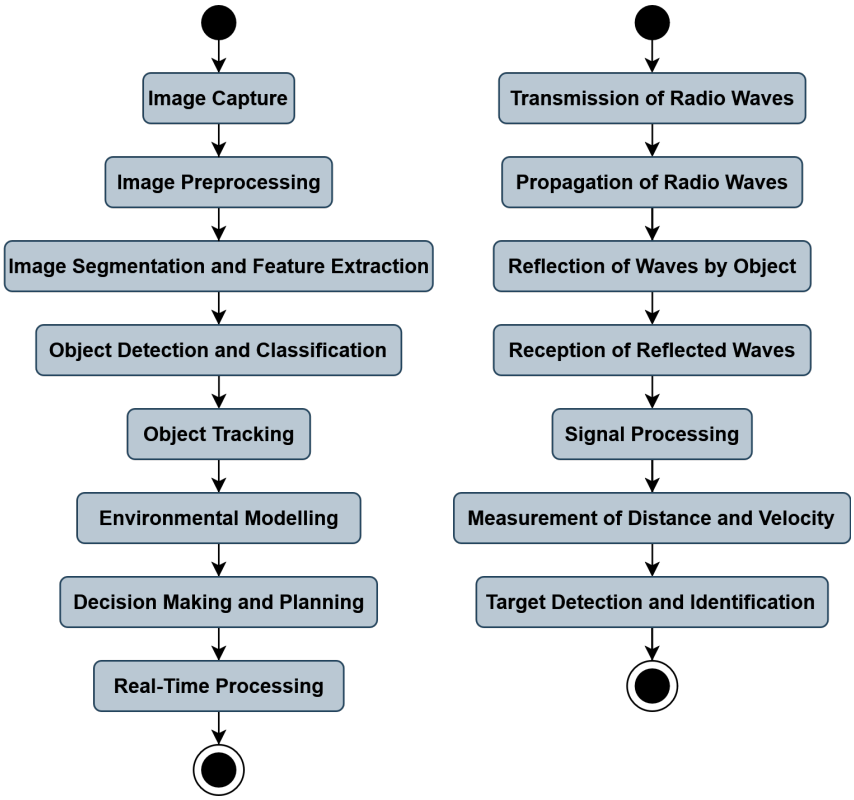


Figure 2.16: Workflow of Camera and Radar Sensor Fusion [37].

Sensor fusion technology leverages the complementary strengths of RADAR and cameras to address the limitations of individual sensors, resulting in improved accuracy and robustness in environmental perception. Data-level fusion combines raw data from multiple sensors to retain maximum information, thereby enhancing perception accuracy. Feature-level fusion extracts and integrates features from sensor data, reducing data volume and improving computational efficiency. Decision-level fusion processes data independently for each sensor, generating decisions that are subsequently combined. Feature-level fusion enhances the system's perception capability by exploiting feature complementarity, while decision-level fusion aggregates independent sensor decisions into a unified output, further improving overall performance [37].

3 State of the Art

The advent of 6G is poised to revolutionize sensing technologies, enabling precise radar object detection through its ultra-low latency, terahertz communication, and joint communication-sensing capabilities[48]. In logistic industries, precise object detection and real-time data processing are crucial as increasing adoption of autonomous systems and smart warehouse. Radar sensing, a key technology in this domain, stands to benefit significantly from 6G advancements. The sensing capabilities of 6G are substantially enhanced due to the expansion of the spectrum, leading to an explosive growth in sensing data volume[49]. This unprecedented increase in data generation imposes extremely stringent requirements on the precision and real-time performance of data processing[50]. The integration of 6G technologies is expected to provide a seamless, high-capacity network that can support advanced sensing applications in logistics environments[51].

3.1 Intralogistics: Current Sensing and Localization Technologies

Currently in autonomous logistic environment, various methods and techniques are used such automated guide vehicle(AGV) and inventory tracking, etc. This increasing adaption of AI, machine learning, and computer vision has further improving handling of the goods in the warehouse. AI-driven algorithms optimize space utilization, predict demand, and automate stock reordering, while machine learning models analyze data to detect patterns and forecast future trends[52]. Real-time people counting using video footage and computer vision techniques, primarily utilizing libraries like OpenCV, has been implemented in practical applications[53]. Radar systems are widely recognized for their ability to detect objects in diverse environmental conditions, making them invaluable in logistics applications such as automated guided vehicles (AGVs)[54], robotic sorting systems, and real-time inventory tracking[55]. Unlike alternative sensing technologies like LiDAR and cameras, radar is inherently robust to environmental factors such as dust, fog, and low-light conditions[56]. Furthermore, radar systems provide additional data, such as object velocity, which is critical for dynamic environments. For instance, FMCW (Frequency-Modulated Continuous Wave) radar is widely used for tracking moving objects[57].

3.2 The Role of Radar

Radar systems have traditionally been paired with other sensors like cameras and LiDAR in sensor fusion setups to overcome the individual weaknesses of each sensor[58]. While combining data from multiple sensors can enhance object detection by providing a fuller picture, but it also introduces added complexity, higher costs, and increased power consumption[59]. In contrast, modern radar systems, especially those that integrate advanced signal processing and machine learning (ML) techniques, are now capable of performing effective object detection without the need for additional sensors. Recent advances, such as the development of frequency-modulated continuous wave (FMCW) radar[57], have improved radar's range and resolution, making it more accurate in tracking objects. These improvements mean radar systems can operate independently in more complex logistics environments, such as detecting numerous objects in busy, cluttered spaces. Machine learning has played a crucial role in this transformation. Previously, radar systems relied on manual feature extraction and heuristic methods, which often struggled with noise and ambiguity in radar signals[60]. However, ML algorithms can now automate and refine these processes, enabling radar systems to analyze data more effectively and adapt to real-world challenges[61].

Millimeter-wave (mmWave) radar is an emerging technology that offers significant advantages for a variety of sensing applications. One of its primary benefits is its high resolution, speed and accuracy, which is made possible by its short wavelength. This allows mmWave radar to detect and track objects with greater precision, even in environments that are cluttered or complex[62][16]. In addition to its resolution, mmWave radar is highly resistant to difficult environmental conditions. Unlike optical sensors, which struggle in poor weather or low-light situations, mmWave radar continues to perform well in fog, rain, snow, or complete darkness, making it ideal for 24/7 operation[63]. This capability is particularly useful in outdoor logistics and transportation, where reliable sensing is critical regardless of environmental factors. Another major advantage of mmWave radar is its non-intrusive nature. It can detect presence, movement, and even vital signs without capturing personally identifiable information, addressing privacy concerns that can arise with other sensor types[64]. Furthermore, the compact design of mmWave radar systems—thanks to the high frequency of the technology—allows them to be integrated into smaller devices, vehicles, and robots[62]. This makes it easier to deploy radar systems in a wide range of settings. Finally, mmWave radar is capable of simultaneously tracking multiple objects, providing real-time data on their position, velocity, and trajectory—information that is essential for efficient and dynamic logistics operations[65].

3.3 State-of-the-Art Radar Applications

The evolution of mmWave radar sensor research is expected to have a significant impact on logistics automation. These sensors offer the potential for more accurate and reliable object detection and tracking in warehouse environments, leading to improved efficiency and safety in autonomous operations. Example applications include automated guided vehicles (AGVs) and warehouse inventory systems, where cost and simplicity favor standalone radar solutions [54]. The integration of 6G technologies with advanced radar sensing is anticipated to enable real-time, high-precision tracking of goods and equipment, optimizing warehouse operations and supply chain management [66]. As research in this field progresses, the development of more sophisticated, AI-driven radar systems is expected. These systems can adapt to dynamic logis-

tics environments, potentially revolutionizing the industry's approach to automation and inventory management [67].

The unique capabilities of mm-Wave radar make it particularly valuable in various logistics applications. One of its key applications is in warehouse management, where mm-Wave radar can be used for inventory tracking, monitoring storage space utilization, and detecting unauthorized access to restricted areas [68]. In warehouse and factory settings, mm-Wave radar plays a crucial role in enabling Automated Guided Vehicles (AGVs) to navigate safely, avoid obstacles, and interact with their environment more effectively [69]. Additionally, radar sensors can assist in precise positioning of trucks at loading docks, enhancing safety and efficiency in loading and unloading operations [70]. Another important application is asset tracking, where mm-Wave radar can be employed to track high-value assets throughout the supply chain, providing real-time location and movement data [71]. The technology's ability to penetrate non-metallic materials allows for tracking even when assets are not in direct line of sight, which is particularly useful in warehouses and distribution centers. Moreover, in logistics facilities, radar sensors can monitor occupancy levels in different areas, optimizing space utilization and improving energy efficiency [72]. The ability of mm-Wave radar to detect presence and movement without compromising privacy makes it an excellent choice for monitoring occupancy in various zones of a logistics facility.

3.4 Machine Learning Models for Object Detection

Recent advancements in deep learning have significantly improved the performance of object detection, particularly through the use of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Object detection typically involves three key steps: region selection (localization), feature extraction, and classification. These algorithms are categorized into one-stage and two-stage detectors. One-stage detectors, such as RetinaNet [73], SSD [74], and YOLO [75, 76], perform detection directly over all possible foreground and background positions, skipping the region proposal step. On the other hand, two-stage detectors like Faster R-CNN [77] and R-FCN [78] first select regions of interest, followed by classification of positive region candidates. The performance of these algorithms is evaluated using mean average precision (mAP), which combines precision and recall, and is based on the Intersection over Union (IoU) of the predicted bounding box and the ground truth [79]. These advancements have significantly enhanced object detection accuracy, especially in sensor fusion applications.

The rapid development of state-of-the-art object detection technologies has paved the way for their integration into automated intralogistics industries, particularly with the advent of 6G sensing. In these applications, the efficiency and accuracy of real-time object detection systems play a pivotal role in enabling seamless operation. Among various detectors, the YOLOv7 model demonstrates a superior balance of speed and accuracy. On the MS COCO dataset, YOLOv7 achieves a 120% improvement in speed while maintaining an average precision (AP) of 56.8%, outperforming other real-time detectors [18]. Figure 3.1 illustrates the comparison of average precision and inference time for different real-time object detectors on an NVIDIA V100 GPU, while Table 3.1 provides a detailed breakdown of YOLOv7 variants.

The performance metrics of YOLOv7 indicate a significant reduction in parameters (up to 40%) and computational cost (up to 20%), along with improved accuracy, making it an ideal candidate for applications requiring robust, high-speed detection. Table 3.1 summarizes the performance

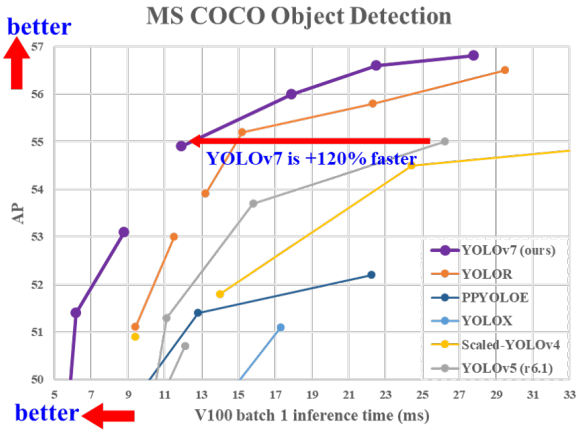


Figure 3.1: Comparison of AP vs. inference time on V100 GPU for various real-time object detectors. YOLOv7 achieves state-of-the-art performance [18].

of various YOLOv7 variants, highlighting their parameter efficiency and computational superiority [18].

Model	Test Size	AP	AP50	AP75	Batch 1 FPS
YOLOv7	640	51.4%	69.7%	55.9%	161
YOLOv7-X	640	53.1%	71.2%	57.8%	114
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36

Table 3.1: Performance comparison of YOLOv7 variants on MS COCO dataset [18].

Two-stage detectors like Faster R-CNN also continue to serve as benchmarks for performance. As shown in Table 3.2, Faster R-CNN achieves consistent accuracy across various configurations, although at higher computational cost compared to one-stage detectors [19]. Similarly, RetinaNet maintains competitive inference times while achieving reliable accuracy for many applications, as summarized in Table 3.3.

Table 3.2: COCO Object Detection Baselines for Faster R-CNN [19].

Model	Learning Rate Schedule	Training Time (s/iter)	Inference Time (s/im)	Training Memory (GB)
R50-C4	1x	0.551	0.102	4.8
R50-DC5	1x	0.380	0.068	5.0
R50-FPN	1x	0.210	0.038	3.0
R50-C4	3x	0.543	0.104	4.8
R50-DC5	3x	0.378	0.070	5.0
R50-FPN	3x	0.209	0.038	3.0

Table 3.3: COCO Object Detection Baselines for RetinaNet [19].

Model	Learning Rate Schedule	Training Time (s/iter)	Inference Time (s/im)	Training Memory (GB)
R50	1x	0.205	0.041	4.1
R50	3x	0.205	0.041	4.1
R101	3x	0.291	0.054	5.2

Datasets such as the CRUW [80], RaDICaL [81], CARRADA [82], MMVR [83], and MiliPoint [84] showcase the use of 6G sensing technologies, particularly mmWave radar sensors, for

object detection. Although these datasets primarily focus on autonomous driving and human activity recognition, they provide valuable insights that inform the creation of a 6G sensing dataset tailored for object detection applications in automated logistics. By leveraging similar radar sensing technologies, a dataset can be developed to meet the specific needs of logistics applications such as warehouse operations and supply chain management.

Despite the advancements in radar-based object detection and sensor fusion technologies, there remain several challenges in the logistics industry that require further exploration. Current systems often rely on sensor fusion, combining radar, LiDAR, and cameras, which increases system complexity, cost, and power consumption. Furthermore, while these systems work well in many controlled environments, they still struggle in dynamic and cluttered logistics scenarios where occlusion and environmental interference can degrade performance. There is also a lack of standardized datasets specific to indoor logistics applications with 6G sensing, which limits the development of robust machine learning models tailored for these environments. This research addresses these gaps by focusing on the potential of mmWave radar sensors integrated with machine learning techniques for standalone object detection. Unlike traditional approaches, mmWave radar offers high resilience to environmental changes such as dust, fog, and low-light conditions, while also providing the advantage of non-intrusive sensing. The combination of mmWave radar with emerging 6G technologies, which can support high data transfer rates and low-latency communication, further enhances its applicability in real-time logistics operations. By developing a machine learning-based object detection system using mmWave radar, this study aims to bridge these gaps, offering a more efficient, accurate, and reliable solution for logistics automation. This approach not only addresses the limitations of conventional systems but also paves the way for the future of intelligent and autonomous logistics solutions.

4 Methodology

This chapter provides a detailed explanation of the methodology employed for model selection, dataset preparation, data preprocessing, and the design of the input pipeline for machine learning-based object detection models. Building on the discussion in the Chapter 3, which explores various types of neural network backbones, this chapter focuses on the design pipeline and implementation steps as part of the methodology. The selection of appropriate models aligns with the given architectures and is tailored for custom datasets. To achieve optimal results, understanding the architecture of the selected models and their compatibility with custom datasets is essential. Additionally, since custom data is used, ensuring that the data format is supported and that the input pipeline is properly configured for these models is crucial. This process involves evaluating model-specific requirements, such as input size, annotation formats, and preprocessing steps, to ensure seamless integration and effective training. By following this approach, a robust framework is established, allowing for the efficient use of state-of-the-art object detection models tailored to the specific requirements of the task at hand.

4.1 Model Selection

Model selection is a critical step in developing effective radar based object detection systems for intralogistics environments. Various models have been published for tasks such as object detection, image segmentation, classification, keypoint detection, and depth estimation. It is essential to evaluate these models based on their architecture, performance, and suitability for the specific application requirements.

In indoor logistics environments, robust object detection capabilities are crucial. The system must handle large, dense datasets and provide precise spatial information between objects [85]. Two prominent frameworks that address these needs are YOLO (You Only Look Once) and Detectron2. YOLO is a real-time object detection system that predicts bounding boxes and class probabilities directly from images in a single evaluation. It is renowned for its speed and accuracy in simultaneously detecting multiple objects [86]. YOLO's architecture has evolved through several versions, including YOLOv4, Scaled-YOLOv4, and YOLOv7, each improving performance, efficiency, and accuracy [87, 88, 18]. These models are designed for easy integration, with pre-trained weights and straightforward APIs available in popular frameworks like PyTorch and TensorFlow. Detectron2, developed by Facebook AI, is a modular framework for object detection and segmentation that offers high accuracy and flexibility for various vision tasks [89]. It supports advanced techniques such as Faster R-CNN and RetinaNet, making it suitable for both research and production environments. Detectron2's continuous evolution incorporates the latest breakthroughs in computer vision, ensuring state-of-the-art results for object detection and segmentation tasks.

For the intralogistics application, both YOLO and Detectron2 models are employed. YOLO utilizes a neural network backbone based on Convolutional Neural Networks (CNNs), while Detectron2 leverages advanced techniques such as Faster R-CNN [90] and RetinaNet [91]. By applying these diverse approaches, a comparative analysis is conducted to determine the most suitable technique for object detection in the specific environment. This analysis evaluates the models' performance in terms of accuracy, speed, and robustness, considering the unique challenges posed by intralogistics settings. By comparing these state-of-the-art object detection frameworks, an informed decision can be made regarding the most appropriate model for the application, ensuring optimal performance in detecting and localizing objects within the complex and dynamic environment of indoor logistics operations.

4.1.1 Yolov7 for Object Detection

YOLOv7 represents a significant advancement in object detection, surpassing all known real-time object detectors in both speed and accuracy within the range of 5 to 120 FPS. Specifically, YOLOv7 achieves the highest accuracy of 56.8% AP among all real-time object detectors operating at 30 FPS or higher on the NVIDIA V100 GPU. The source code for YOLOv7 is publicly available at `https://github.com/WongKinYiu/yolov7`. Unlike conventional real-time object detectors, YOLOv7's development focuses on optimizing the training process in addition to architectural enhancements. It incorporates specialized modules and optimization methods that may increase the training cost to improve detection accuracy without affecting inference efficiency [18]. Figure 4.1 depicts the architecture of the YOLOv7 model, showcasing the sequence of layers including convolutional layers and max pooling operations. It starts with an input layer typically sized at 640×640 pixels, followed by multiple convolutional and pooling layers that progressively reduce spatial dimensions while increasing feature depth, essential for efficient object detection and feature extraction. The visualization helps in understanding the flow and transformation of data through the network, highlighting the model's capability to handle complex visual tasks.

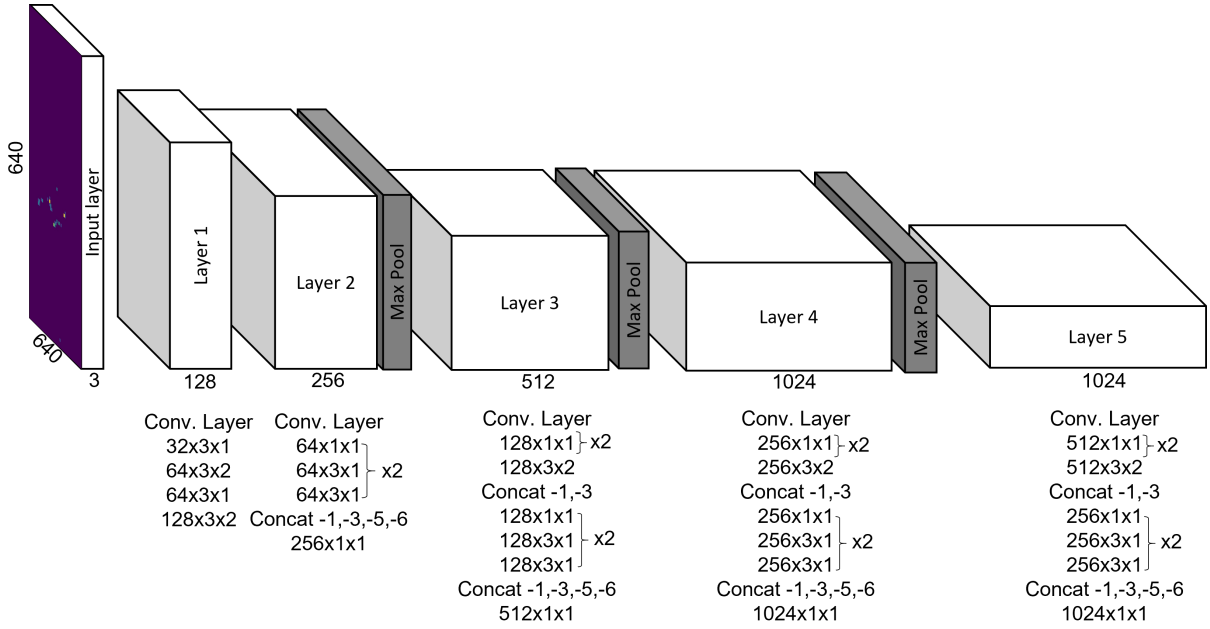


Figure 4.1: Architecture of the YOLOv7 model

E-ELAN Architecture: The Extended Efficient Layer Aggregation Networks (Enhanced Efficient Layer Aggregation Network (E-ELAN)) architecture is a key innovation in YOLOv7. E-ELAN employs operations such as expansion, shuffling, and merge cardinality to enhance the network's learning capacity while preserving gradient flow. Structurally, E-ELAN modifies only the computational blocks while leaving the transition layers intact. The main strategy involves using group convolution to expand the channels and cardinality within computational blocks. A consistent group parameter and channel multiplier are applied across all computational blocks in a layer.

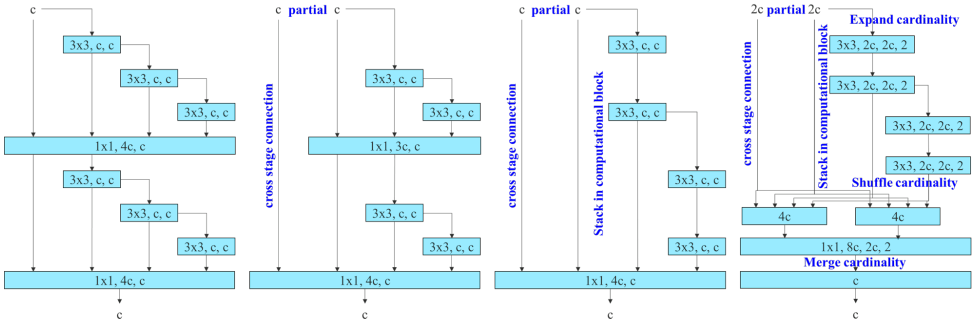


Figure 4.2: Extended Efficient Layer Aggregation Networks (E-ELAN) [18].

The process includes shuffling feature maps into g groups based on the group parameter g, followed by concatenating the shuffled feature maps. This ensures that the number of channels in each group remains consistent with the original architecture. Finally, merge cardinality is applied by adding g groups of feature maps. By retaining the original ELAN design and diversifying the features learned by different computational blocks, E-ELAN improves parameter utilization and computation efficiency. The main architecture of E-ELAN is illustrated in Figure 4.2 [18].

4.1.2 Detectron2 Framework

Detectron2 is an advanced modular object detection framework developed by Facebook AI, designed to improve upon its predecessor, Detectron. It offers a more modular and extensible design, supporting dynamic computational graphs that are compatible with modern machine learning frameworks like PyTorch[19]. Enhanced performance, better hardware utilization, and simplified model customization make Detectron2 a preferred choice for a wide range of vision tasks. The framework supports tasks such as object detection, instance segmentation, keypoint detection, and panoptic segmentation. Among these, object detection is a primary focus of this thesis. Detectron2 integrates state-of-the-art algorithms like Faster R-CNN[90] and RetinaNet[91], which offer robust detection capabilities suitable for various applications. Faster R-CNN, a two-stage detector, generates region proposals and classifies objects within these regions, balancing speed and accuracy. On the other hand, RetinaNet, a single-stage detector, employs the focal loss to address class imbalance, excelling in detecting small and densely packed objects.

Detectron2's architecture emphasizes modularity, facilitating experimentation and deployment. It incorporates several key components:

1. **Data Handling:** A flexible data pipeline processes datasets in Common Objects in Context (COCO) or custom formats, seamlessly integrating with various sources.
2. **Backbone Networks:** ResNet and other architectures serve as the backbone for feature extraction.
3. **Feature Pyramid Network (Feature Pyramid Network (FPN)):** Enhances multi-scale feature representation, crucial for detecting objects of varying sizes.
4. **Region Proposal Network (Region Proposal Network (RPN)):** Efficiently generates high-quality region proposals.
5. **ROI Heads:** Specialized heads perform bounding box regression, classification, and mask prediction.
6. **Configurable Training and Inference:** Users can customize hyperparameters, augmentations, and evaluation metrics through a simple configuration system.

This modular design allows researchers to innovate and adapt models to specific needs. The COCO object detection baselines for Faster R-CNN and RetinaNet, summarized in the following two tables, highlight the framework's effectiveness in achieving competitive speed-accuracy trade-offs. Detectron2 incorporates a highly flexible and efficient architecture that supports various computer vision tasks. One of its key components is the RetinaNet architecture, which excels in detecting objects at different scales. RetinaNet employs a single-stage detector that utilizes a feature pyramid network (FPN) to extract multi-scale features. It leverages the focal loss to address the issue of class imbalance, especially in tasks where foreground objects are significantly fewer than background objects. The combination of these design choices makes RetinaNet particularly effective in detecting small and densely packed objects, ensuring both high accuracy and speed. A visual representation of the RetinaNet architecture is shown in Figure 4.3, which highlights the key components and flow of information within the model.

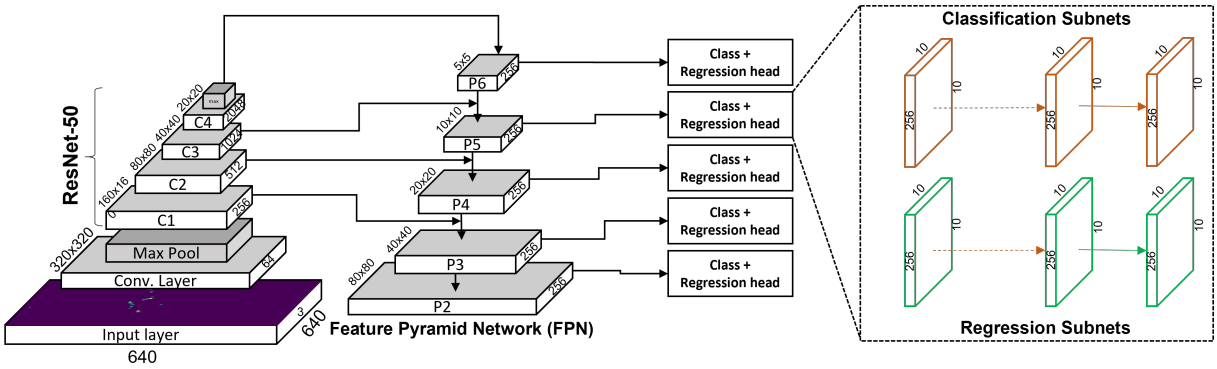


Figure 4.3: RetinaNet Architecture.

Faster R-CNN, another core architecture in Detectron2, is composed of two main modules that work together as a unified object detection network. The first module is a deep fully convolutional network called the Region Proposal Network (RPN), which generates region proposals from the input image. These regions are then passed to the second module, the Fast R-CNN detector, which performs object classification and bounding box regression using the proposed regions. The two modules are tightly integrated, forming a single, end-to-end trainable system for efficient object detection. Additionally, using concepts from neural networks with 'attention' mechanisms, the RPN serves as the 'attention' module, guiding the Fast R-CNN detector

to focus on the most relevant parts of the image for detection. This unified architecture enhances the model's performance by reducing redundant computations and improving accuracy. Figure 4.4 illustrates the complete Faster R-CNN architecture, demonstrating the interaction between the RPN and Fast R-CNN modules.

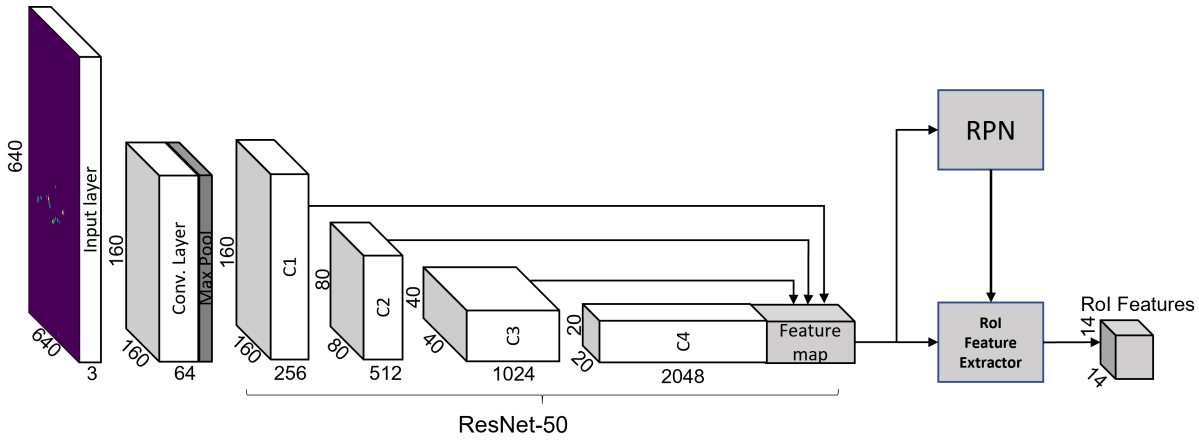


Figure 4.4: Faster R-CNN detailed architecture.

4.2 Design Pipeline

The aim of this experiment is to design a process for object detection in both offline and online environments. The offline process consists of six distinct steps: data collection, data preprocessing, data annotation, model training, model evaluation, and object detection output. After integrating the mmWave radar, a dataset is collected by simulating real-time logistic scenarios. This step is referred to as data collection. To ensure accurate model training, the dataset must be clean so that the model can effectively extract features, thereby reducing its complexity during object detection. Hence, the collected dataset undergoes preprocessing before being provided to the model. This step is referred to as data preprocessing. Since a supervised machine learning approach is employed, the dataset must be appropriately labeled. Therefore, the preprocessed dataset is annotated, which constitutes the data annotation step. After successfully preparing the dataset, the selected machine learning model is trained on the labeled dataset, which corresponds to the model training step. To evaluate the performance of the trained model, it is assessed using performance metrics. This constitutes the model evaluation step. Finally, the trained model generates object detection outputs, which is the concluding step of the pipeline, referred to as object detection output. Figure 4.5 illustrates the design pipeline for the offline model testing experiment.

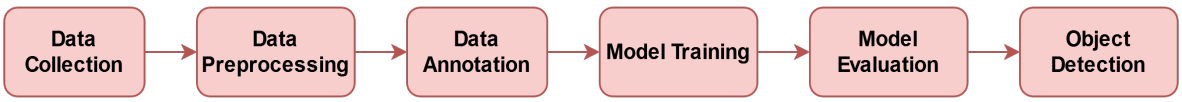


Figure 4.5: Design pipeline for the offline model testing experiment.

In the online testing process, the trained model is evaluated for robustness and compatibility with new, unseen data. The online model testing process consists of the following steps: real-time data collection, data preprocessing, trained model, object detection, and performance monitoring. Using the integrated sensor, data is continuously collected in real-time, constituting the

real-time data collection step. This real-time dataset is continuously preprocessed and passed to the trained model, covering the data preprocessing step. The trained model processes the preprocessed dataset and, using its weights and learned parameters, generates object detection outputs, corresponding to the trained model and object detection steps. Finally, the performance of the detection is continuously monitored, constituting the performance monitoring step. Figure 4.6 illustrates the design pipeline for the online model testing experiment.

Figure 4.6: Design pipeline for the online model testing experiment.

4.3 Dataset Preparation

Dataset preparation is a fundamental step in machine learning, particularly in supervised learning. It involves collecting, organizing, and annotating data to teach models how to make accurate predictions or decisions. This process is analogous to teaching a child; the quality and accuracy of what they learn determine their future performance [92]. Similarly, natural language processing (Natural Language Processing (NLP)) and other machine learning models heavily depend on high-quality datasets. Currently, there is a wealth of publicly available datasets on platforms like Roboflow [93] and Kaggle, which provide extensive resources for training various models.

4.3.1 Data Annotation

Data annotation is the process of labeling data to make it usable for machine learning models. The format and accuracy of these labels are crucial, as they directly influence the performance of the model being developed. There are numerous open-source software tools available for dataset labeling, enabling efficient and precise annotation. Additionally, several collaborative platforms allow multiple users to contribute to data labeling tasks, enhancing efficiency and scalability[93]. The datasets used to train and evaluate object detection models require substantial human effort for accurate labeling, which is often prone to imperfections. Annotators are tasked with inspecting each image and, for every object depicted, drawing bounding boxes around it and assigning a discrete class label for categorization [94].

While semi-automated label generation tools are available to accelerate the annotation process, these tools often require manual fine-tuning to ensure the labels meet the necessary quality standards [95]. The combination of automation and human oversight provides a robust approach to generating high-quality annotated datasets.

- **Bounding Box Annotation**: Annotators draw boxes around objects and assign labels for classification.
- **Semantic Segmentation**: Each pixel in the image is labeled to indicate the object it belongs to.

- **Collaboration Tools**: Platforms that enable community-driven or team-based annotation for large datasets.
- **Semi-Automated Annotation**: Tools that leverage AI to assist annotators, reducing time but requiring human verification for accuracy.

High-quality annotation ensures the dataset is well-suited for training machine learning models, minimizing errors during model evaluation and deployment.

4.3.2 Data Augmentation

Data augmentation is a crucial technique in machine learning to enhance the diversity and size of datasets. The primary goals of a data augmentation system include:

1. Supporting the augmentation of multiple data types, such as images with their associated bounding boxes and masks.
2. Allowing the application of a sequence of pre-defined augmentation operations.
3. Enabling custom augmentation for new data types, such as rotated bounding boxes, video clips, etc.
4. Providing flexibility to process and modify operations applied during augmentation [19].

Data augmentation techniques effectively expand the dataset, reduce the sensitivity of detection models, and improve their overall performance [96]. For instance, in a study involving indoor object detection using YOLOv5, data augmentation contributed to achieving a high average accuracy of 93.9% at an intersection over union of 0.9 [97].

4.3.3 Data Splitting

Proper data splitting is essential in machine learning to ensure models generalize well to unseen data and avoid overfitting or underfitting. Splitting the dataset into subsets for training and testing ensures that the model is trained on one subset and evaluated on another. Commonly, 70–80% of the data is used for training, and the remaining 20–30% is reserved for testing [98, 99]. Several methods exist for data splitting, such as cross-validation, bootstrapping, and systematic sampling. These techniques aim to provide an unbiased evaluation of the model's performance [100]. Optimal data splitting methods, like the one proposed by Joseph and Vakayil, further enhance the reliability of the generalization performance of machine learning models [101].

4.4 Model Training

Model training is the process of teaching a machine learning model to make predictions or decisions by optimizing its parameters using a labeled dataset. This involves feeding the model input data and adjusting its internal weights to minimize the difference between its predictions

and the actual target values, a difference quantified by the **loss**. Common parameters for model training include the learning rate, batch size, number of epochs, optimizer type, and regularization terms. The **loss** represents the model's error during training. It is computed using a predefined loss function that measures how far the predicted outputs deviate from the true outputs. A **loss curve** visualizes the loss over training epochs, providing insight into the model's performance during training. For a well-trained model, the training loss should gradually decrease and stabilize, while the validation loss should also decrease and remain low, without significant divergence. A substantial gap between the training and validation loss might indicate overfitting[102].. Machine learning-based recognition methods, such as Convolutional Neural Networks (CNNs) or similar architectures, are widely used for tasks like character recognition. These methods directly learn to classify and recognize input samples. The effectiveness of this approach depends on factors such as the size of the dataset and the number of training iterations. Optimizing such models aims to enhance recognition performance while reducing training time [103].

4.4.1 Training Yolov7

After preparing the dataset, configuring the dataset files, and setting up the model parameters, the training process for YOLOv7 can begin. Training commands are designed for both single and multi-GPU setups, enabling flexibility in resource utilization.

For **single GPU training**, two variations are supported: training smaller P5 models and larger P6 models. P5 models use an image resolution of 640×640 with a batch size of 32, while P6 models employ a higher resolution of 1280×1280 and a smaller batch size of 16 due to increased computational demands. The `train.py` script is used for P5 model training, and the `train_aux.py` script handles P6 models. Both configurations are initiated with the command-line argument `--device 0` to specify the GPU and `--weights ''` to start training from scratch. Model architecture and hyperparameters are defined in configuration files passed using the `--cfg` and `--hyp` options [18, 104].

For **multi-GPU training**, the commands leverage PyTorch's distributed training framework. P5 models are trained using 4 GPUs (`--nproc_per_node 4, --device 0,1,2,3`) with a batch size of 128, utilizing synchronized batch normalization (`--sync-bn`). P6 models, being more computationally intensive, are trained using 8 GPUs (`--nproc_per_node 8, --device 0,1,2,3,4,5,6,7`), maintaining the same batch size of 128 distributed across GPUs. Proper communication between GPUs is ensured using the `--master_port` parameter. Both setups use the same dataset (`data/coco.yaml`) but adapt the architecture and hyperparameters to suit the model type. This setup maximizes hardware utilization, expediting training for large-scale tasks [18, 104].

4.4.2 Training Detectron2 Models

In Detectron2, Dataloader is the component that provides data to models. A dataloader usually (but not necessarily) takes raw information from datasets, and process them into a format needed by the model. After defining the model and preparing the data loader, training logic can be implemented using PyTorch. This allows researchers to customize the entire training loop for full control. An example training loop is provided in the script `tools/plain_train_net.py`,

where users can directly modify the logic to suit their needs [19].

Detectron2 provides a trainer abstraction with a hook system to simplify training workflows. Two key trainers include[19]:

- **SimpleTrainer**: A minimal training loop designed for single-cost, single-optimizer, and single-data-source training. Other features like checkpointing and logging can be added using the hook system.
- **DefaultTrainer**: Built upon SimpleTrainer, this is initialized using a Yet Another Configuration System (YACS) configuration and is commonly used in scripts like `tools/train_net.py`. It includes standard features such as default optimizer configurations, learning rate schedules, logging, evaluation, and checkpointing.

4.5 Sensor Integration

The integration of the mmWave radar sensor involves both hardware setup and sensor configuration to ensure seamless operation and optimal performance. The `IWR6843ISK` mmWave radar sensor, along with the `MMWAVEICBOOST` evaluation board, serves as the core of the system. This combination facilitates radar data acquisition, processing, and communication with external devices. To support the operation of the system, specific hardware configurations and sensor settings must be applied, which are further refined using software tools such as the `mmWave Visualizer` and other relevant code repositories.

4.5.1 Flashing the IWR6843ISK Evaluation Module

Before connecting the IWR6843ISK to the MMWAVEICBOOST module, the sensor must be flashed with the default firmware provided by Texas Instruments (TI). To perform this operation, you will need the IWR6843ISK mmWave radar sensor, a micro-USB cable, and a computer. Start by configuring the Evaluation Module (Embedded Visual Model (EVM)) into Flashing Mode by setting the S1 switch as specified in Table 4.1. Connect the micro-USB cable to the EVM at the designated connector and then to a PC. Note that a separate power supply is unnecessary as the IWR6843ISK receives power through the USB connection. Upon connecting, the LEDs on the EVM should illuminate. To ensure the correct SOP mode is latched after applying power, cycle the power and press and release the RST switch. The green LED near the S3 switch should toggle OFF and ON when the switch is depressed and then released. For additional details, refer to the hardware setup documentation provided by TI[105].

Table 4.1: S1 Switch Configuration for Flash Mode

Mode	S1.1	S1.2	S1.3	S1.4	S1.5	S1.6
Flashing Mode	On	Off	On	On	Off	N/A

After configuring the EVM, install the UniFlash tool on a Windows OS system. Launch UniFlash and, in the ”New Configuration” section, select the device corresponding to your EVM type, then click ”Start” to proceed. Navigate to the ”Settings & Utilities” tab and, under ”Setup,”

enter the COM port number associated with the CFG_Port of your EVM. Switch to the ”Program” tab, click ”Browse” for Meta Image 1, and select the appropriate image file (.bin) to be flashed. After selecting the .bin file, toggle the NRST switch on the EVM to power cycle the device. Once the file is loaded, the ”Load Image” button will become active. Click ”Load Image” to flash the firmware onto the device. The flashing process is now complete. For further guidance, refer to the TI software setup documentation [106].

4.5.2 Hardware Setup

The integration of the mmWave radar sensor system begins with configuring and assembling the hardware. The setup requires the IWR6843ISK module and the MMWAVEICBOOST board, which facilitate communication and operation of the radar system. The MMWAVEICBOOST board is stacked with the IWR6843ISK module using two 60-pin high-density connectors. To ensure mechanical stability and improve thermal performance, the assembly is secured with 12 nuts, 4 washers, and 4 M3 screws, as shown in Figure 4.7.

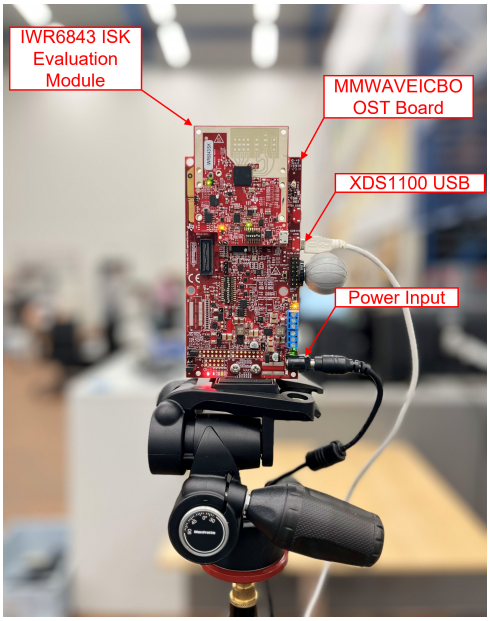


Figure 4.7: Assembly of the IWR6843ISK module and MMWAVEICBOOST board.

The system is powered through a 5-V DC power jack with a 2-A current limit. For flashing the radar module with application binaries, connect only the XDS110 USB cable to the PC. Using both XDS110 USB and FTDI USB simultaneously can interfere with the flashing process. The flashing is performed using the Uniflash utility, and it is recommended to install the latest XDS110 and FTDI drivers from the mmWave SDK package [107]. To operate the system in MMWAVEICBOOST mode, proper configuration of the S1 switch on the IWR6843ISK module is essential. Table 4.2 provides the required switch settings for this mode. In this configuration, UART communication is routed through the 60-pin connector to the XDS110 USB, enabling data transfer between the radar chip and the host system.

For advanced configurations and detailed data capture, the system can be interfaced with mmWave Studio. This tool, available online at https://www.ti.com/tool/MMWAVE-STUDIO, supports additional debugging and optimization of radar parameters [108]. This hardware setup

Table 4.2: S1 Switch Configuration for MMWAVEICBOOST Mode

Mode	S1.1	S1.2	S1.3	S1.4	S1.5	S1.6
MMWAVEICBOOST Mode	Off	Off	Off	Off	On	-

ensures a stable and functional system, enabling accurate data acquisition and processing for radar-based object detection applications.

4.5.3 Sensor Configuration

The mmWave radar sensor, `IWR6843ISK`, is configured using the `mmWave Visualizer` tool. This GUI-based tool facilitates setting up and optimizing sensor parameters by providing an interactive interface that highlights valid parameter ranges, reducing the chances of configuration errors. Once the desired settings are finalized, the configuration is exported as a cfg file, which is then flashed onto the sensor. While manual editing of the configuration file is possible, using the `mmWave Visualizer` is recommended for its ease of use and error prevention. The Constant False Alarm Rate (CFAR) is a key parameter in radar signal processing that differentiates targets from noise. It dynamically adjusts the detection threshold based on the noise level in the radar data, ensuring consistent detection performance in varying noise conditions. By modifying the CFAR parameter, the expected number of point clouds representing an object is controlled, filtering out noise while preserving meaningful detections. In this experiment, CFAR is set to 15. This value balances the sensitivity of object detection against noise suppression, ensuring that objects of interest are accurately detected while minimizing the inclusion of spurious noise points.

In this experiment, the sensor is configured to optimize range resolution and azimuth resolution for the object detection task. The details of the sensor configuration are presented in Table 4.3.

Table 4.3: Sensor Configuration for mmWave Radar (`IWR6843ISK`)

Parameter	Value	Parameter	Value
Frequency	60 GHz	Rise Time and Fall Time	0
Platform	xWR68xx	Tx Antenna Count	1
Scene Classifier	best_range_res	Samples per Chirp	256
Azimuth Resolution (deg)	15 + Elevation	CFAR (dB)	15
Range Resolution (m)	0.047	Range Detection Threshold (dB)	15
Maximum Unambiguous Range (m)	9.04	Doppler Detection Threshold (dB)	15
Maximum Radial Velocity (m/s)	5.09	Range Peak Grouping	Enabled
Radial Velocity Resolution (m/s)	0.64	Doppler Peak Grouping	Enabled
Frame Duration (ms)	33.333	Static Clutter Removal	Disabled
Angle of Arrival FoV	Full FoV	Angle of Arrival FoV	Full FoV
Range FoV	Full FoV	Doppler FoV	Full FoV
Frequency Slope (MHz/μs)	7	Number of Chirps per Frame	3
Idle Time (μs)	24	ADC Sampling Rate	12.5 MHz

4.5.4 Available Code Repositories

There are several code repositories and configuration resources available for integrating and configuring the `IWR6843ISK` mmWave sensor. These repositories, hosted on GitHub and other

platforms, provide essential tools for sensor integration, data logging, and visualization. One of the primary repositories used in this work is `py_mmwave_dev` [109], which offers a Python-based framework for real-time data visualization and logging. This repository enables integration with common Integrated Development Environments (Integrated Development Environment (IDE)s) such as VSCode, allowing users to easily visualize radar data in real-time. It is based on the `pymmw` repository [110], which provides similar functionalities for handling mmWave radar data in Python.

In addition to `py_mmwave_dev`, the repository `iwr6843aop_pub` [111] offers an integration of the IWR6843AOPEVM mmWave sensor with ROS2 (Robot Operating System) on Ubuntu systems (18.04.5 and 20.04.3). While it provides ROS2-compatible drivers for the sensor, it does not support dense radar data output or a data logging feature, making it less suitable for high-resolution data acquisition. For more robust configurations, the `mmwave_ti_ros` repository [112] offers tools for using the `IWR6843ISK` with ROS, including a well-structured configuration file that facilitates dense data acquisition. This repository was particularly useful in this experiment, as it enabled the collection of high-density radar data, crucial for the analysis. By combining the data visualization and logging capabilities of `py_mmwave_dev` with the configuration file from `mmwave_ti_ros`, the most effective approach for obtaining the necessary data was achieved.

5 Experiments and Results

This chapter presents the experimental setup, results, and analysis of the object detection models employed in this research. It provides a detailed explanation of the dataset collection, preprocessing steps, and evaluation of different object detection techniques using both range-azimuth and 3D point cloud data. The objective is to assess the performance of various models under realistic intralogistics conditions, emphasizing the effectiveness of sensor data in diverse environments. By examining offline and online testing scenarios and conducting comparative analyses between the models, this study demonstrates the strengths and limitations of the proposed system in real-world applications. The results obtained from these experiments inform future improvements in sensor integration, dataset preparation, and model optimization for enhanced object detection.

5.1 Dataset Collection

The process of dataset collection focuses on replicating real-world intralogistics scenarios using advanced sensing and recording devices. This approach facilitates the acquisition of high-quality data for research and development in the fields of logistics and automation.

5.1.1 Introduction to Objects and Environment

In the automated and semi-automated logistics industry, detecting various objects is essential for ensuring smooth operations and preventing hazards. These objects include Boxes/Cartons, Forklifts, *Kleinladungsträger* (KLT, or "Small Load Carrier" in German), workstations, small robots, pallets, crates, bins, barrels, trays, loose items, bundles, and rolls. This study focuses on the collection of datasets for four key objects: forklifts, *Kleinladungsträger* (KLT), workstations, and small robots (specifically Robotnik models). Both static and dynamic datasets for these objects are collected, as illustrated in Figure 5.1. These datasets aim to capture diverse scenarios and configurations to support robust object detection. Currently, no publicly available datasets comprehensively address indoor logistics entities with 6G sensing, creating a significant gap in the field. This research addresses this gap by developing a specialized dataset tailored to indoor logistics applications. This dataset provides a valuable resource for further research and practical applications in the logistics domain.

Figure 5.1: Illustrations of dataset objects: (a) Forklift, (b) Forklift with KLT, (c) Workstation, (d) Robotnik.

5.1.2 Setup for Dataset Collection

This dataset is collected at the Technical University of Dortmund, in the logistics hall of the Chair of Materials Handling and Warehousing (FLW), in collaboration with the Fraunhofer Institute for Material Flow and Logistics (IML). The joint projects with IML complement and expand the expertise and scope of the work. The logistics hall, measuring 20 meters by 10 meters, serves as a controlled environment equipped with a Vicon motion capture system. The dimensions and Vicon coordinates of the hall are illustrated in Figure 5.2.

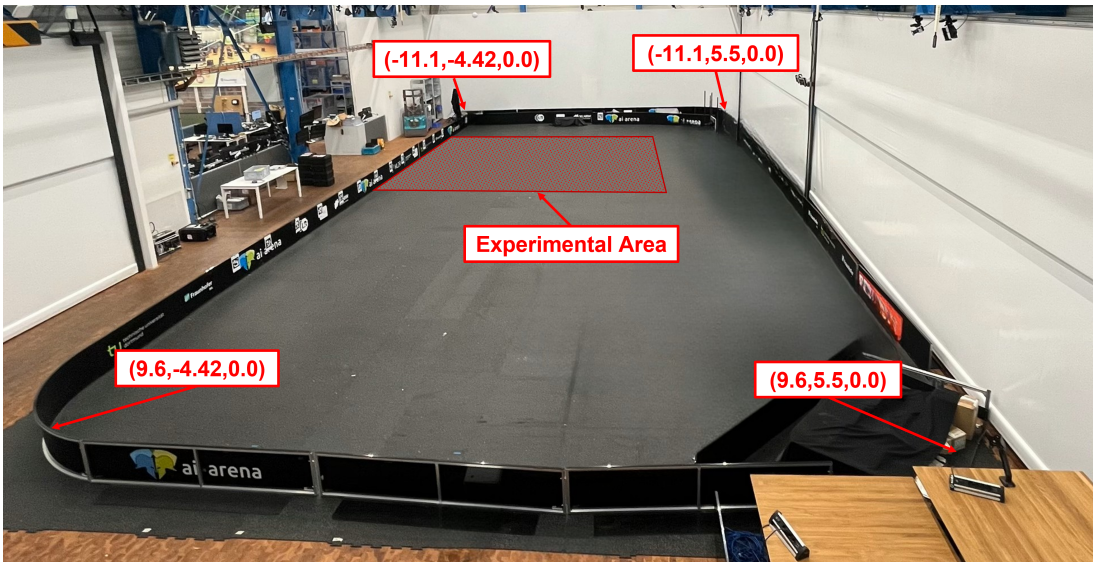


Figure 5.2: Dimensions and Vicon coordinates of the FLW logistics hall.

This study primarily focuses on capturing data using a millimeter-wave (mmWave) radar sensor. The mmWave radar sensor, provided by Texas Instruments (TIs), is mounted on a tripod at a height of 156 cm from the ground, facing the hall arena. Additionally, data is collected using a Logitech Mevo camera for visual reference and result visualization. This camera, mounted at a height of 115 cm from the ground, is also utilized for supervising models in future research. The complete dataset collection setup is depicted in Figure 5.3.

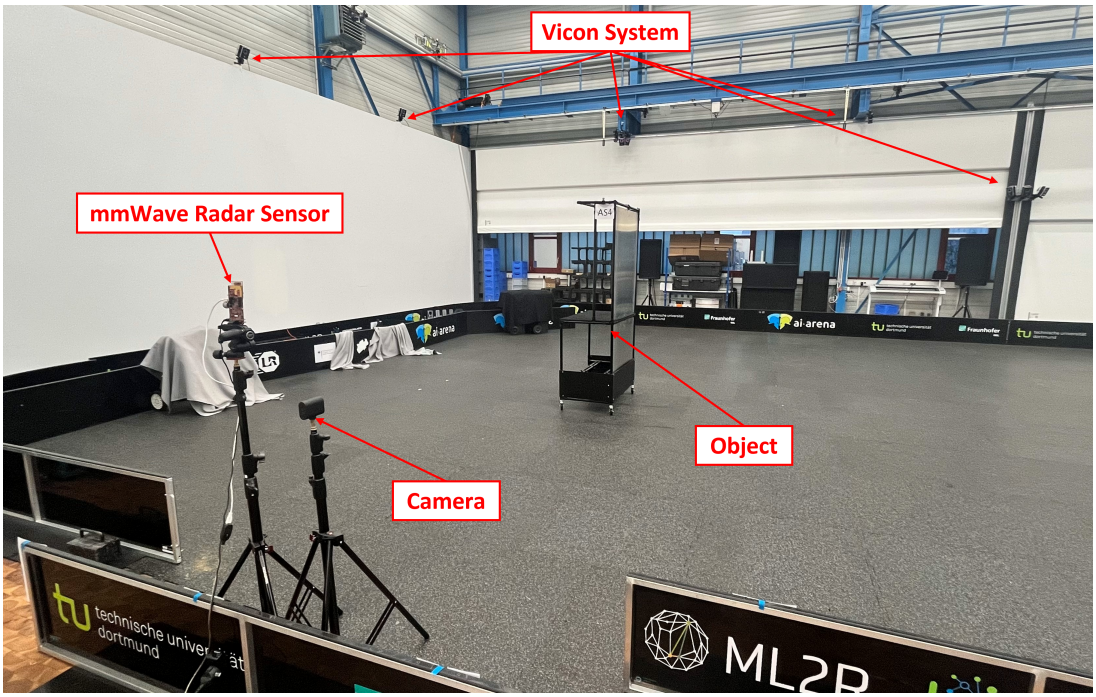


Figure 5.3: Representation of the dataset collection setup.

5.1.3 Dataset Collection Details

In this experiment, a dataset is collected for the specified objects over a period of approximately 3 hours. The data is captured using a combination of a millimeter-wave (mmWave) radar sensor, a camera, and a Vicon motion capture system. The mmWave radar sensor collects data at a rate of 5 frames per second (fps), while the camera operates at 30 fps with a 1080p resolution, covering a 10 meter by 10 meter area of the hall. The Vicon system continuously captures 3D positions and orientations of the objects at a rate of 200 Hz, providing accurate motion tracking throughout the hall. The mmWave radar sensor records several types of data, including 3D point clouds (x, y, and z coordinates), range, azimuth, velocity, and signal-to-noise ratio (Signal-to-Noise Ratio (SNR)) at 5 fps. This configuration enables comprehensive data collection across the entire 10 meter by 10 meter area of the hall, addressing limitations of the camera system, such as blind spots and low visibility.

The static data collection focuses on capturing data for non-moving objects, providing a baseline for object detection. The mmWave radar sensor captures point clouds, velocity, range, SNR, and azimuth data, amounting to 31,713 frames in 1 hour and 45 minutes, with a storage size of 200 MB. The camera records 189,000 video frames in 1080p resolution, consuming approximately 15 GB of storage in the same duration. The Vicon system captures 378,000 frames of 3D positions and orientations, totaling 53 MB of storage. Additionally, annotations are generated for key frames, with 31,713 labels for the objects. These details are summarized in Table 5.1. For the dynamic data collection, which replicates real-world logistic scenarios, the same sensors are used to capture data for moving objects. The mmWave radar collects 7,922 frames, amounting to 100 MB of data. The camera captures 129,600 video frames, consuming 9 GB of storage. The Vicon system records 259,200 frames of 3D positions and orientations, requiring 33 MB of storage. As with the static data, 16,863 annotations are generated for key frames, facilitating future object detection training and evaluation. The time duration for dynamic data collection is 1 hour and 12 minutes. These details are summarized in Table 5.2.

Table 5.1: Static Data Collection

Sensor/System	Data Type	Total Data Points	Storage Size	Time Duration
mmWave Radar	Point clouds, velocity, range, SNR, azimuth	31,713 frames	200 MB	1 hour 45 min
Camera	Video frames	189,000 frames	15 GB	1 hour 45 min
Vicon System	3D positions, orientations	378,000 frames	53 MB	1 hour 45 min
Annotations	Labels (objects)	31,713 annotations	1,47MB	-

Table 5.2: Dynamic Data Collection

Sensor/System	Data Type	Total Data Points	Storage Size	Time Duration
mmWave Radar	Point clouds, velocity, range, SNR, azimuth	16,863 frames	100 MB	1 hour 12 min
Camera	Video frames	129,600 frames	9 GB	1 hour 12 min
Vicon System	3D positions, orientations	259,200 frames	33 MB	1 hour 12 min
Annotations	Labels (objects)	16,863 annotations	762,4KB	-

5.1.4 Example of Logistic Scenarios

Intralogistics involves various scenarios that optimize the internal flow of goods within a warehouse or production facility. These include inbound logistics, where goods are unloaded from delivery vehicles and moved to storage, and order picking, where specific items are retrieved to fulfill customer orders. Cross-docking operations streamline processes by directly transferring goods from inbound to outbound vehicles, bypassing storage. Other examples include storage and inventory management, where goods are organized and repositioned as needed, and returns processing, where returned items are inspected and restocked or disposed of. Each scenario is designed to maximize efficiency while minimizing time, space, and resource usage [113, 114].

In this experiment, intralogistics scenarios are replicated by capturing the movement of hand pallet forklifts, *Kleinladungsträger* (KLT, or small load carriers), and workstations within a hall. The hand pallet forklift is used to transport pallets over short distances, while KLT boxes and workstations represent modular items and assembly stations, respectively. These objects are moved systematically to replicate tasks such as unloading, storage, and order picking. The goal is to capture datasets for these scenarios using sensors, cameras, and motion-capturing devices, enabling the collection of effective datasets for use in machine learning or deep learning models in further research on intralogistics.

5.2 Preprocessing of Collected Dataset

The collected dataset comprises range, azimuth, 3D point cloud, velocity, and signal-to-noise ratio (SNR). To make this data compatible with the selected model, preprocessing is essential. This involves structuring the data in a format suitable for input while ensuring proper annotations to label objects accurately. These annotations are critical for training the model to perform object detection tasks effectively.

5.2.1 Range-Azimuth 2D Histogram

The range-azimuth 2D histogram is a visual representation of radar data, mapping the spatial distribution of objects based on their range (distance from the radar) and azimuth (angular posi-

tion). This type of representation provides a compact and intuitive way to interpret radar signals, making it easier to understand compared to outputs from other sensing devices like cameras or lidar. Unlike camera-based systems, which rely heavily on lighting and environmental conditions, radar systems produce reliable results under various conditions, such as low light or visual obstructions. The range-azimuth 2D histogram simplifies the interpretation of radar data by presenting it in a format that highlights spatial relationships effectively, making it a valuable feature for machine learning model development. Its ability to present radar data in a structured, interpretable format enhances the model's capacity to detect and track objects accurately, especially in complex and challenging environments. The range-azimuth 2D histogram is generated using the Python library `matplotlib` to visualize the radar data by plotting range against azimuth values. Using this approach, the radar data is successfully represented as range-azimuth 2D histograms. An example of this representation is shown in Figure 5.4, which includes a corresponding camera frame for reference. The camera frame provides a visual context for interpreting the radar data by offering a real-world view of the scene.

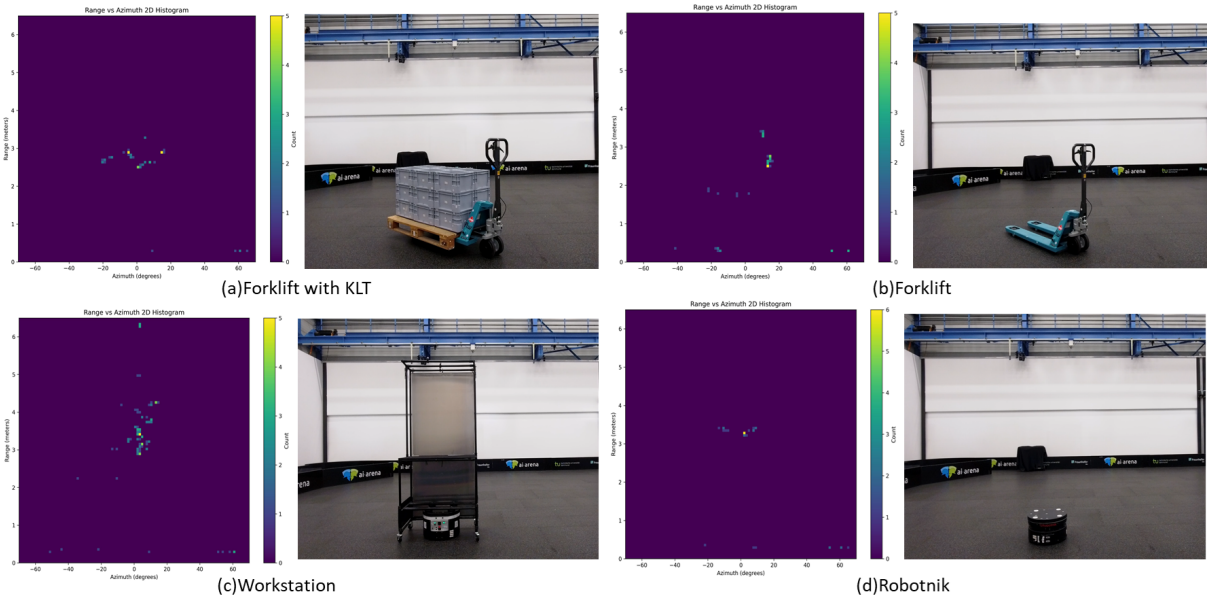


Figure 5.4: Example of a range-azimuth 2D histogram with a reference camera frame.

5.2.2 3D Point Cloud

A 3D point cloud is a collection of data points in a three-dimensional space, typically generated by 3D scanning technologies such as LiDAR, stereo vision, or radar systems. These points represent the external surface of an object or the environment, defined by their spatial coordinates (x, y, z). The 3D point cloud, as generated by the radar system, plays a crucial role in machine learning model development by providing rich spatial information, enabling accurate object detection, tracking, and scene understanding in dynamic environments. In many applications, such as motion tracking or sensor fusion, the 3D point cloud data needs to be aligned with a specific global coordinate system for further processing or analysis.

As mentioned in Section 2.2.2 and shown in Figure 2.11, 3D point clouds are created here. In this context, the radar sensor's coordinate system is mapped to a global reference frame provided by the Vicon motion capture system. The Vicon system serves as the base system, providing precise 3D positional information of objects within the environment. To align the

radar data with the Vicon system, the radar world coordinates are transformed to the Vicon world coordinates [115].

Calibration of 3D point clouds

There are four possible configurations for placing the radar sensor on the border of the arena. The arena has a rectangular shape, and each side represents a potential placement for the radar. These four cases are illustrated in Figure 5.5, which shows the different possible placements of the radar sensor. For this experiment, **Case 1** is used, where the radar sensor is placed on the bottom side of the arena, facing upwards.

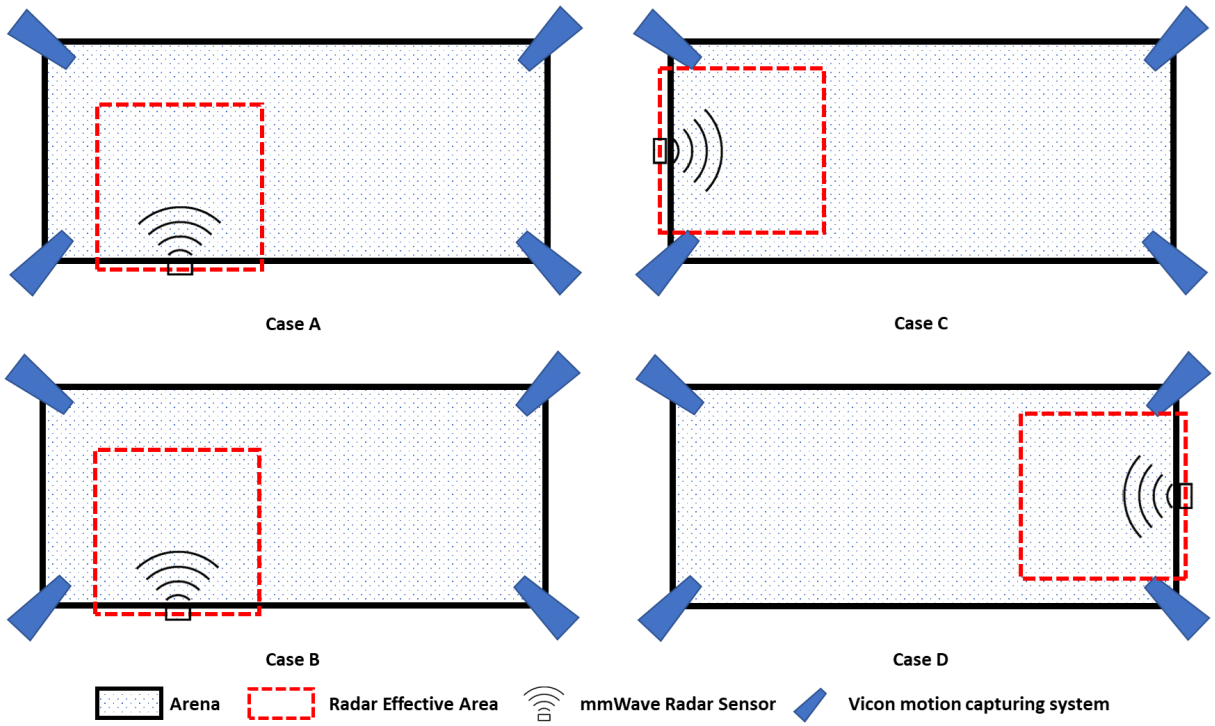


Figure 5.5: Four possible radar sensor positions in the arena.

In each case, the goal is to transform the radar's world coordinates, denoted as (Pr_x, Pr_y, Pr_z), into the Vicon world coordinates, denoted as (Pv_x, Pv_y, Pv_z). The general transformation from radar coordinates to Vicon coordinates can be represented as:

$$\begin{bmatrix} P_{vx} \\ P_{vy} \\ P_{vz} \end{bmatrix} = \begin{bmatrix} T_x & T_y & T_z \\ T_y & T_x & T_z \\ T_z & T_z & T_x \end{bmatrix} \begin{bmatrix} P_{rx} \\ P_{ry} \\ P_{rz} \end{bmatrix} + \begin{bmatrix} R_{ox} \\ R_{oy} \\ R_{oz} \end{bmatrix}$$

Where:

- P_{rx}, P_{ry}, P_{rz} are the radar world coordinates.
- P_{vx}, P_{vy}, P_{vz} are the Vicon world coordinates.
- R_{ox}, R_{oy}, R_{oz} represent the origin of the radar sensor in the Vicon coordinate system.
- T_x, T_y, T_z are the translation parameters of the radar sensor relative to the Vicon coordinate system.

Cases for Radar Sensor Placement

- **Case A: Radar Sensor on the Bottom Side of the Arena, Facing Upwards**

  In this case, the radar is placed on the bottom side of the arena and is oriented upwards, meaning there is no rotation of the radar coordinates relative to the Vicon coordinates. Only translation is applied to the radar coordinates. The transformation is:

$$\begin{bmatrix} P_{vx} \\ P_{vy} \\ P_{vz} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{rx} \\ P_{ry} \\ P_{rz} \end{bmatrix} + \begin{bmatrix} R_{ox} \\ R_{oy} \\ R_{oz} \end{bmatrix}$$

- **Case B: Radar Sensor on the Top Side of the Arena, Facing Downwards**

  In this case, the radar is placed on the top side of the arena and is oriented downwards. The radar coordinates are inverted along the x and y axes, meaning the signs of the x and y components are flipped. The transformation is:

$$\begin{bmatrix} P_{vx} \\ P_{vy} \\ P_{vz} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{rx} \\ P_{ry} \\ P_{rz} \end{bmatrix} + \begin{bmatrix} R_{ox} \\ R_{oy} \\ R_{oz} \end{bmatrix}$$

- **Case C: Radar Sensor on the Right Side of the Arena, Facing Left**

  In this case, the radar is placed on the right side of the arena and is facing left. The radar is rotated 90° counterclockwise, so the x and y coordinates are swapped, and the signs are adjusted accordingly. The transformation is:

$$\begin{bmatrix} P_{vx} \\ P_{vy} \\ P_{vz} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{rx} \\ P_{ry} \\ P_{rz} \end{bmatrix} + \begin{bmatrix} R_{ox} \\ R_{oy} \\ R_{oz} \end{bmatrix}$$

- **Case D: Radar Sensor on the Left Side of the Arena, Facing Right**

  In this case, the radar is placed on the left side of the arena and is facing right. The radar is rotated 90° clockwise, so the x and y coordinates are swapped, and the signs are adjusted. The transformation is:

$$\begin{bmatrix} P_{vx} \\ P_{vy} \\ P_{vz} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{rx} \\ P_{ry} \\ P_{rz} \end{bmatrix} + \begin{bmatrix} R_{ox} \\ R_{oy} \\ R_{oz} \end{bmatrix}$$

Each of these transformations ensures that the radar's coordinates are mapped accurately to the Vicon world coordinates, depending on the specific placement and orientation of the radar sensor within the arena.

After mapping the 3D coordinates, the resulting point cloud can be directly fed into the model, provided it is formatted appropriately for the model's requirements. Python libraries offer convenient tools for converting point cloud data into various file formats suitable for processing, such as .npy, .bin, and .txt. These formats are commonly used in machine learning and computer vision tasks to ensure compatibility with different frameworks and models. Additionally, when preparing the point cloud for use, considerations such as scaling, normalization, and removal of noise or outliers may be necessary to optimize performance and ensure accurate model predictions.

5.2.3 Filtering Multi-Paths and Boundary Constraints

The first filtering technique used in this experiment involves applying boundary conditions to remove any data points that fall outside the predefined boundaries of the experimental area. As described in Section 5.1.2 and Figure 5.2, the experiment area is defined, and all radar readings outside this region are considered invalid and are excluded from further processing. The boundary condition can be expressed mathematically as:

$$\text{Filtered Point Cloud} = \{p \mid p_x \in [x_{\min}, x_{\max}], p_y \in [y_{\min}, y_{\max}], p_z \in [z_{\min}, z_{\max}]\} \tag{5.1}$$

This ensures that only points within the experimental area are considered, reducing the influence of irrelevant or outlier data. Figure 5.6 illustrates the difference between raw input and filtered input data with boundary conditions, showing how the application of boundary constraints refines the radar data by eliminating irrelevant points outside the specified region.

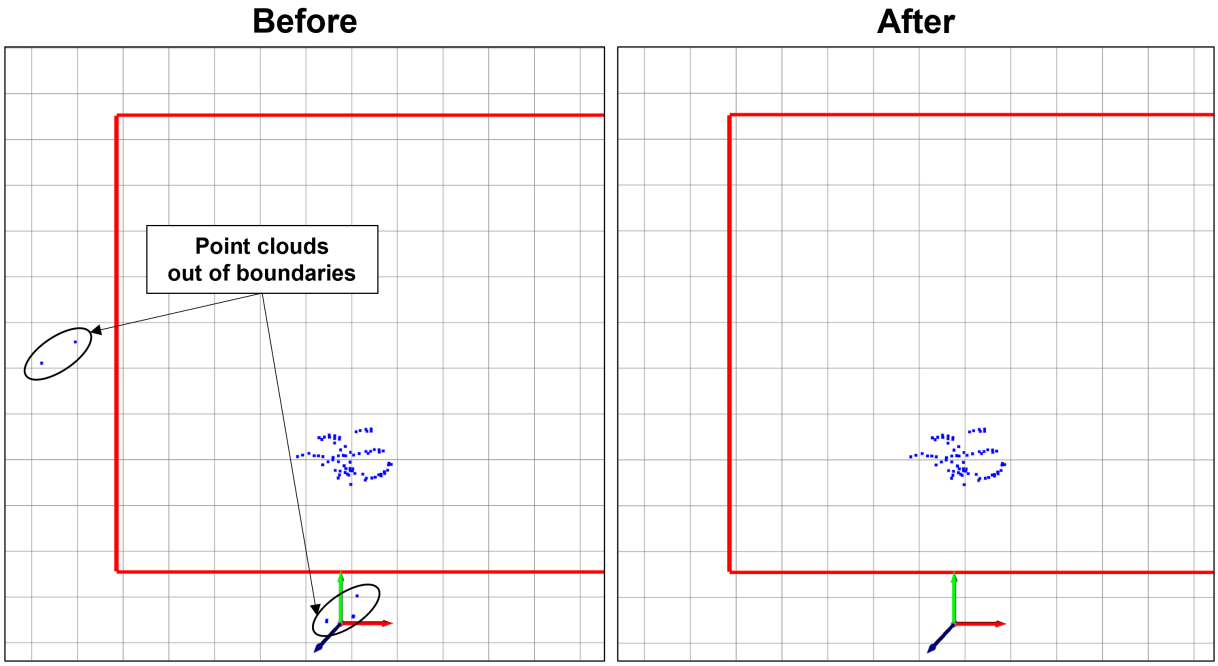


Figure 5.6: Boundary Condition: Comparison between raw input and filtered input with boundary conditions. Points outside the experiment area are excluded from further processing.

The second filtering technique is Multipath reflection in radar sensors refers to the phenomenon where radar signals bounce off multiple surfaces before returning to the sensor, resulting in several reflected signals with varying path lengths. This can distort the radar readings by causing multiple echoes from a single object, leading to inaccurate distance and position estimations. In mmWave radar sensors, the Signal-to-Noise Ratio (SNR) values are collected for each reflected point, representing the ratio of the power of the reflected signal to the noise. Higher SNR values indicate clearer, more reliable readings, while lower values correspond to weaker or noisy signals. To mitigate the effects of multipath reflections, point cloud data is filtered based on these SNR values, removing less reliable points and correcting the sensing readings [116]. The filtering process can be mathematically expressed by removing points where the SNR value falls below a threshold, as shown in Equation 5.2:

$$\text{Filtered Point Cloud} = \{p \mid \text{SNR}(p) > \text{SNR}_{\text{threshold}}\} \tag{5.2}$$

Figure 5.7 illustrates the difference between raw input data and the filtered input data with multipath reflections, highlighting how the SNR-based filtering enhances the accuracy of the radar sensor readings.

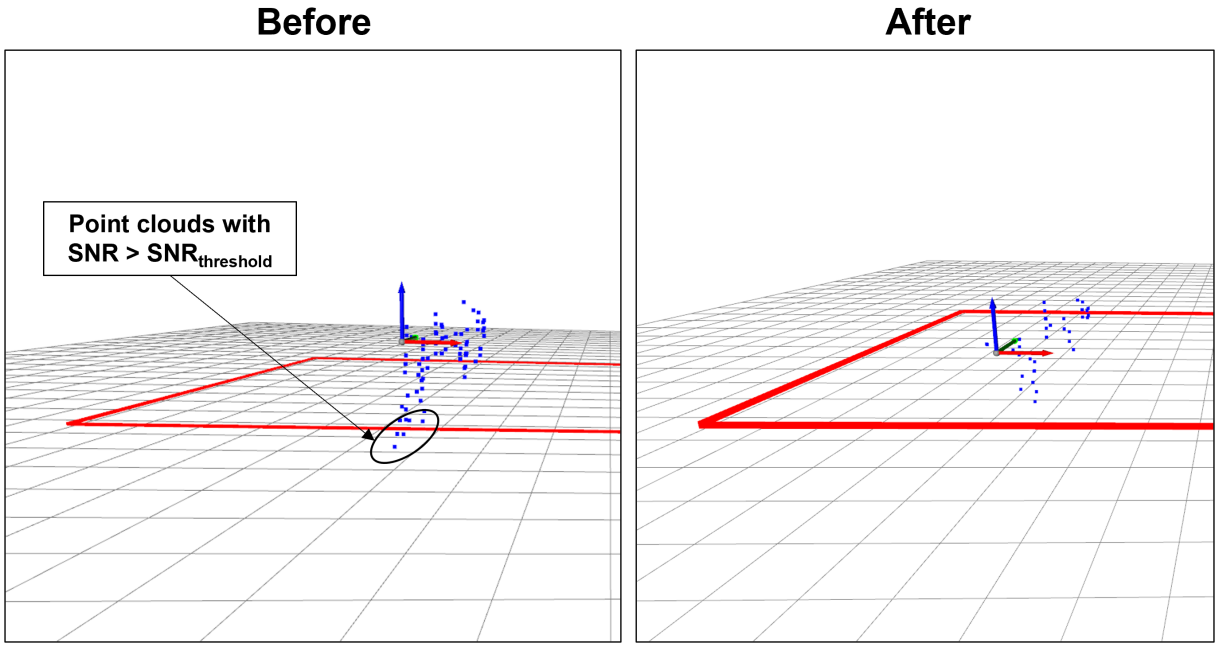


Figure 5.7: Multipath Reflection: Comparison between raw input and filtered input with multipath. The filtering process removes less reliable readings based on SNR values.

5.2.4 Data annotation

To train a supervised machine learning model, it is essential to have annotations corresponding to each frame of the dataset. Several annotation tools are available online, tailored to different models and frameworks. For this work, specific annotation tools are utilized to label the collected dataset effectively.

For the **range-azimuth 2D histogram** data, which is saved in image format after preprocessing, the **LabelImg** tool is employed. LabelImg is a lightweight and user-friendly image annotation tool designed for labeling object bounding boxes in images. It is an open-source software developed by Tzutalin in 2015, written in Python, and utilizes QT for its graphical interface. LabelImg supports generating labels in YOLO text format or VOC XML format. In this study, the YOLO format is used, as the dataset is intended for training the YOLOv7 model. [117].

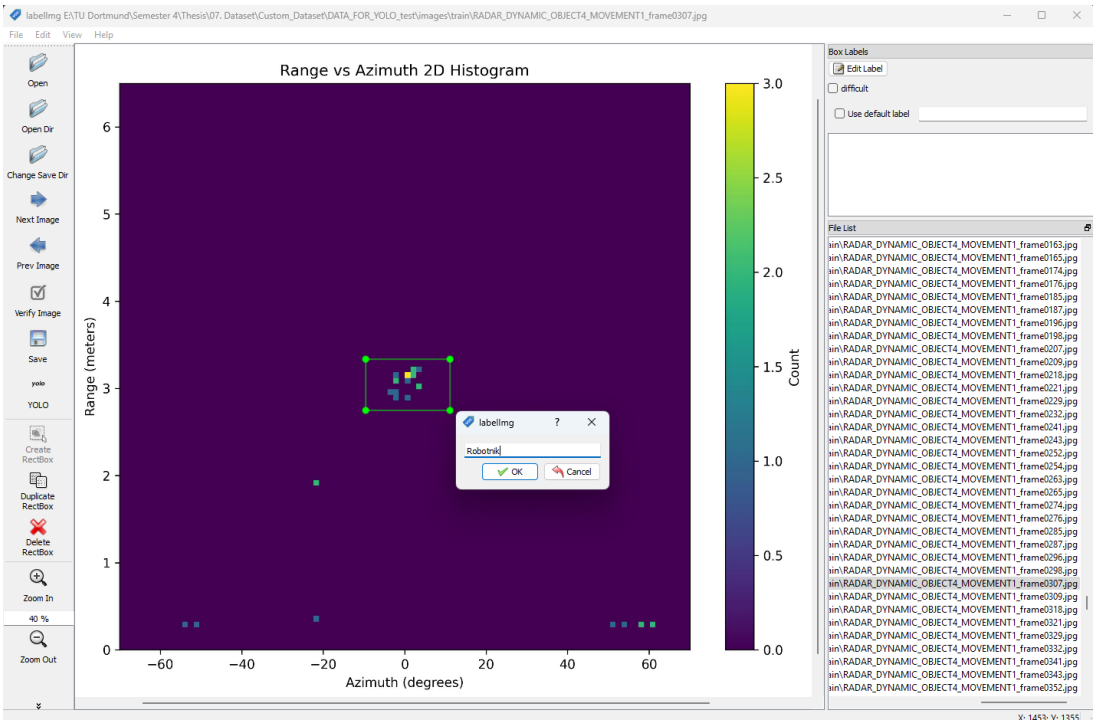


Figure 5.8: Labeling range-azimuth data using the LabelImg tool.

Although LabelImg is straightforward, it requires some technical knowledge, such as familiarity with the command line, for efficient usage. The software can be easily installed via pip, assuming Python 3 is already available on the system. Figure 5.8 demonstrates the interface of LabelImg. The labeling process involves selecting the images to be annotated, drawing bounding boxes around objects, and saving the annotations in a specified directory. The tool allows for the assignment of predefined object classes and generates files containing the bounding box coordinates for use in training object detection models.

For annotating 3D point cloud data, the **labelCloud** tool is utilized, a lightweight application for creating 3D bounding boxes [118]. LabelCloud supports two modes of labeling: picking mode and spanning mode, along with various keyboard and mouse controls for efficient correction. Figure 5.9 illustrates the interface of the labelCloud tool and its workflow for annotating point clouds. The labeling process in labelCloud involves selecting the bounding box's position and dimensions in the point cloud data using either the picking or spanning modes. Picking mode allows for selecting the front-top edge of the box, while spanning mode requires defining the box by selecting four vertices, with simplified constraints for the last two vertices. Bounding boxes can be adjusted in translation, dimensions, and rotation through the control panel or keyboard shortcuts, ensuring accurate labeling. By default, x- and y-rotations are locked. To label 9-Degrees of Freedom (DoF) bounding boxes, disable the *z-Rotation Only Mode* in the settings or `config.ini` file. This allows free rotation around all three axes.

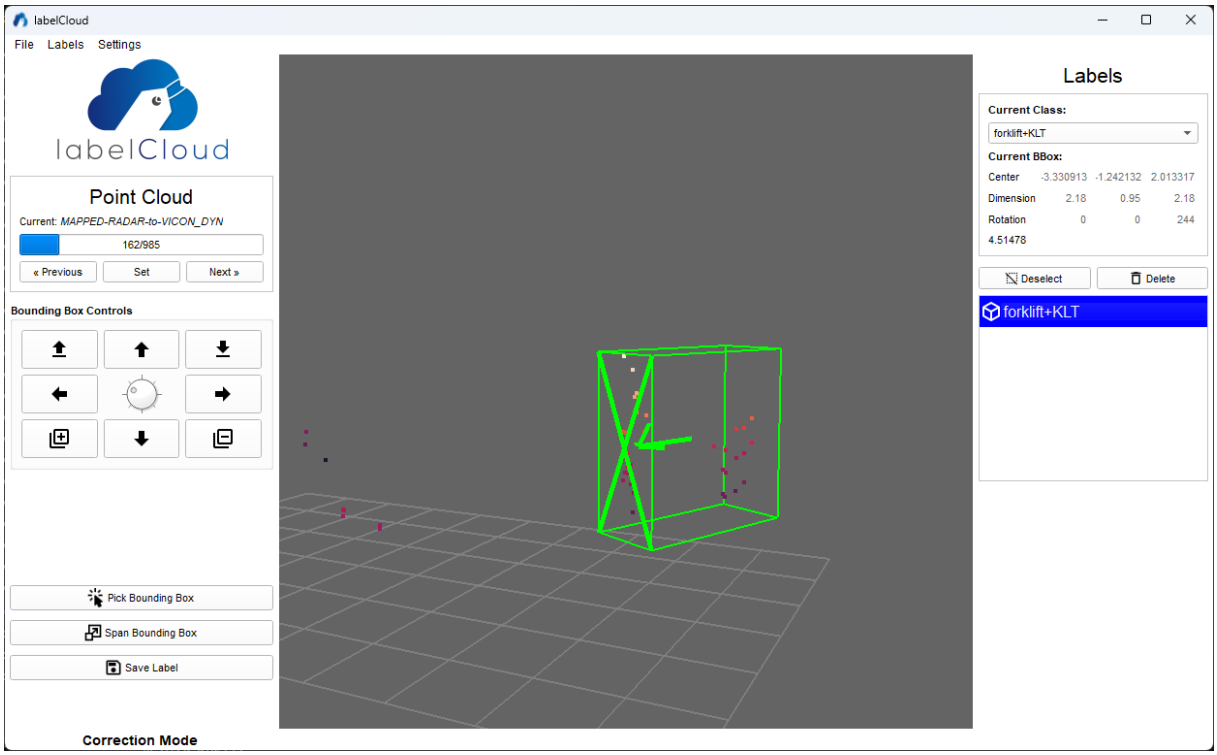


Figure 5.9: Labeling 3D point cloud data using the LabelCloud tool.

Supported Formats in LabelCloud:

- **Import Formats:** Colored: `*.pcd`, `*.ply`, `*.pts`, `*.xyzrgb`. Colorless: `*.xyz`, `*.xyzn`, `*.bin (KITTI)`.
- **Export Formats:** Centroid Relative, Centroid Absolute, Vertices, KITTI, KITTI Untransformed.

In this work, the `.pcd` format is used as the import format, and the *Centroid Absolute* format is employed for export. This labeled 3D point cloud data is now ready for use in object detection models.

5.2.5 Data splitting

In this experiment, the dataset is split into 80% for training and 20% for testing and validation. Specifically, for the range-azimuth 2D histogram, a total of 25,065 frames are allocated for training, while 6,269 frames are used for testing. Similarly, for the 3D point cloud data, 13,793 frames are used for training, and 3,451 frames are designated for testing. This 80-20 split ensures that the model is trained on a sufficiently large portion of the data, while still having a separate test set to evaluate its performance and generalization capabilities.

5.3 Object Detection with Range-Azimuth

The process of object detection with range-azimuth data focuses on leveraging radar-generated spatial information for robust object identification in indoor logistics environments. This involves analyzing the collected dataset to develop models capable of interpreting range-azimuth features for accurate and efficient object detection. By processing radar data to extract meaningful features, the goal is to build a system that operates effectively across diverse logistics scenarios, ensuring reliable performance under static and dynamic conditions. This section details the methodologies adopted for offline and online model testing, the comparative analysis of different models, and the insights gained through experimental evaluations.

5.3.1 Offline Model Testing

Offline model testing involves evaluating a trained model on a collected and preprocessed dataset without real-time deployment. This process aims to identify the most suitable framework and backbone architecture that yield optimal results for object detection tasks in intralogistic environments. Given the prepared dataset, the focus is on ensuring a seamless integration of all components involved in the testing pipeline. To achieve this, the dataset is preprocessed and organized to align with the specific input requirements of the selected models. The offline model testing workflow, as illustrated in Section 4.2 and Figure 4.5, demonstrates the sequential steps for an effective evaluation process. Initially, data collection is conducted as outlined in Section 5.1, followed by preprocessing that involves structuring the data based on range-azimuth information. Subsequently, labels are generated for the dataset under a supervised learning paradigm. This processed and labeled dataset is then split into training and testing subsets, which are fed into the model for training and evaluation. The output is the detection of objects relevant to intra-logistic operations. In this experiment, the model is tested on both static and dynamic datasets to evaluate its adaptability and robustness under varying data conditions.

The configuration of the model plays a crucial role in this process. It involves specifying the class names (i.e., object categories) and the total number of unique classes in the dataset. Additionally, the directory location of the dataset must be provided. Certain models may require metadata for the dataset, necessitating data registration before initiating the training process. For training purposes, the openly available YOLOv7 model with a backbone convolutional neural network (CNN) is employed. This choice of model leverages YOLOv7's state-of-the-art capabilities in object detection and provides a robust foundation for accurate and efficient predictions in intra-logistic environments [18]. During the training phase, parameters such as the number of epochs and batch size are pivotal. An epoch represents one complete pass through the entire training dataset, while the batch size determines the number of samples processed before updating the model's parameters. For this study, the model is trained for 100 epochs with a batch size of 16. This configuration ensures sufficient iterations for the model to learn complex patterns effectively.

For this experiment, a system running Ubuntu 22.04, equipped with an NVIDIA GeForce RTX 4070 GPU with 12.0 GB of memory, is utilized. This hardware configuration ensures efficient handling of computationally intensive training and testing processes. Offline model testing integrates data preprocessing, model configuration, and training to identify the most effective model framework and architecture for object detection in intra-logistic environments. This

structured approach ensures reliable and meaningful evaluation of the models.

5.3.2 Analysis and Results for Offline Model Testing: Radar Azimuth

The model is trained for 100 epochs and evaluated using both static and dynamic datasets collected from indoor logistic scenarios. Due to the larger size of the static dataset compared to the dynamic dataset, the time required for training on the static dataset is significantly higher. The model training for the static dataset takes approximately 22 hours and 14 minutes, whereas training on the dynamic dataset completes in around 12 hours and 55 minutes. Following the evaluation, the model generates results including output frames with detected objects, relative performance graphs, and other performance metrics.

The output frames demonstrate successful object detection, indicating that the model is effectively trained on the custom dataset of indoor logistics. To further validate the results, camera frames collected during data recording are used as a reference. By matching the timestamps of radar and camera data, the detected objects are visualized in a more comprehensive manner. Some representative frames showcasing object detection alongside corresponding camera frames are presented in Figures 5.10, 5.11, 5.12, and 5.13. Figure 5.10 shows the detection output for a Forklift with KLT, Figure 5.11 shows a Forklift, Figure 5.12 displays a Workstation, and Figure 5.13 illustrates a Robotnik robot. It is evident that the performance of the model trained on the static dataset is superior to that of the dynamic dataset, primarily due to the limited object movement in static scenarios. However, the overall performance difference between the two datasets is minimal.

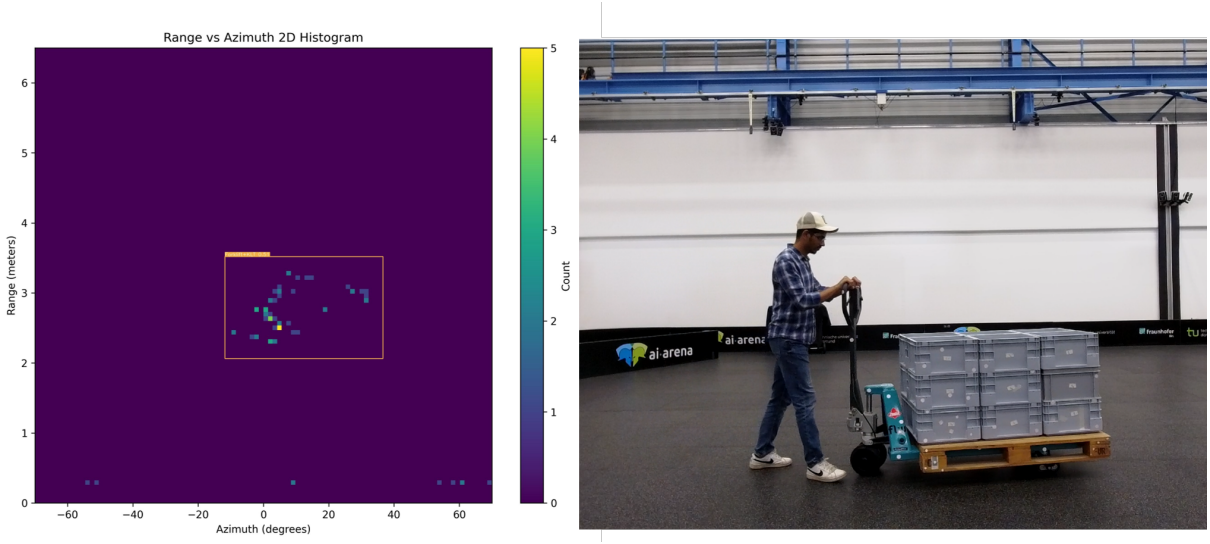


Figure 5.10: Object detection output for Forklift with KLT.

To better understand the evaluation metrics, key terms such as Intersection over Union (Intersection over Union (IoU)), Average Precision (AP), mean Average Precision (mAP), background false positive (False Positive (FP)), false negative (False Negative (FN)), precision, and recall are defined as follows. IoU is a measure that quantifies the overlap between a predicted bounding box and a ground truth bounding box. It plays a fundamental role in evaluating the accuracy of object localization. Average Precision (AP) computes the area under the precision-

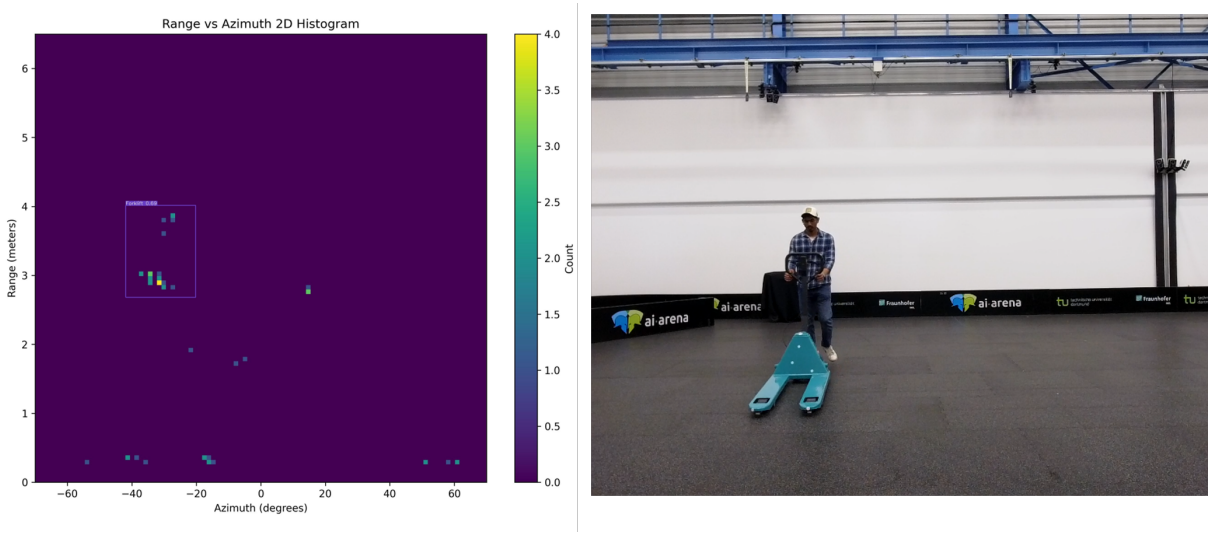


Figure 5.11: Object detection output for Forklift.

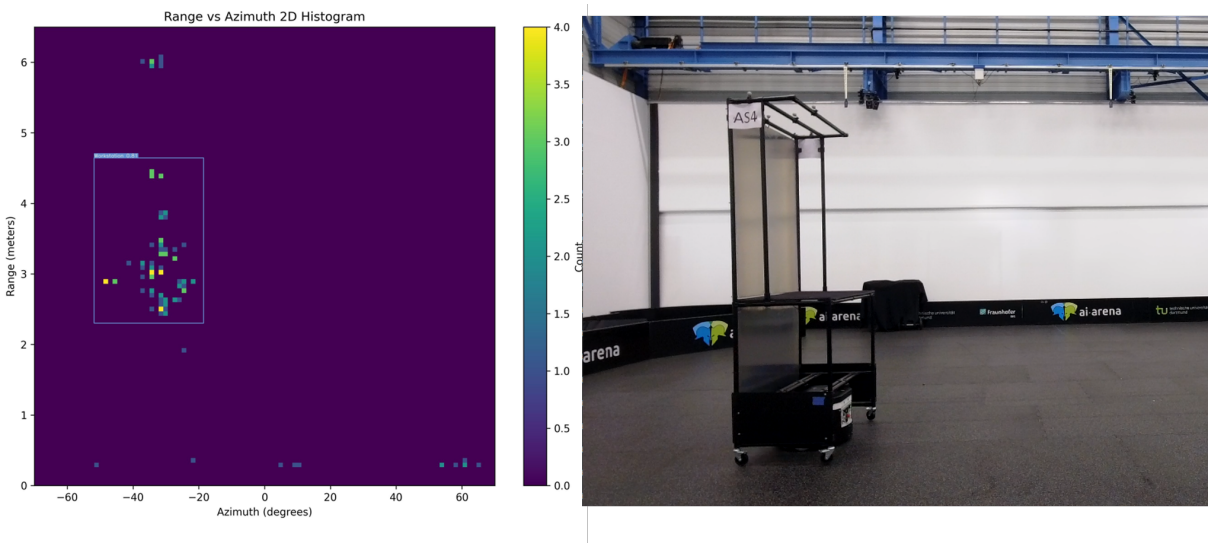


Figure 5.12: Object detection output for Workstation.

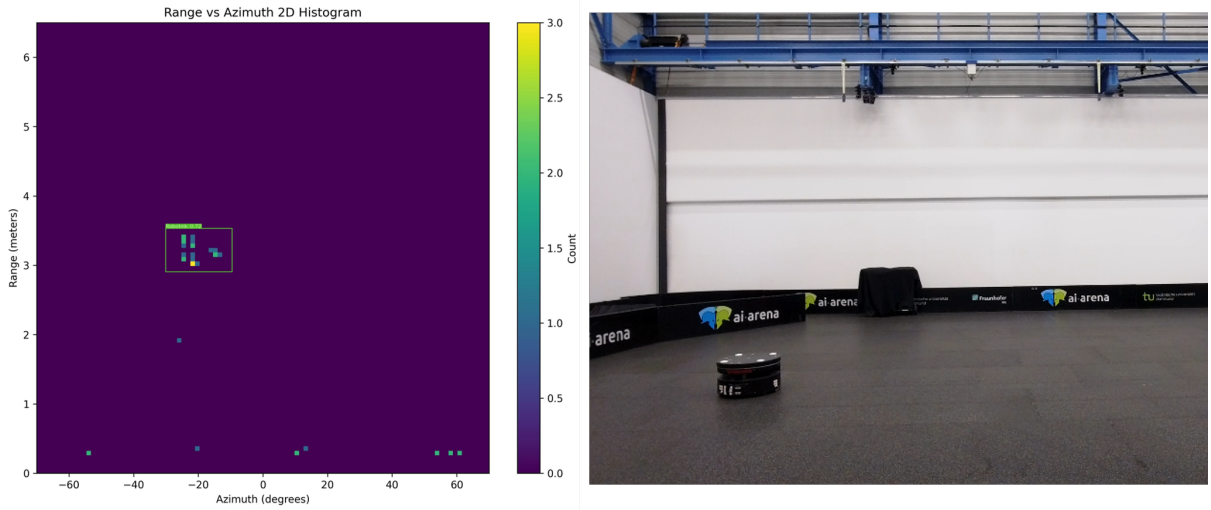


Figure 5.13: Object detection output for Robotnik robot.

recall curve, providing a single value that encapsulates the model's precision and recall performance:

$$\mathrm{AP}_i = \int_0^1 \mathrm{Precision}(r)\, dr. \tag{5.3}$$

The mAP extends the concept of AP by calculating the average AP values across multiple object classes, defined as:

$$\mathrm{mAP} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{AP}_i, \tag{5.4}$$

where N is the number of classes, and AP_i is the Average Precision for class i. The mAP@0.5 and mAP@0.95 represent the mean Average Precision computed at Intersection over Union (IoU) thresholds of 0.5 and 0.95, respectively, where mAP@0.5 evaluates performance at a lower threshold and mAP@0.95 emphasizes stricter detection accuracy with a higher threshold. Precision quantifies the proportion of correctly identified objects out of all predicted objects and is given by:

$$\mathrm{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}, \tag{5.5}$$

while recall measures the ability of the model to correctly identify all objects of interest and is expressed as:

$$\mathrm{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}. \tag{5.6}$$

Background false positive (FP) refers to the incorrect classification of background regions as objects, while false negative (FN) refers to instances where actual objects are not detected by the model. These errors directly affect precision (Equation 5.5) and recall (Equation 5.6). The mAP is a more holistic evaluation metric, as it accounts for both precision and recall across all detection thresholds [119].

The confusion matrix provides insights into the performance of the trained models. For the static dataset-trained model, the confusion matrix values are close to 1 across all classes, with the exception of the forklift class, which shows a background false positive (FP) rate of 0.62. Other background FP and false negative (FN) values are nearly zero. On the other hand, the confusion matrix for the dynamic dataset-trained model exhibits values in the range of 0.7 to 0.9, with minor confusion between objects, reflected in values near 0.1. Overall, the confusion matrices for both models indicate good performance, confirming that the trained models can effectively detect and distinguish objects using radar range-azimuth data. The confusion matrices for static and dynamic models are illustrated in Figure 5.14 and Figure 5.15 respectively.

Additionally, a comparison of performance metrics, including mAP values and model curves, is conducted to evaluate accuracy improvements over time. The mAP@0.5 for the static dataset-trained model is 0.998, whereas for the dynamic dataset-trained model, it is 0.816. The precision-recall (PR) curve analysis provides further insights. For the dynamic dataset, the smaller size of the Robotnik robot results in sparse point cloud data in dynamic frames, which significantly affects detection performance. Specifically, the mAP for the Robotnik robot class is 0.65, contributing to an overall lower mAP. For other classes, the mAP values are 0.909 for the Forklift with KLT, 0.745 for the Forklift, and the workstation class. The F1 curve for the static dataset appears almost rectangular, indicating consistent high performance with values of 0.99 for all classes at 0.59. In contrast, the F1 curve for the dynamic dataset resembles the positive part of a sine wave, with values of 0.79 for all classes at 0.367. These results are summarized in Figure 5.16.

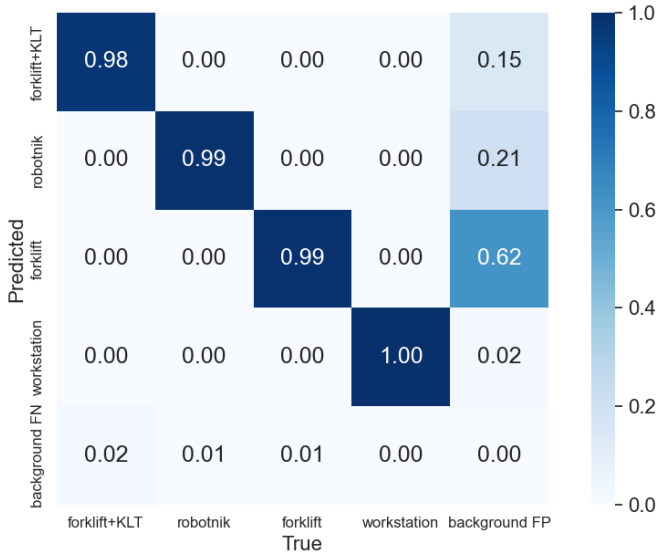


Figure 5.14: Confusion matrix for static dataset-trained models evaluated on test data.

Figure 5.15: Confusion matrix for dynamic dataset-trained models evaluated on test data.

The training results over epochs are presented in Figure 5.17, which illustrates the performance of the model across different evaluation metrics. As observed in the precision curve, the precision value increases steadily from approximately 0 to near 1.00 as the number of epochs progresses, eventually saturating close to 1.0. This saturation is achieved within the first 10 epochs and remains almost constant thereafter, showing that the model has learned to make accurate predictions. A similar trend is observed for the Recall and mAP@0.5, with both metrics steadily improving as the model trains and eventually stabilizing at values near 1.0. This indicates that the model's ability to identify relevant objects (Recall) and its mean average precision (mAP@0.5) converge to optimal performance.

However, the behavior of the mAP@0.95 curve deviates from this pattern. Unlike mAP@0.5,

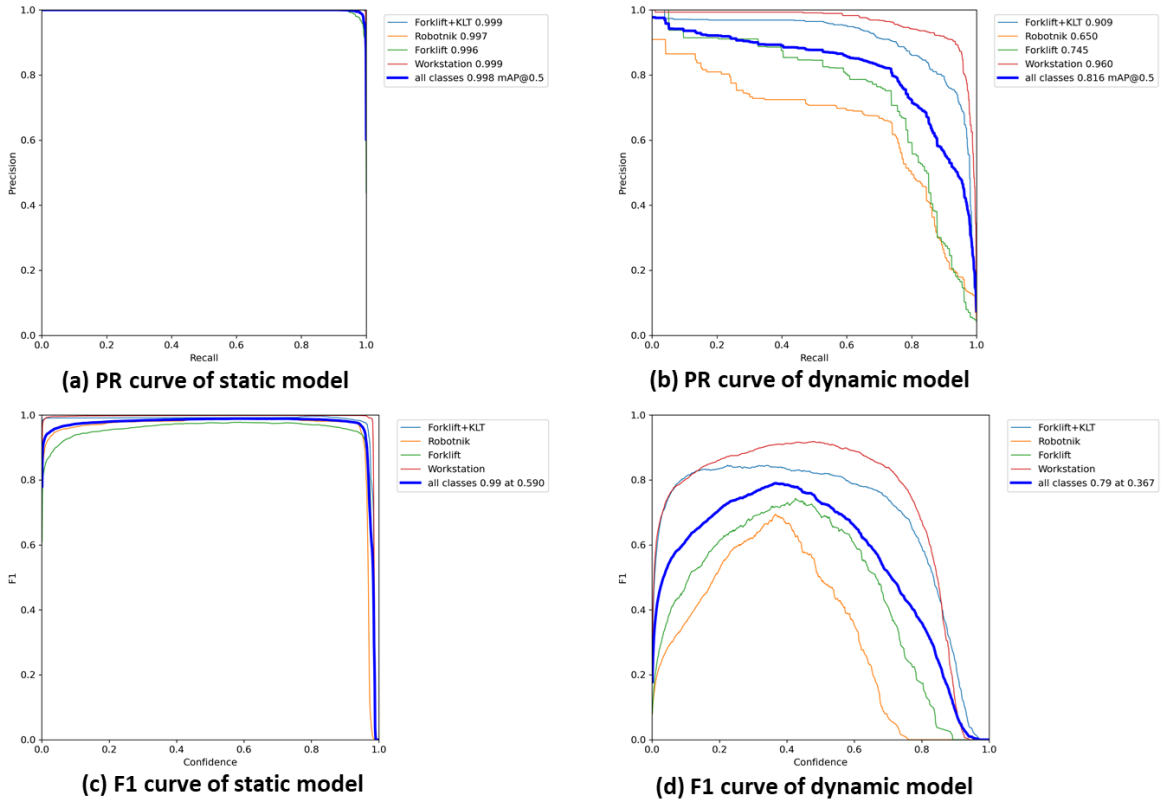


Figure 5.16: (a) Precision-recall (PR) curve of static model, (b) PR curve of dynamic model, (c) F1 curve of static model, and (d) F1 curve of dynamic model.

the mAP@0.95 metric takes a longer time to reach saturation. Initially, the curve shows a more gradual increase, particularly in the first 0 to 60 epochs. During this phase, the model improves its precision, but the rate of improvement slows as it approaches saturation. After epoch 60, the mAP@0.95 curve begins to stabilize, approaching a value close to 1.0, although at a slower rate compared to mAP@0.5. This suggests that the model takes more time to accurately predict objects at higher confidence thresholds (0.95) compared to lower thresholds (0.5), which may be due to the increased difficulty of making precise predictions at higher IoU (Intersection over Union) thresholds. The metrics for Box, Objectness, and Classification exhibit a logarithmic growth pattern across the epochs. This behavior indicates that these values improve gradually during the initial epochs and stabilize at a slower rate as the training progresses. The logarithmic nature of these curves is typical when the model is learning basic features (such as object detection and classification) early on and becomes more confident over time as it fine-tunes these aspects.

In summary, the results demonstrate that the trained models are capable of effectively detecting and distinguishing objects in indoor logistic environments using radar data. The static dataset yields a superior performance with an mAP@0.5 of 0.998 across all classes, while the dynamic dataset achieves an mAP@0.5 of 0.816. Despite the performance difference, the dynamic dataset results remain competitive, highlighting the adaptability of the model.

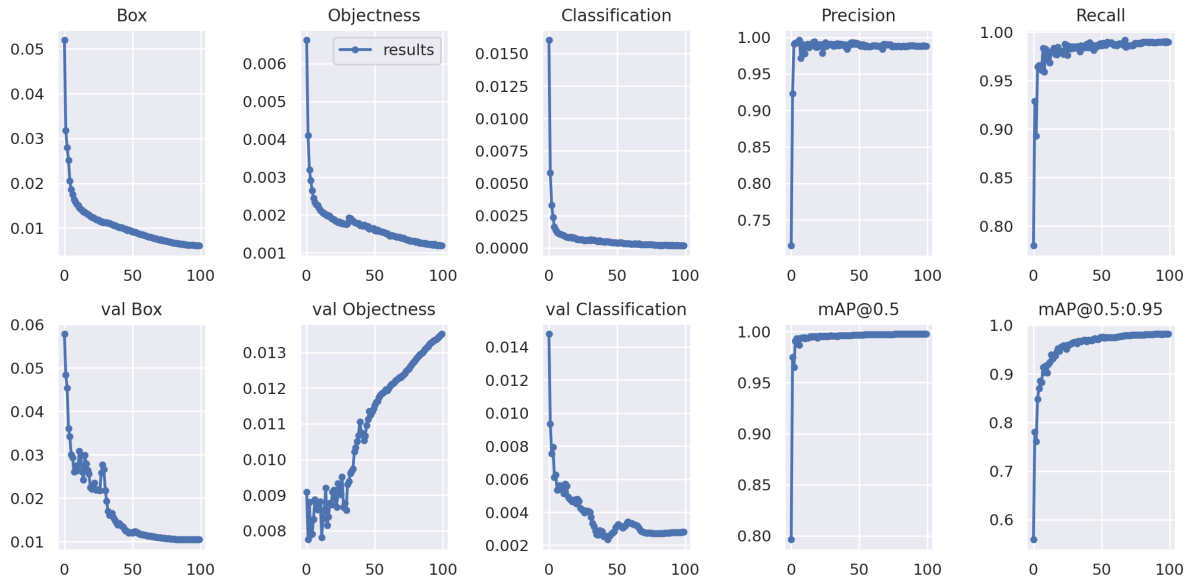


Figure 5.17: Training results over epochs showing precision, recall, mAP@0.5, mAP@0.95, box, objectness, and classification metrics.

5.3.3 Comparative Analysis of Models

In this section, a comparative analysis of model performance is performed by evaluating precision with different backbone algorithms. The objective is to identify the most suitable backbone algorithm for an indoor logistics environment. Previously, a CNN backbone was utilized; now, the comparison is extended by including Faster R-CNN and RetinaNet as backbone algorithms. To implement Faster R-CNN and RetinaNet, the Detectron2 framework is used. All models are trained for 100 epochs with a batch size of 16, ensuring sufficient training iterations for convergence. Furthermore, an identical data split of 80% for training and 20% for testing is applied across all backbone algorithms. This ensures consistency and fairness in the evaluation process.

Figure 5.18 presents the confusion matrices for the different backbone algorithms: CNN, Faster R-CNN, and RetinaNet. By analyzing these matrices, insights can be gained into the predictive performance of each model in relation to the ground truth. From Figure 5.18, it is evident that the CNN backbone demonstrates the most precise predictions compared to the other algorithms. The CNN model achieves high precision in identifying Forklift, KLT, Robotnik, and Workstation objects, with values nearing 0.9. However, the Forklift object shows slightly lower precision, with a value of 0.71. In contrast, the Faster R-CNN model exhibits lower precision overall, with values ranging from 0.7 to 0.8 for most objects except Robotnik, which has a value of 0.59. RetinaNet displays a more balanced performance, with precision values consistently in the range of 0.7 to 0.8. However, its overall precision is slightly lower than that of the CNN model.

This analysis shows that while both RetinaNet and Faster R-CNN provide solid predictions, the CNN backbone proves to be the best fit for the indoor logistics environment. Faster R-CNN, with its two-stage approach that emphasizes accuracy through region proposals and refined classification, performs well but can struggle with speed, making it less ideal for real-time applications. Similarly, RetinaNet, which uses focal loss to handle class imbalance, achieves good results but doesn't quite match the precision of the CNN backbone in this specific scenario. The CNN model stands out because of its ability to learn spatial hierarchies and capture detailed

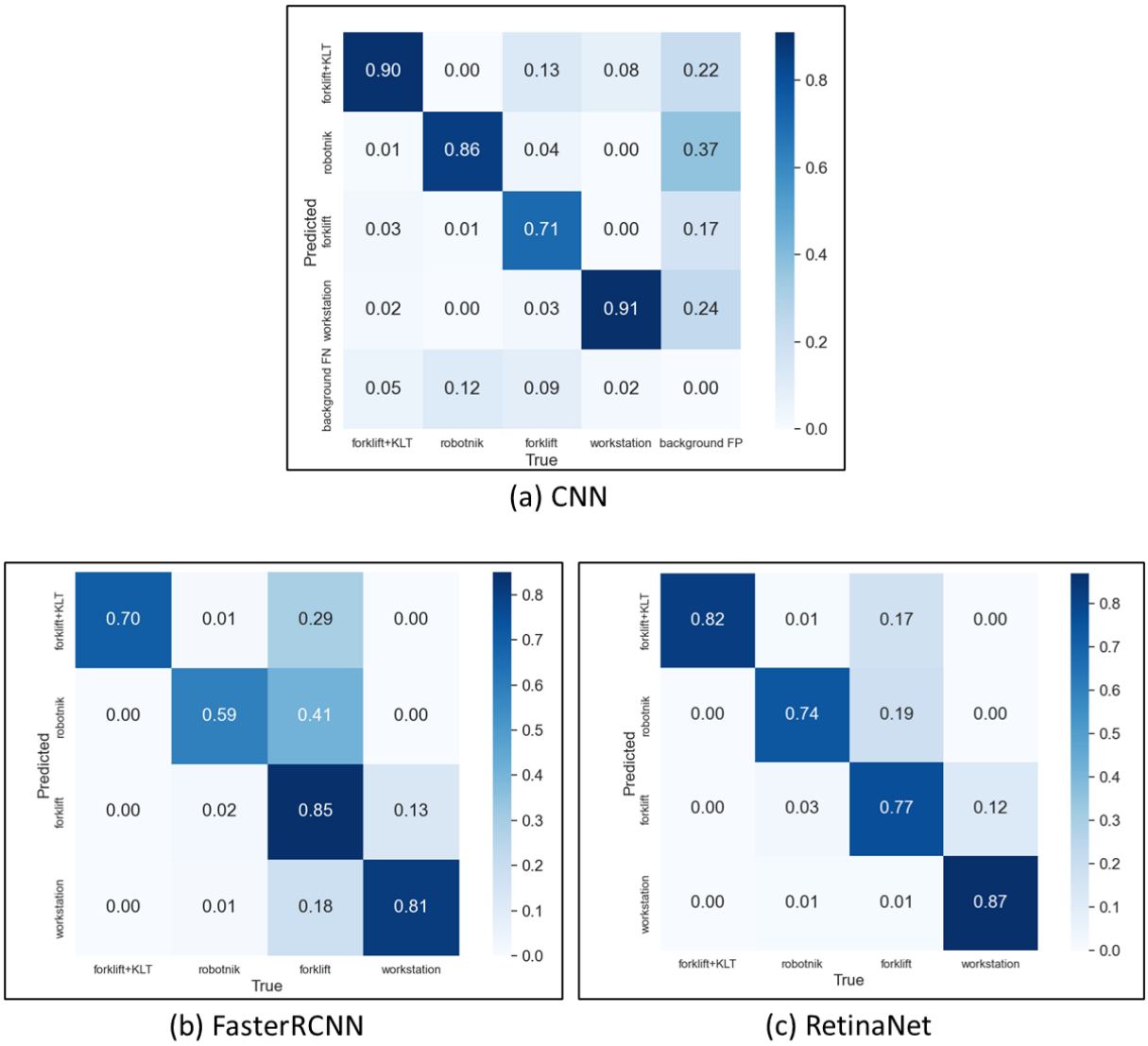


Figure 5.18: Confusion matrix comparison of CNN, Faster R-CNN, and RetinaNet backbone algorithms

local features, which is crucial in cluttered and dynamic indoor spaces. This makes it the most reliable option for accurate object detection in this setting.

5.3.4 Online Model Testing

Online model testing refers to the evaluation of a trained model in a real-time environment to assess its performance under dynamic, real-world conditions. In this experiment, the trained YOLOv7 model with a CNN backbone, as described in the previous section, is used. The procedure for online model testing is depicted in Section 4.2 and Figure 4.6. This block diagram outlines the complete workflow, starting from sensor data collection, preprocessing, feeding the data into the trained YOLOv7 model, and finally visualizing the output. The visualization includes both processed data and real-time detection results, providing insights into the model's behavior in a dynamic environment. In this experiment, to test the durability of the model during online testing, additional changes were introduced to create scenarios that differ from those in the trained dataset. Specifically, the dynamic motion of objects in the trained dataset

was limited to movements in either vertical or horizontal directions within the hall arena. For the online testing, 200 frames were collected for each object, with the objects moving in diagonal or other complex directional patterns across the hall arena, while keeping the rest of the setup similar to the original dataset collection process.

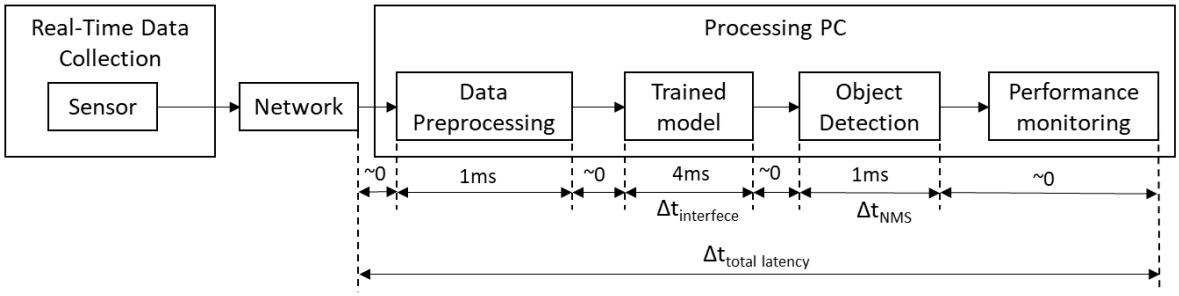


Figure 5.19: Overview of the latency components in online model.

This setup allows for the evaluation of the durability and robustness of the model when exposed to new and unseen data. The diagram in Figure 5.19 illustrates the overview of latency component in the online model testing. The latencies at each stage are measured: 1 ms for data collection, 4 ms for network transmission, and 1 ms for data preprocessing. The total latency ($\Delta t_{\text{total latency}}$) is influenced by the interface latency ($\Delta t_{\text{interface}}$) and the non-maximum suppression latency (Δt_{NMS}). Inference refers to the process of running the trained YOLO model on input data to make predictions. Non-Maximum Suppression (Non-Maximum Suppression (NMS)) is a post-processing step used in object detection models like YOLO. It removes overlapping bounding boxes that refer to the same object, keeping only the box with the highest confidence score. This sequence of steps demonstrates the critical components and time delays involved in real-time object detection systems.

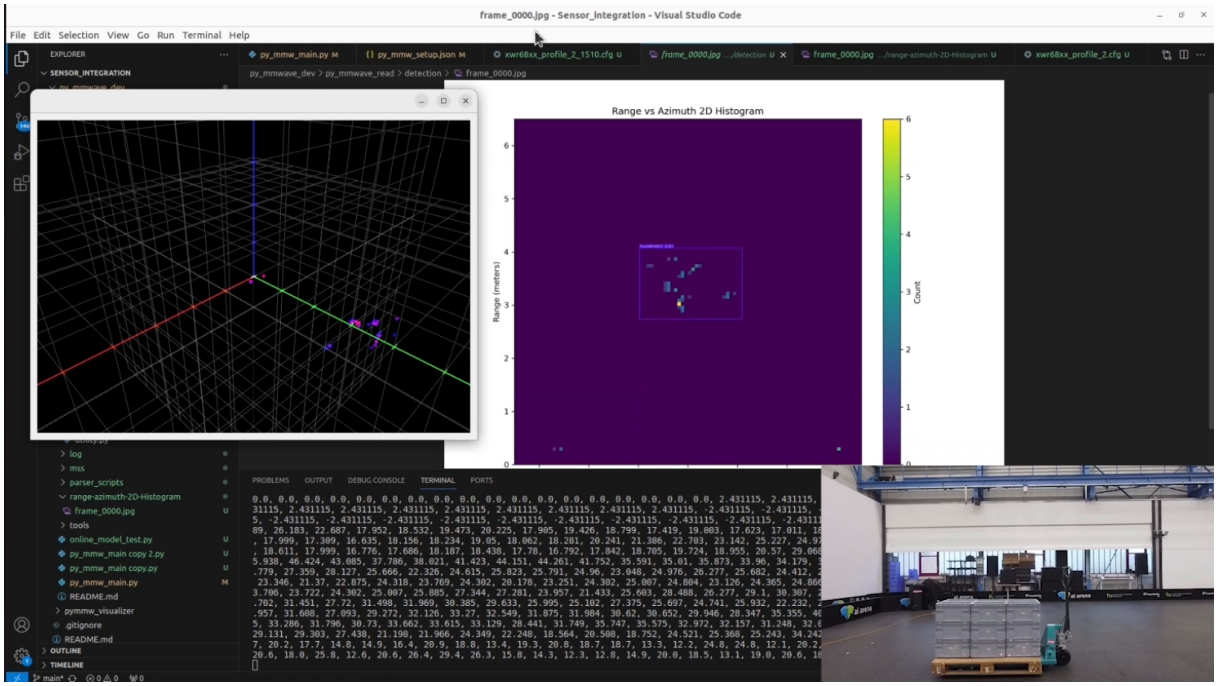


Figure 5.20: Examples of online model testing outputs, showing processed sensor data and detected objects. Live visual representation through the camera frame illustrates the real environment and the model's detection performance.

Figure 5.20 presents some examples of the outputs generated during online model testing. These examples include processed sensor data with corresponding object detection's. The live visual representation, obtained through a camera feed, provides an additional perspective for understanding the real environment and visualizing the model's detection capabilities. The results demonstrate that the model effectively detects objects in an indoor logistics environment, provided there is sufficient separation between objects. However, the model's detection accuracy decreases when multiple objects are in close proximity or collide with each other. This limitation highlights the challenges in complex scenarios where object overlap or occlusion occurs.

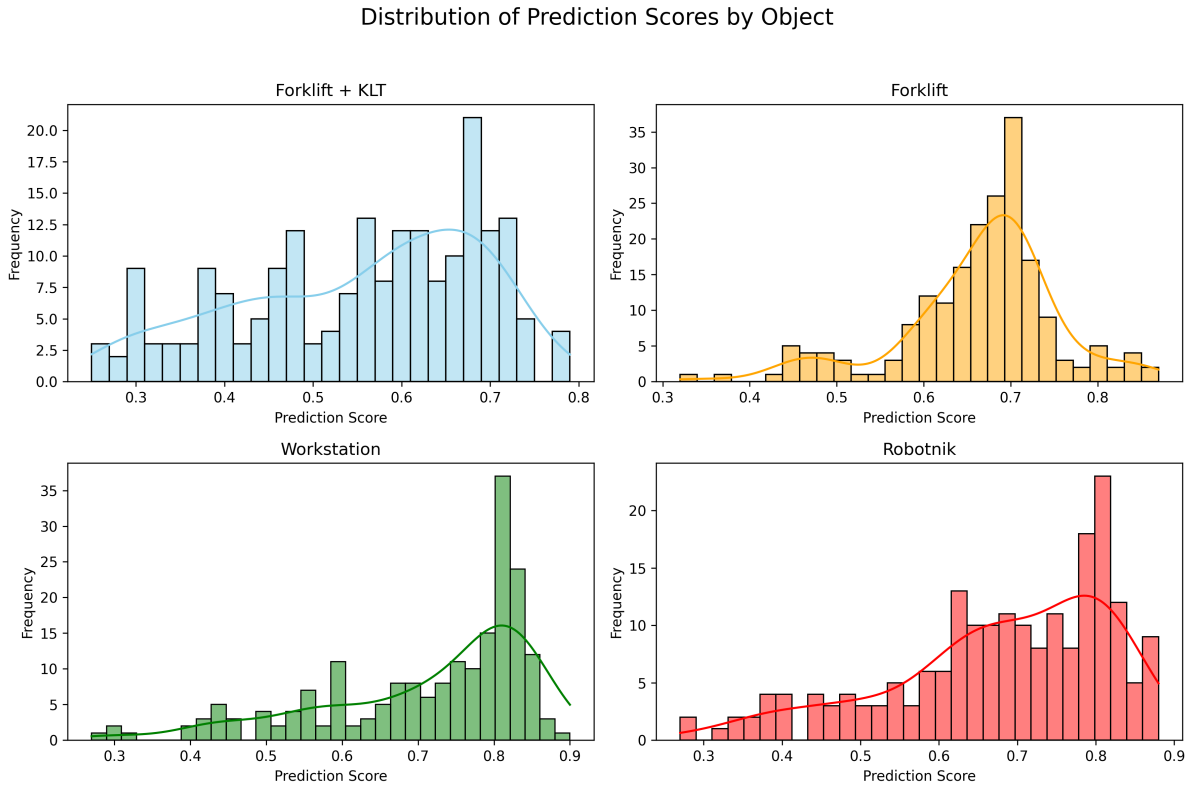


Figure 5.21: Distribution of prediction scores for the online object detection model across different objects.

The analysis of the prediction score distribution, as shown in Figure 5.21, highlights varying levels of confidence in the model's detections, which can be understood by considering the nature of radar signals. Radar detects objects based on how they reflect emitted signals, and the strength and clarity of these reflections significantly influence the model's ability to make accurate predictions. For example, the Forklift + KLT object has a higher concentration of prediction scores around 0.6 to 0.7, suggesting that the radar system receives strong and consistent reflections from this object. This could be due to its size or reflective surface, which results in clearer returns, leading to higher confidence in its detection. In contrast, the Workstation object shows a more spread-out distribution with frequencies peaking around 0.5, indicating that the radar signals reflecting off it are less consistent. This may be due to the Workstation's different materials, which cause weaker or more scattered reflections, making it harder for the model to achieve the same level of certainty in detection. Similarly, the Forklift object shows a concentration of scores around 0.7, reflecting higher confidence in its detection. The more reflective nature of the Forklift results in stronger and more reliable radar returns. On the other hand, the Robotnik object displays a broader range of scores, peaking around 0.4 to 0.5, suggesting that the radar receives more variable reflections. This could be due to the Robotnik's smaller size or

less reflective characteristics, leading to inconsistent returns and, consequently, less confidence in detection.

The model's prediction confidence is closely tied to the nature of the radar reflections, with objects like Forklift + KLT and Forklift being detected with higher confidence due to their stronger and more consistent radar returns. In contrast, objects like the Workstation and Robotnik generate weaker or less predictable radar signals, leading to lower and more varied confidence in detection. These patterns suggest the need for further model optimization to improve detection accuracy, particularly for objects with weaker or more scattered radar reflections. To address this challenge and enhance the model's performance in complex detection scenarios, additional training with data from mmWave radar sensors is proposed. Specifically, leveraging 3D point cloud data for object detection will provide more robust spatial information, helping to resolve object overlaps and improve overall detection accuracy and reliability.

5.4 Object Detection with 3D Point Cloud

The performance of range-azimuth-based object detection systems often degrades in scenarios involving multiple objects or when objects are colliding. To overcome this limitation, an alternative feature of the mmWave radar sensor, namely 3D point clouds, is explored for training an object detection model.

5.4.1 Offline Model Testing

Similar to the range-azimuth model, offline model testing involves evaluating the trained model on a pre-collected and pre-processed dataset, without real-time deployment. In this case, the OpenPCDet model with the Point Cloud Region-based Convolutional Neural Network (PCRCNN) backbone is selected for offline testing. OpenPCDet is an open-source, clear, and self-contained framework for 3D object detection, developed by the mmlab team [120]. The primary objective of this offline testing phase is to ensure seamless integration of the components in the testing pipeline, using a well-prepared dataset. The dataset is pre-processed and organized to align with the specific input requirements of the selected model. A baseline is established to verify the output using a motion capturing system (Vicon). A new step is added to the data preprocessing phase, which involves mapping the 3D coordinates of the radar sensor with the Vicon system. The offline model testing workflow, as illustrated in Figure 4.5, closely follows the range-azimuth offline model, with the only difference being the mapping of coordinates.

The use of the OpenPCDet model for processing 3D point clouds requires careful configuration to ensure the model trains effectively. Two critical aspects are data configuration and model configuration. In the data configuration, the number of classes, the names of the classes, and the number of features for each data point are specified. In this case, four classes are used: Forklift+KLT, Forklift, Workstation, and Robotnik. The features of the data are the 3D coordinates (x, y, z) and the intensity values. Additionally, the range for each of the coordinates (x, y, z) is defined, and appropriate voxel sizes are selected based on the defined ranges. The voxel size is an important parameter that influences the resolution of the model in 3D space. According to the guidelines provided by mmlab, the range and voxel size should follow these

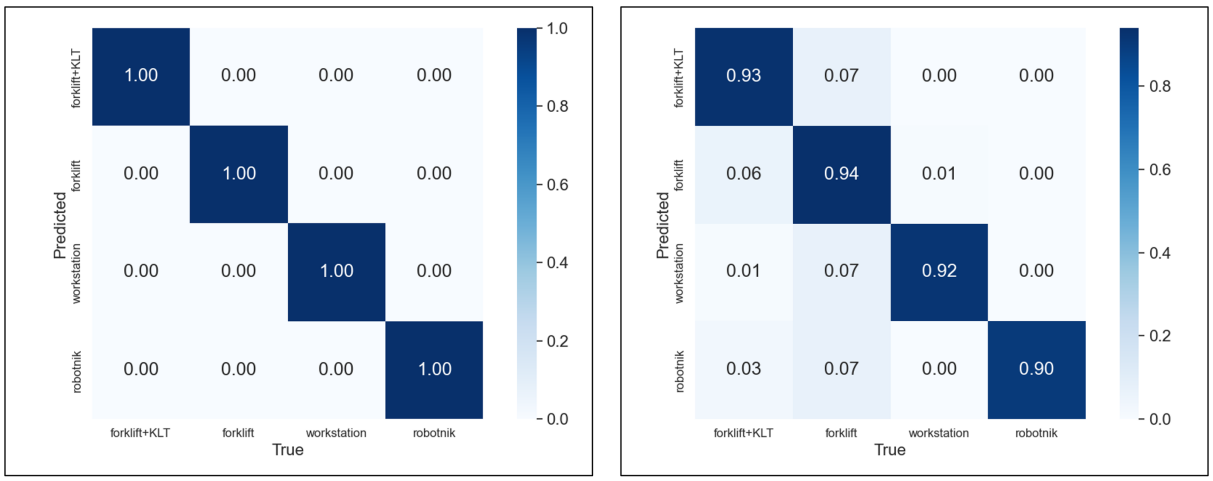
rules for voxel-based detectors like Sparse Encoded ConvNet Detector (SECOND), PointVoxel Region-based Convolutional Neural Network (PV-RCNN), and CenterPoint:

- The point cloud range along the z-axis divided by the voxel size should be 40.
- The point cloud range along the x- and y-axes divided by the voxel size should be a multiple of 16.

In the model configuration, the anchor size for each class is specified. Anchor size represents the dimensions of the anchor boxes used for object detection and must be adapted to the custom dataset. After completing the data and model configurations, the data information is generated by running a command from the repository. For evaluation, the classes of the KITTI dataset are mapped to the custom dataset. The training process begins after these configurations, using 100 epochs with a batch size of 2, which is chosen based on the hardware specifications of the system.

5.4.2 Analysis and Results for Offline Model Testing: 3D Point Cloud

The model is successfully trained and evaluated using both static and dynamic datasets collected from the logistics scenario. The static dataset, where the objects do not move, yields better results than the dynamic dataset, which contains moving objects. The training of the static dataset takes approximately 31 hours, 24 minutes, and 7 seconds, whereas training with the dynamic dataset completes in about 24 hours, 19 minutes, and 54 seconds. After evaluation, the model's performance is analyzed, including relative performance metrics and performance graphs. The Open3D library is used to visualize the detection results in a 3D environment.



(a) Confusion matrix of static model **(b) Confusion matrix of dynamic model**

Figure 5.22: Confusion matrix for static and dynamic dataset-trained models evaluated on test data.

The confusion matrix provides valuable insights by comparing the ground truth with the predicted results for each class, revealing any overlap or misclassification. The confusion matrix for the static model, shown in Figure 5.22, indicates that the model is able to detect all four

object classes correctly, as reflected by the diagonal entries being 1.0 and the off-diagonal entries being 0.0. This suggests that the static model achieves perfect detection for all objects. In contrast, the confusion matrix for the dynamic model, also shown in Figure 5.22, reveals that while the model achieves good results, there are some misclassifications, particularly with the forklift class. The confusion matrix for the dynamic model shows diagonal values around 0.9, indicating that the model is able to correctly classify most of the objects, with some confusion between the forklift and other classes. Notably, the Robotnik class has zero confusion with the other classes, showing perfect detection.

Key performance metrics for evaluating 3D object detection models include:

- **AP:** Average Precision, a measure of overall detection accuracy.
- **BBox AP:** Accuracy of bounding boxes around detected objects.
- **BEV AP:** Accuracy in the Bird's Eye View (top-down) representation.
- **3D AP:** Accuracy in three-dimensional space.
- **AOS AP:** Accuracy of Angular Overlap Score (measuring object rotation).
- **IoU:** Intersection over Union, the threshold for overlap between detected and ground truth objects.

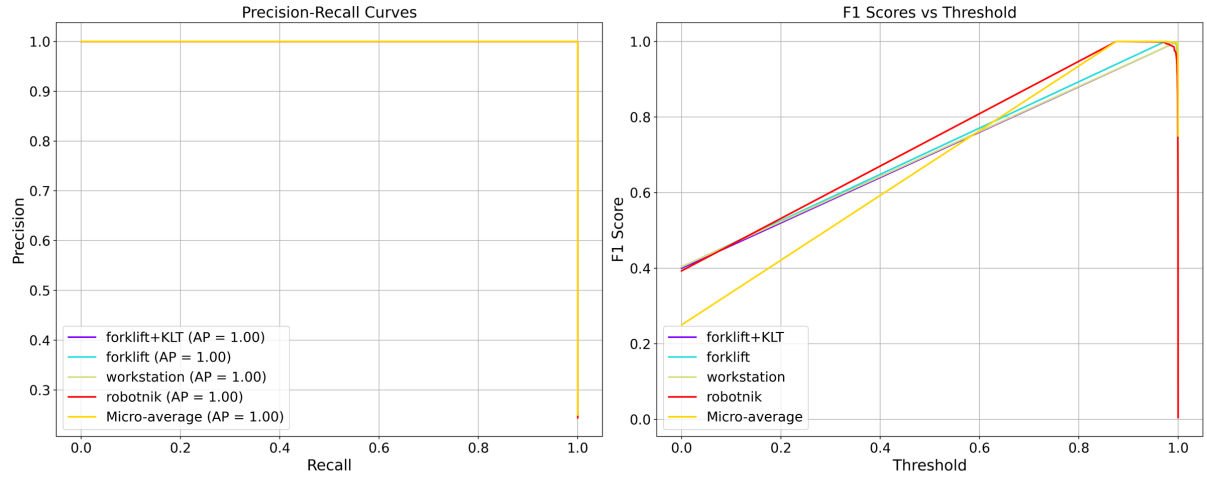
Metric	**Static Model**	**Dynamic Model**
Recall (ROI @ 0.3)	1.000000	0.995678
Recall (RCNN @ 0.3)	1.000000	0.996543
Recall (ROI @ 0.5)	1.000000	0.971478
Recall (RCNN @ 0.5)	1.000000	0.974071
Recall (ROI @ 0.7)	0.999398	0.649957
Recall (RCNN @ 0.7)	1.000000	0.732066
Average Predicted Objects	1.000	2.032
Forklift+KLT AP (0.70, 0.70)	100.0000 (bbox, bev, 3d)	71.2638 (bbox), 5.6099 (bev), 4.9121 (3d)
Forklift AP (0.50, 0.50)	100.0000 (bbox, bev, 3d)	30.4272 (bbox), 19.9479 (bev), 15.3246 (3d)
Workstation AP (0.50, 0.50)	100.0000 (bbox, bev, 3d)	88.7417 (bbox), 71.2633 (bev), 63.5486 (3d)
Robotnik AP (0.70, 0.70)	100.0000 (bbox, bev, 3d)	77.0608 (bbox), 17.5822 (bev), 7.9154 (3d)

Table 5.3: Comparison of key metrics for static and dynamic models.

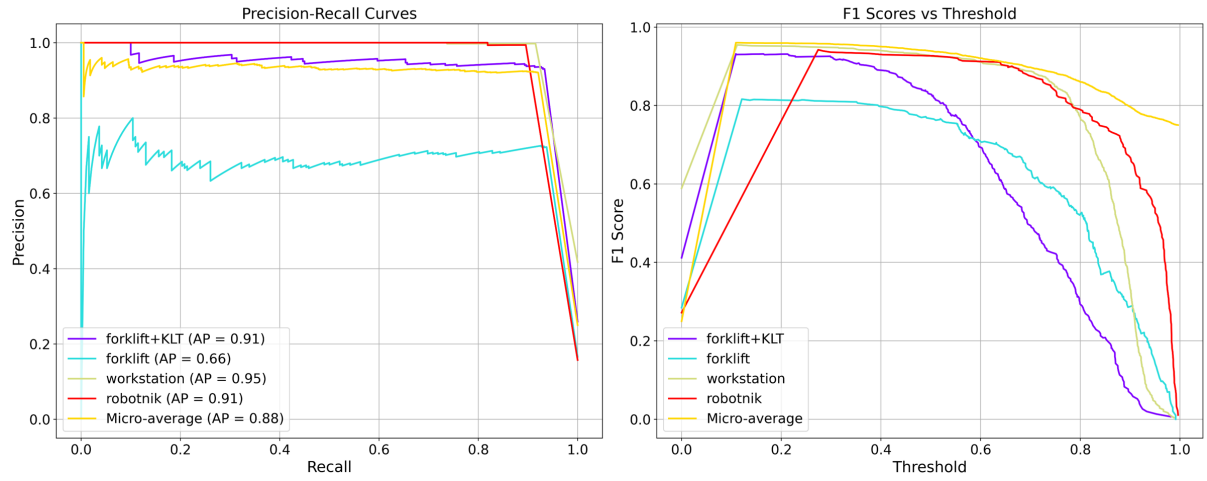
The IoU threshold determines the degree of overlap required for a prediction to be considered correct. In the static model, the system processes each example in 0.0721 seconds, demonstrating high efficiency. The recall values for detecting objects are nearly perfect, with most values reaching 1.0. Forklift and Workstation detections achieve 100% accuracy for all metrics (Bounding Box, BEV, and 3D). Robotnik detection shows slightly lower accuracy in AOS, with scores ranging from 99.98% to 99.99%. On the other hand, the dynamic model processes each example in 0.0737 seconds, also showing good efficiency. At lower IoU thresholds (0.3 and 0.5), both the Region-of-Interest (Region of Interest (ROI)) and RCNN methods achieve high recall values, detecting most objects. However, at higher IoU (0.7), performance drops significantly,

with recall values of 0.650 (ROI) and 0.732 (RCNN). The Forklift and Forklift+KLT models struggle at higher IoU thresholds, particularly in BEV and 3D performance. The best performance is observed at IoU 0.5, where the 3D AP reaches approximately 29.09. The Workstation and Robotnik classes perform well across all thresholds, with 3D AP values ranging from 63.55 to 76.10. Detailed information are given in the Table 5.3.

The Precision-Recall and F1 score curves further corroborate these findings. The static model shows a perfect precision-recall curve with an AP value of 1.00 for all objects, resulting in a micro-average of 1.0. In contrast, the dynamic model exhibits some fluctuations, with lower precision for the Forklift class (AP = 0.66) and higher precision for other objects (AP around 0.9). The F1 score curve also shows similar trends, with the static model displaying minimal fluctuations, while the dynamic model exhibits some variability. The Precision-Recall and F1 curves for both models are shown in Figure 5.23.



(a) PR and F1 curve of static



(b) PR and F1 curve of dynamic model

Figure 5.23: Precision-Recall and F1 curves for static and dynamic models.

Using the Open3D library, the model's predictions are visualized in a 3D environment. Figure 5.24 illustrates the detection of the Forklift and Workstation, while Figure 5.25 shows the prediction of Forklift+KLT and Robotnik. To provide a better context for the detection, a red-bordered rectangle represents the indoor logistics hall, with the origin at the center, corresponding to the Vicon motion detection system. Another origin, representing the radar sensor, is

placed slightly off-center. The detected bounding boxes are shown in green, and corresponding camera frames are presented for comparison. The visualizations demonstrate that the model can accurately detect objects using the 3D point cloud feature of the mmWave radar sensor.

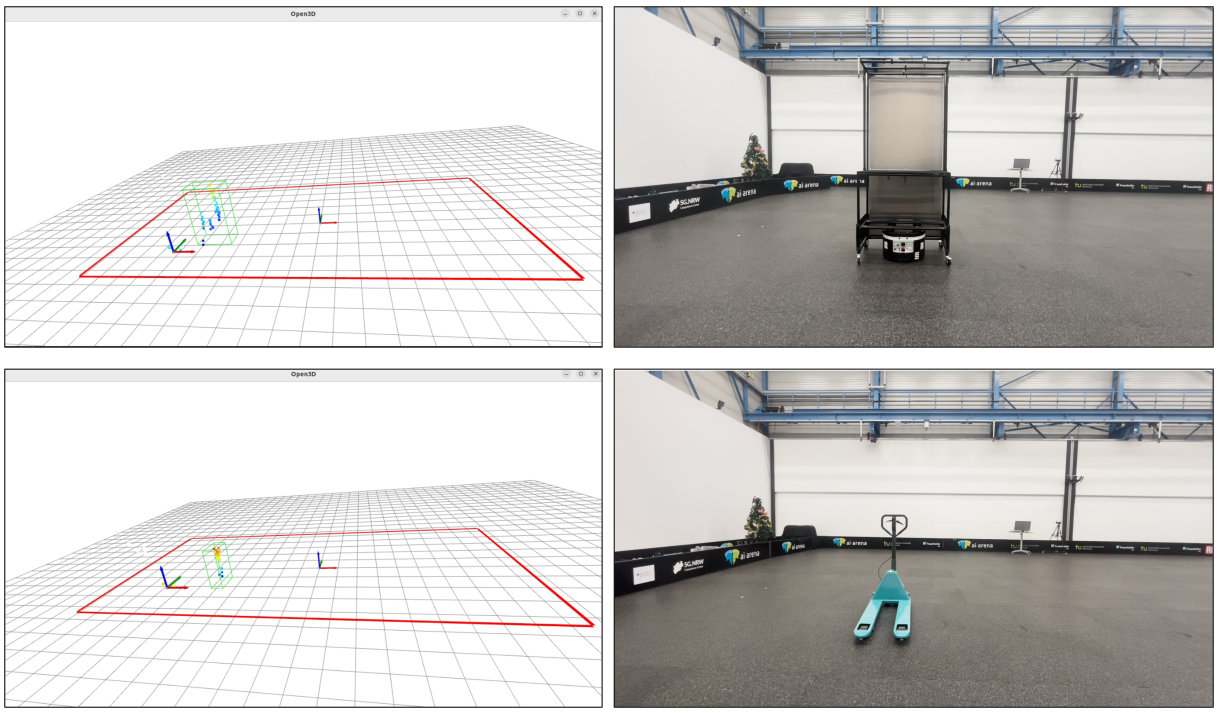


Figure 5.24: 3D point cloud model detection output for Forklift and Workstation.

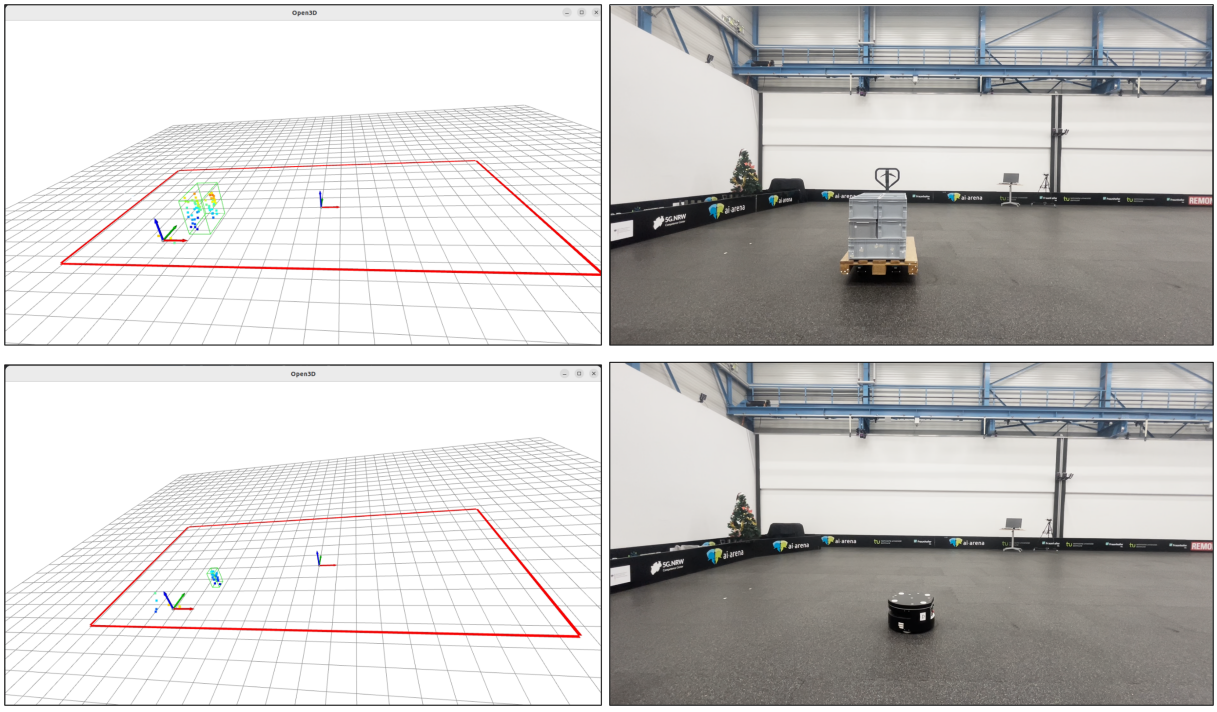


Figure 5.25: 3D point cloud model detection output for Forklift+KLT and Robotnik.

5.4.3 Result Comparison with existing motion detection system

In this experiment, the model's performance is compared with the baseline Vicon motion detection system, which provides the x, y, and z coordinates of detected objects. After being trained on 3D point cloud data, the radar sensor predicts the coordinates of the center of the bounding box for each detected object. The comparison is shown in Figures 5.26 and 5.27. These figures demonstrate the difference in accuracy between the Vicon system and the radar sensor predictions. In these figures, classes 1, 2, 3, and 4 represent the objects Forklift with KLT, Forklift, Workstation, and Robotnik, respectively.

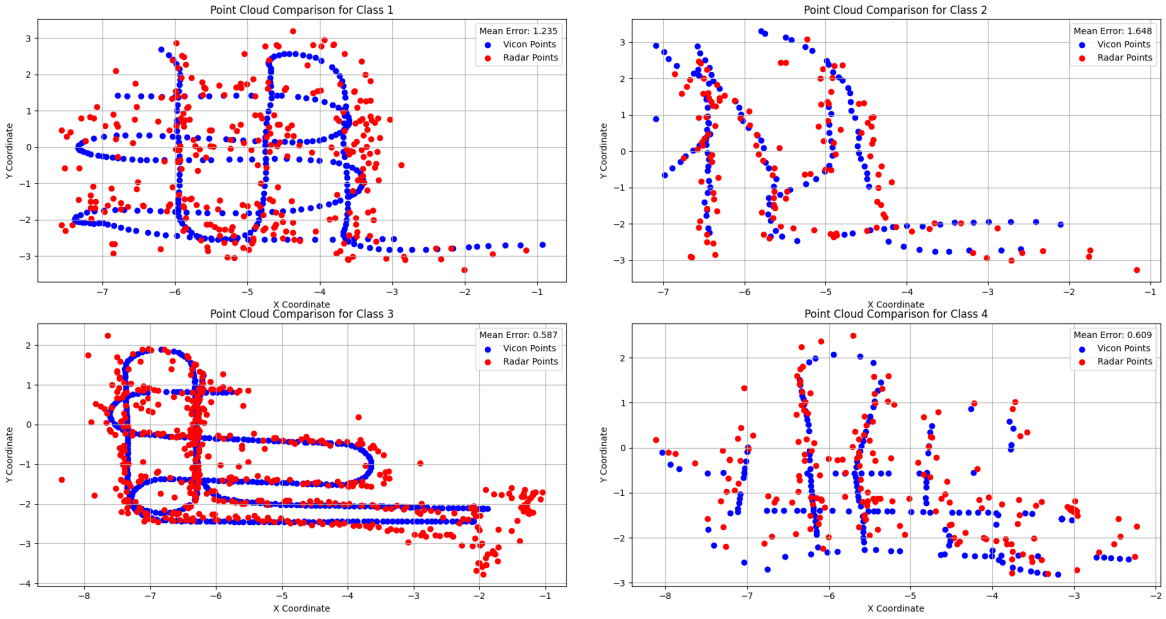


Figure 5.26: Comparison of radar and Vicon system detections in 2D (x, y coordinates).

Figure 5.26 shows the comparison of the x and y coordinates, where the red dots represent the radar sensor coordinates and the blue dots represent the Vicon coordinates. The error is minimal for the Workstation and Robotnik objects, with mean errors of 0.587 and 0.609, respectively. This suggests that the model performs well in detecting these objects. However, the Forklift and Forklift+KLT objects show higher mean errors, 1.648 and 1.235, respectively. This discrepancy can be attributed to the fact that the Vicon system detects points at the center of the objects for Workstation and Robotnik, while for Forklift and Forklift+KLT, the detection points are not at the center. Nevertheless, the radar sensor consistently detects objects at the center of their bounding boxes, and the model performs reasonably well overall.

Additionally, The 3D comparison of the detected points is illustrated in Figure 5.27, highlighting the consistency and accuracy of the model's predictions relative to the Vicon system. The 3D representation also provides additional insight into how the radar sensor tracks objects in three-dimensional space, capturing not only their position but also their orientation. While the Vicon system offers precise localization, the radar sensor's 3D predictions demonstrate good alignment, especially for larger, more reflective objects, validating the model's effectiveness in real-world applications.

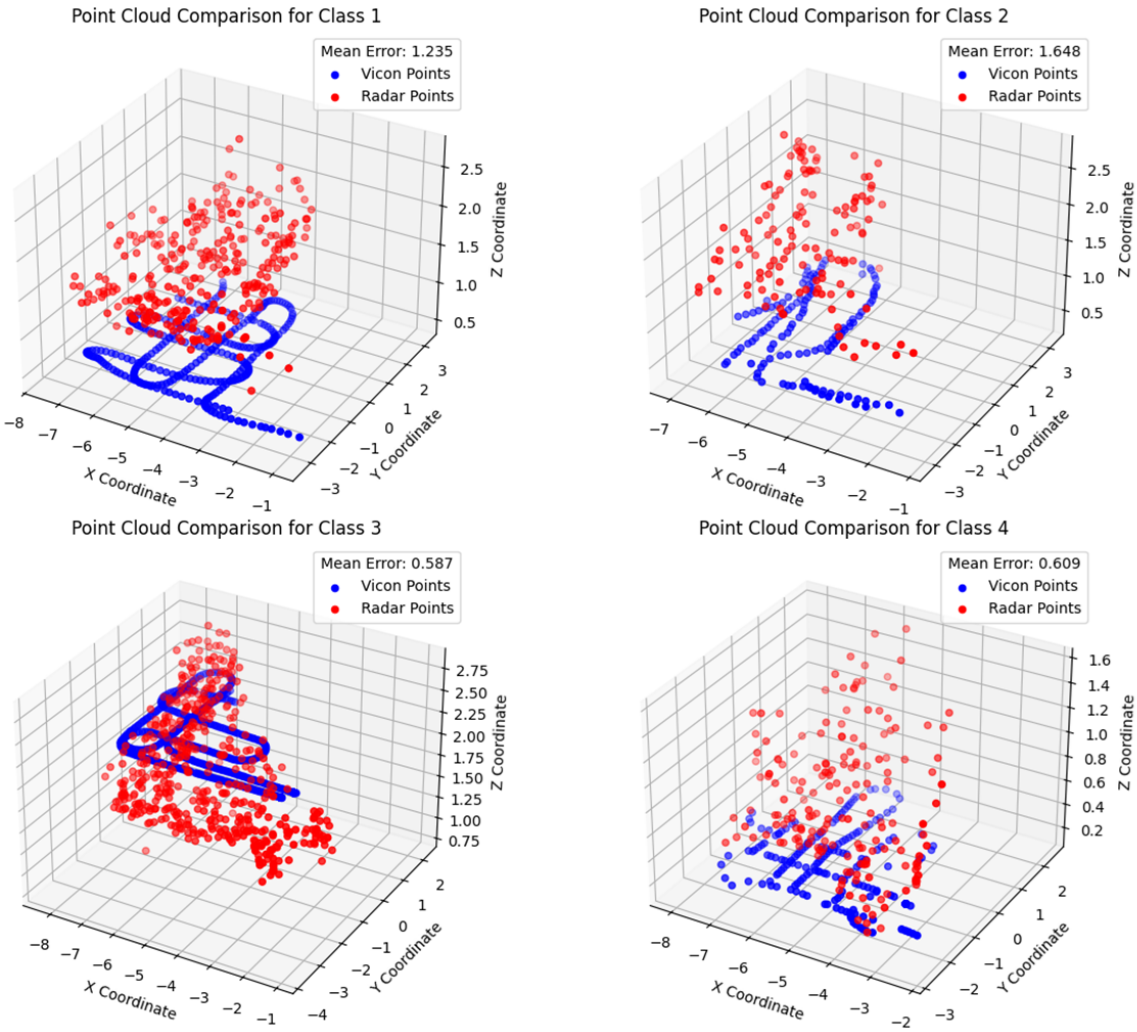


Figure 5.27: 3D comparison of radar and Vicon system detections.

5.5 Algorithms and Datasets

The algorithms and datasets utilized in the studies presented in this thesis are accessible at the following locations:

- **Sciebo Repository**: https://tu-dortmund.sciebo.de/f/562679402
- **GitHub Repository**: https://github.com/sahilrajpurkar03/Master-Thesis.git
- **Kaggle Repository**:
  - **Arranged Dataset**: https://www.kaggle.com/mmwave-radar-dataset
  - **Raw Dataset**: https://www.kaggle.com/mmwave-radar-dataset-raw

6 Conclusion and Future Work

6.1 Conclusion

This research provides an in-depth exploration of object detection for logistic entities using mmWave radar sensors, focusing on range-azimuth and 3D point cloud data to enhance detection in indoor logistics environments. By leveraging frameworks such as YOLOv7, Detectron2, and OpenPCdet, the study develops robust offline and online models tailored for precise feature extraction in machine learning tasks, addressing key challenges including scalability, communication robustness, and real-time implementation. The results demonstrate that leveraging radar-based range-azimuth data significantly enhances object detection performance in indoor logistics environments. The inclusion of advanced preprocessing techniques and robust model configurations, such as YOLOv7 with a CNN backbone, yields tangible benefits. These include improved detection accuracy, adaptability across static and dynamic scenarios, and reliable performance in challenging environments where object occlusion and sparse data points are critical factors. Comparative analysis of backbone architectures revealed that CNN outperforms alternatives like Faster R-CNN and RetinaNet, achieving higher precision and robustness in grid-like logistic setups. Although dynamic datasets posed challenges, particularly for smaller objects like Robotnik, the models showcased consistent performance, demonstrating their scalability and applicability in diverse scenarios. The structured methodology employed in offline and online testing highlights the model's capability to balance computational efficiency, detection accuracy, and adaptability, cementing its utility for real-time logistic operations.

The investigation into 3D point cloud data as a complementary approach to range-azimuth-based object detection demonstrated significant improvements in handling complex scenarios such as multi-object interactions and collisions. Leveraging the OpenPCDet framework with a Point Cloud Region-based Convolutional Neural Network (PCRCNN) backbone, the study achieved robust offline model performance, validated through precise coordinate mapping with the Vicon motion capture system. The model, trained on both static and dynamic datasets, exhibited exceptional accuracy in static environments, achieving perfect detection with no misclassifications, as evidenced by the confusion matrix. However, dynamic scenarios introduced challenges, particularly for the Forklift class, where movement led to occasional misclassifications. Performance metrics, including Average Precision (AP), Bounding Box AP (BBox AP), Bird's Eye View AP (BEV AP), and 3D AP, underscored the model's efficiency, with static datasets processed in 0.0721 seconds per example and dynamic datasets in 0.0737 seconds. Visualization using the Open3D library confirmed the model's ability to accurately detect and localize objects in three-dimensional space. Comparative analysis with the Vicon system revealed minimal errors for Workstation and Robotnik objects (mean errors of 0.587 and 0.609, respectively), while higher discrepancies were observed for Forklift and Forklift+KLT (mean

errors of 1.648 and 1.235, respectively), attributed to differences in detection point locations. These findings highlight the effectiveness of 3D point cloud data in enhancing object detection accuracy and robustness, particularly in complex indoor logistics environments. The integration of 3D point cloud processing with mmWave radar sensors thus represents a significant advancement, offering a reliable and scalable solution for real-time object detection in dynamic settings.

6.2 Future Work

This study focuses on object detection for logistic entities in an indoor logistics environment using mmWave radar sensors. The object detection process is based on the output data from mmWave radar sensors, specifically leveraging range-azimuth information and 3D point cloud data. However, there are several avenues for improvement and expansion of this research, contributing to advancements in 6G sensing technologies. To enhance model prediction accuracy, additional output parameters from the mmWave radar sensor, such as velocities and Signal-to-Noise Ratio (SNR), can be utilized. A potential approach could involve employing separate models for each output parameter and running these models in parallel. By integrating their predictions using robust decision-making algorithms, the overall prediction performance could be significantly improved. Currently, the study is limited to a single sensor, which restricts coverage to a specific area. Incorporating multiple sensors and employing stitching algorithms to merge their data could expand the coverage to the entire arena. This approach would also improve the accuracy, potentially matching the precision levels of systems like the Vicon motion sensing system.

The 3D point cloud data obtained from the current mmWave radar sensor is not highly dense. Using an industrial-grade mmWave radar sensor with better resolution, coupled with optimized configuration and calibration settings, could produce denser 3D point clouds. This advancement could transition the research focus from object detection to object segmentation, enabling more granular and detailed analysis. Additionally, versatility of the mmWave radar sensor allows for various applications beyond static detection. For example, mounting the sensor on mobile robots would enable these robots to sense objects in real time, facilitating effective object avoidance and enhancing their autonomous navigation capabilities. Furthermore, integrating these sensors with drone systems could enable aerial monitoring and object detection in warehouses or manufacturing environments. They could also be used in collaborative robotic systems for dynamic task allocation, where real-time sensing of object locations is crucial.

Overall, these enhancements and applications could contribute to the development of more robust, accurate, and versatile systems for indoor logistics and other domains reliant on advanced sensing technologies.

Bibliography

[1] Bárbara Ferreira and João Reis. "A Systematic Literature Review on the Application of Automation in Logistics". In: *Logistics* 7.4 (2023). ISSN: 2305-6290. DOI: 10.3390/logistics7040080. URL: https://www.mdpi.com/2305-6290/7/4/80.

[2] Jinhe Wang, Nan Zhang, and Qinggang He. "Application of Automated Warehouse Logistics in Manufacturing Industry". In: *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009* 4 (Aug. 2009). DOI: 10.1109/CCCM.2009.5267696.

[3] Mathias Rieder and Marius Breitmayer. "Object Detection in Picking: Handling variety of a warehouse's articles". In: Sept. 2022. DOI: 10.15480/882.4688.

[4] Harry Mafukidze et al. "Scattering Centers to Point Clouds: A Review of mmWave Radars for Non-Radar-Engineers". In: *IEEE Access* PP (Jan. 2022), pp. 1–1. DOI: 10.1109/ACCESS.2022.3211673.

[5] Arthur Venon et al. "Millimeter Wave FMCW RADARs for Perception, Recognition and Localization in Automotive Applications: A Survey". In: *IEEE Transactions on Intelligent Vehicles* 7 (Sept. 2022), pp. 1–1. DOI: 10.1109/TIV.2022.3167733.

[6] Fan Liu et al. "Integrated Sensing and Communications: Toward Dual-Functional Wireless Networks for 6G and Beyond". In: *IEEE Journal on Selected Areas in Communications* 40 (Mar. 2022), pp. 1–1. DOI: 10.1109/JSAC.2022.3156632.

[7] Anish Shastri et al. "A Review of Millimeter Wave Device-Based Localization and Device-Free Sensing Technologies and Applications". In: *IEEE Communications Surveys Tutorials* 24.3 (2022), pp. 1708–1749. DOI: 10.1109/COMST.2022.3177305.

[8] J. Zhang et al. "A Survey of mmWave-Based Human Sensing: Technology, Platforms and Applications". In: *IEEE Communications Surveys & Tutorials* 25.4 (2023), pp. 2052–2087. DOI: 10.1109/COMST.2023.3298300.

[9] Arsalan Haider et al. "A Methodology to Model the Rain and Fog Effect on the Performance of Automotive LiDAR Sensors". In: *Sensors* 23 (2023), p. 6891. DOI: 10.3390/s23156891.

[10] Ju Wang et al. "Low Human-Effort, Device-Free Localization with Fine-Grained Subcarrier Information". In: *IEEE Transactions on Mobile Computing* (2018). DOI: 10.1109/TMC.2018.2812746.

[11] A. Alarifi et al. "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances". In: *Sensors* 16 (2016), p. 707. DOI: 10.3390/s16050707.

[12] Phui San Cheong et al. "Comparison of LoRaWAN Classes and their Power Consumption". In: *Proceedings of the 2017 Symposium on Communications and Vehicular Technology (SCVT)*. 2017. DOI: 10.1109/SCVT.2017.8240313.

[13] Patrick Palmer et al. “Multi-Object Tracking based on Imaging Radar 3D Object Detection”. In: *IEEE Transactions on Intelligent Vehicles* (2024). to be published.

[14] Maaz Awan et al. “Towards mmWave Altimetry for UAS: Exploring the Potential of 77 GHz Automotive Radars”. In: *Drones* 8 (2024), p. 94. DOI: 10.3390/drones8030094.

[15] Irfan Priyanta et al. “Towards 6G-Driven Digital Continuum in Logistics”. In: (2023). URL: 10.2195/lj_proc_priyanta_en_202310_01.

[16] Xinran Fang et al. “Joint Communication and Sensing Toward 6G: Models and Potential of Using MIMO”. In: *IEEE Internet of Things Journal* (2022). DOI: 10.1109/JIOT.2022.3227215.

[17] M. I. Skolnik. *Radar Handbook*. 3rd. McGraw-Hill Education, 2008.

[18] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[19] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[20] Snehal Buche. “Understanding mmWave RADAR, its Principle & Applications”. In: *eInfochips* (2022). URL: https://www.design-reuse.com/articles/55851/mmwave-radar-principle-applications.html.

[21] SELECTION GUIDE. *How Accurate are mmWave Sensors?* 2022. URL: https://www.dfrobot.com/blog-1654.html.

[22] Serena Liang. *What is mmWave Radar? Everything You Need to Know About FMCW*. 2022. URL: https://www.seeedstudio.com/blog/2022/01/03/mmwave-radar-sensing-fmcw-radar/.

[23] Akanksha Bhutani et al. “The Role of Millimeter-Waves in the Distance Measurement Accuracy of an FMCW Radar Sensor”. In: *Sensors* 19.3938 (2019). DOI: 10.3390/s19183938.

[24] S. Scherr et al. “An Efficient Frequency and Phase Estimation Algorithm With CRB Performance for FMCW Radar Applications”. In: *IEEE Transactions on Instrumentation and Measurement* 64 (2015), pp. 1868–1875.

[25] S. Ayhan et al. “FMCW radar system with additional phase evaluation for high accuracy range detection”. In: *Proceedings of the 8th European Radar Conference*. Manchester, UK, 2011, pp. 117–120.

[26] Ömer Başpınar, Berk Omuz, and Ahmet Oncu. “Detection of the Altitude and On-the-Ground Objects Using 77-GHz FMCW Radar Onboard Small Drones”. In: *Drones* 7 (Jan. 2023), p. 86. DOI: 10.3390/drones7020086.

[27] Texas Instruments. *IWR6843, IWR6443 Single-Chip 60- to 64-GHz mmWave Sensor*. SWRS219E. June 2021. URL: https://www.ti.com/lit/ds/symlink/iwr6843.pdf.

[28] Texas Instruments. *60GHz mmWave Sensor EVMs User’s Guide*. SWRU546E. May 2022. URL: https://www.ti.com/lit/ug/swru546e/swru546e.pdf.

[29] Fahad Jibrin Abdu et al. “Application of Deep Learning on Millimeter-Wave Radar Signals: A Review”. In: *Sensors* 21.6 (2021). ISSN: 1424-8220. DOI: 10.3390/s21061951. URL: https://www.mdpi.com/1424-8220/21/6/1951.

[30] Shanliang Yao et al. *Exploring Radar Data Representations in Autonomous Driving: A Comprehensive Review*. 2024. arXiv: 2312.04861 [cs.CV]. URL: https://arxiv.org/abs/2312.04861.

[31] E. Gambi et al. "Millimeter Wave Radar data of people walking". In: *Data in Brief* 31 (July 2020), p. 105996. DOI: 10.1016/j.dib.2020.105996.

[32] Yueheng Li et al. "User Detection in RIS-based mmWave JCAS: Concept and Demonstration". In: *IEEE Transactions on Wireless Communications* (2024), pp. 1–1. DOI: 10.1109/TWC.2024.3364048.

[33] Radosław Maksymiuk. "Emerging Trends in Radar: 5G/6G Joint Communication and Active & Passive Sensing". In: *IEEE Aerospace and Electronic Systems Magazine* (2025), pp. 1–4. DOI: 10.1109/MAES.2025.3533943.

[34] F. Liu et al. "Integrated Sensing and Communications: Toward Dual-Functional Wireless Networks for 6G and Beyond". In: *IEEE Journal on Selected Areas in Communications* 40.6 (2022), pp. 1728–1767. DOI: 10.1109/JSAC.2022.

[35] Ruixing Ren. *Integrated Sensing, Communication, and Computation for IoV Towards 6G*. 2025. DOI: 10.13140/RG.2.2.15634.57285.

[36] Taixia Shi, Yang Chen, and Jianping Yao. "Seamlessly Merging Radar Ranging and Imaging, Wireless Communications, and Spectrum Sensing for 6G Empowered by Microwave Photonics". In: *Communications Engineering* 3 (2024). DOI: 10.1038/s44172-024-00279-0.

[37] Ran Tao. "Sensor Fusion Research in Autonomous Driving Systems Based on Radar and Cameras". In: *Highlights in Science, Engineering and Technology* 119 (2024), pp. 78–84. DOI: 10.54097/srbaft85.

[38] Ms. Sonali, B. Maind, and Priyanka Wankar. "Research Paper on Basic of Artificial Neural Network". In: 2014. URL: https://api.semanticscholar.org/CorpusID:267819742.

[39] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91. URL: https://doi.org/10.1109/CVPR.2016.91.

[40] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324. URL: https://doi.org/10.1109/ICCV.2017.324.

[41] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017. URL: https://papers.nips.cc/paper/2015/hash/14b5d38db9445547d90a9c6eec9b5b6e-Abstract.html.

[42] Zhiqiang Zhao et al. "Object Detection in Real-Time: A Survey". In: *Sensors* 19.9 (2019), p. 2056. DOI: 10.3390/s19092056. URL: https://doi.org/10.3390/s19092056.

[43] Gerard Salton. *Introduction to Modern Information Retrieval*. 1st. New York: McGraw-Hill, 1986. ISBN: 978-0070544840.

[44] Wikipedia contributors. *Supervised learning*. Accessed: 2024-11-26. 2024. URL: https://en.wikipedia.org/wiki/Supervised_learning.

[45] Venkatesh Mane et al. “RADAR and Camera Sensor Data Fusion”. In: *Information and Communication Technology for Competitive Strategies (ICTCS 2021)*. Ed. by Amit Joshi, Mufti Mahmud, and Roshan G. Ragel. Singapore: Springer Nature Singapore, 2023, pp. 791–801.

[46] J. Chen. “Visual and Radar Data Fusion: Complementary or Alternative Sensors in Intelligent Driving Systems”. In: *Applied and Computational Engineering* 93 (2024), pp. 61–67.

[47] Ramin Nabati and Hairong Qi. “Radar-Camera Sensor Fusion for Joint Object Detection and Distance Estimation in Autonomous Vehicles”. In: *arXiv preprint arXiv:2009.08428* (2020). DOI: 10.48550/arXiv.2009.08428. URL: https://arxiv.org/abs/2009.08428.

[48] Ericsson. “6G: Connecting the physical, digital and human worlds”. In: *Ericsson White Paper* (2023).

[49] Ruixing Ren. “Integrated Sensing, Communication, and Computation for IoV Towards 6G”. In: (2025). DOI: 10.13140/RG.2.2.15634.57285.

[50] IHP - Innovations for High Performance Microelectronics. “6G Sensing: Opportunities and Challenges”. In: *IHP Technical Report* (2024).

[51] Qualcomm. “The Future of 6G: Sensing and Positioning”. In: *Qualcomm Research* (2024).

[52] Reshma Mathews et al. “Survey on Warehouse Monitoring and Management using AI”. In: *International Journal of Advances in Engineering and Management* (2024), pp. 391–397. DOI: 10.35629/5252-0611391397.

[53] D Subashree, Shrushti Rohidas Mhaske, and Sonal Rajesh Yeshwantrao Ayush Kumar. “Real Time Crowd Counting using OpenCV”. In: *International Journal of Engineering Research Technology* 10 (2021).

[54] Rong Lin. “Research into the Automatic Guidance System for AGVs Used for Logistics Based on Millimeter Wave Radar Imaging”. In: *Wireless Communications and Mobile Computing* 2022 (2022), pp. 1–8. DOI: 10.1155/2022/3104017.

[55] Patrick Palmer et al. *Multi-Object Tracking based on Imaging Radar 3D Object Detection*. 2024. arXiv: 2406.01011 [cs.RO].

[56] Shijie Liu et al. “Data Fusion of Roadside Camera, LiDAR and Millimeter Wave Radar”. In: *IEEE Sensors Journal* (2024). DOI: 10.1109/JSEN.2024.3448428.

[57] Weichen Tang. “The Application of FMCW Radar Technology in Human Heartbeat Respiration Monitoring”. In: *Highlights in Science, Engineering and Technology* 119 (2024), pp. 249–253. DOI: 10.54097/tjz8gp64.

[58] Hua-mei Chen et al. “Sensor Fusion Algorithms and Performance Limits”. In: 101 (2001).

[59] Yiyi Cai et al. “Intelligent Systems in Motion: A Comprehensive Review on Multi-Sensor Fusion and Information Processing From Sensing to Navigation in Path Planning”. In: *International Journal on Semantic Web and Information Systems* 19 (2023), pp. 1–35. DOI: 10.4018/IJSWIS.333056.

[60] Shitharth et al. “Testing of Emerging Wireless Sensor Networks Using Radar Signals With Machine Learning Algorithms”. In: *IEEE Journal of Selected Areas in Sensors* (2024), pp. 1–11. DOI: 10.1109/JSAS.2024.3395578.

[61] Anas Amaireh. “Improving Radar Sensing Capabilities and Data Quality Through Machine Learning”. In: (2024).

[62] Jurgen Hasch. “Driving towards 2020: Automotive radar technology trends”. In: *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. 2015, pp. 1–4. DOI: 10.1109/ICMIM.2015.7117956.

[63] Keegan Garcia, Mingjian Yan, and Alek Purkovic. *Robust traffic and intersection monitoring using millimeter wave sensors*. White Paper. Texas Instruments, 2018. URL: https://www.ti.com/lit/wp/spyy002b/spyy002b.pdf?ts=1728220367566.

[64] Jaime Lien et al. “Soli: Ubiquitous Gesture Sensing with Millimeter Wave Radar”. In: *ACM Transactions on Graphics* 35 (July 2016), pp. 1–19. DOI: 10.1145/2897824.2925953.

[65] Sujeet Milind Patole et al. “Automotive radars: A review of signal processing techniques”. In: *IEEE Signal Processing Magazine* 34.2 (2017), pp. 22–35. DOI: 10.1109/MSP.2016.2628914.

[66] National Telecommunications and Information Administration. “6G and Advanced Sensing Technologies for Smart Logistics”. In: *NTIA Technical Report* (2024).

[67] Nokia. “The Future of Logistics: 6G and Advanced Radar Sensing”. In: *Nokia Bell Labs Technical Journal* (2023).

[68] Martin Vossiek et al. “Wireless local positioning”. In: *Microwave Magazine, IEEE* 4 (Jan. 2004), pp. 77–86. DOI: 10.1109/MMW.2003.1266069.

[69] Juergen Dickmann et al. “”Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding””. In: *2016 IEEE Radar Conference (RadarConf)*. 2016, pp. 1–6. DOI: 10.1109/RADAR.2016.7485214.

[70] Bruno Neri. “Radar Sensor Signal Acquisition and Multidimensional FFT Processing for Surveillance Applications in Transport Systems”. In: *IEEE Transactions on Instrumentation and Measurement* 66 (Jan. 2017), pp. 604–615. DOI: 10.1109/TIM.2016.2640518.

[71] Yilin Liu et al. “Leveraging the Properties of mmWave Signals for 3D Finger Motion Tracking for Interactive IoT Applications”. In: *ACM SIGMETRICS Performance Evaluation Review* 51 (June 2023), pp. 107–108. DOI: 10.1145/3606376.3593549.

[72] Junhua Wang et al. “Realtime wide-area vehicle trajectory tracking using millimeter-wave radar sensors and the open TJRD TS dataset”. In: *International Journal of Transportation Science and Technology* 12.1 (2023), pp. 273–290. ISSN: 2046-0430. DOI: https://doi.org/10.1016/j.ijtst.2022.02.006. URL: https://www.sciencedirect.com/science/article/pii/S2046043022000247.

[73] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[74] Wei Liu et al. “SSD: Single Shot Multibox Detector”. In: *European Conference on Computer Vision (ECCV)* (2016), pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2.

[75] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[76] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[77] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2015), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[78] Jifeng Dai et al. "R-FCN: Object Detection via Region-based Fully Convolutional Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016).

[79] Zhiqiang Zhao, Hengrong Fu, and Hongdong Li. "Object Detection with Deep Learning: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* (2019), pp. 1–21. DOI: 10.1109/TNNLS.2018.2876865.

[80] Yizhou Wang et al. "RODNet: Radar Object Detection Using Cross-Modal Supervision". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 504–513.

[81] Teck Yian Lim, Spencer Abraham Markowitz, and Minh Do. *RaDICaL: A Synchronized FMCW Radar, Depth, IMU and RGB Camera Data Dataset with Low-Level FMCW Radar Signals*. 2021. DOI: 10.13012/B2IDB-3289560_V1.

[82] Arthur Ouaknine et al. "CARRADA Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 5068–5075. DOI: 10.1109/ICPR48806.2021.9413181.

[83] M. Mahbubur Rahman et al. *MMVR: Millimeter-Wave Multi-View Radar Dataset and Benchmark for Indoor Perception*. 2024. arXiv: 2406.10708 [cs.CV]. URL: https://arxiv.org/abs/2406.10708.

[84] Han Cui et al. "MiliPoint: A Point Cloud Dataset for mmWave Radar". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 62713–62726.

[85] Christopher Mayershofer et al. "LOCO: Logistics Objects in Context". In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2020, pp. 961–966.

[86] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640.

[87] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: https://arxiv.org/abs/2004.10934.

[88] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *Scaled-YOLOv4: Scaling Cross Stage Partial Network*. 2021. arXiv: 2011.08036 [cs.CV]. URL: https://arxiv.org/abs/2011.08036.

[89] Alexander Kirillov et al. *PointRend: Image Segmentation as Rendering*. 2020. arXiv: 1912.08193 [cs.CV]. URL: https://arxiv.org/abs/1912.08193.

[90] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497.

[91] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV]. URL: https://arxiv.org/abs/1708.02002.

[92] H. Assoudi. “Dataset Preparation”. In: (2024). DOI: 10.1007/979-8-8688-1073-2_5.

[93] B. Dwyer, J. Nelson, and et al. Hansen T. *Roboflow (Version 1.0) [Software]*. Available from https://roboflow.com. 2024.

[94] Ulyana Tkachenko, Aditya Thyagarajan, and Jonas Mueller. “ObjectLab: Automated Diagnosis of Mislabeled Images in Object Detection Data”. In: *arXiv preprint* (2023). DOI: 10.48550/arXiv.2309.00832. URL: https://doi.org/10.48550/arXiv.2309.00832.

[95] Florian Wulff et al. “Railway LiDAR Semantic Segmentation Based on Intelligent Semi-Automated Data Annotation”. In: *arXiv preprint* (2024). DOI: 10.48550/arXiv.2410.13383. URL: https://doi.org/10.48550/arXiv.2410.13383.

[96] A. Ishtaiwi et al. “Impact of Data-Augmentation on Brain Tumor Detection Using Different YOLO Versions Models”. In: *The International Arab Journal of Information Technology* 21 (2024). DOI: 10.34028/iajit/21/3/10.

[97] C. Wang et al. “Automatic Detection of Indoor Occupancy Based on Improved YOLOv5 Model”. In: *Neural Computing and Applications* 23 (2023), pp. 2575–2599. DOI: 10.1007/s00521-022-07730-3.

[98] *Data Splitting in Machine Learning Process*. Available at: https://www.datainsightonline.com/post/data-splitting-in-machine-learning-process. 2023.

[99] *Splitting Data for Machine Learning Models*. Available at: https://www.geeksforgeeks.org/splitting-data-for-machine-learning-models/. 2023.

[100] Y. Xu and R. Goodacre. “On Splitting Training and Validation Sets: A Comparative Study of Cross-Validation, Bootstrap, and Systematic Sampling”. In: *Journal of Analysis and Testing* 2.3 (2018), pp. 249–262.

[101] V. R. Joseph and A. Vakayil. “Split: An Optimal Method for Data Splitting”. In: *Technometrics* 64.2 (2022), pp. 166–176. DOI: 10.1080/00401706.2021.1921037.

[102] Zijing Shi. “The distinguish between cats and dogs based on Detectron2 for automatic feeding”. In: *Applied and Computational Engineering* 38 (Feb. 2024), pp. 35–40. DOI: 10.54254/2755-2721/38/20230526.

[103] Ruohao Zhang, Yijie Lu, and Zhengfei Song. “YOLO sparse training and model pruning for street view house numbers recognition”. In: *Journal of Physics: Conference Series* 2646 (Dec. 2023), p. 012025. DOI: 10.1088/1742-6596/2646/1/012025.

[104] Chien-Yao Wang, Hong-Yuan Mark Liao, and I-Hau Yeh. “Designing Network Design Strategies Through Gradient Path Analysis”. In: *Journal of Information Science and Engineering* (2023).

[105] Texas Instruments. *IWR6843ISK Hardware Setup for Flashing Mode*. Accessed: 2025-01-05. 2023. URL: https://dev.ti.com/tirex/explore/content/mmwave_industrial_toolbox_4_7_0/labs/common/docs/hardware_setup/hw_setup_isk_ods_modular_mode_flashing.html.

[106] Texas Instruments. *Using UniFlash with mmWave Sensors*. Accessed: 2025-01-05. 2023. URL: https://dev.ti.com/tirex/explore/content/mmwave_industrial_toolbox_4_7_0/labs/common/docs/software_setup/using_uniflash_with_mmwave.html.

[107] *mmWave SDK Package*. Available at: https://www.ti.com/tool/MMWAVE-SDK. Texas Instruments.

[108] *mmWave Studio*. Available at: https://www.ti.com/tool/MMWAVE-STUDIO. Texas Instruments.

[109] FLW-TUDO. *py_mmwave_dev GitHub Repository*. Accessed: 2025-01-03. 2021. URL: https://github.com/FLW-TUDO/py_mmwave_dev.

[110] m6c7l. *pymmw GitHub Repository*. Accessed: 2025-01-03. 2020. URL: https://github.com/m6c7l/pymmw.

[111] nhma20. *iwr6843aop_pub GitHub Repository*. Accessed: 2025-01-03. 2021. URL: https://github.com/nhma20/iwr6843aop_pub.

[112] kimsooyoung. *mmwave_ti_ros GitHub Repository*. Accessed: 2025-01-03. 2021. URL: https://github.com/kimsooyoung/mmwave_ti_ros.

[113] Nobunori Aiura and Eiichi Taniguchi. "Planning of Loading-Unloading Spaces Considering behavior of Delivery Vehicles". In: *INFRASTRUCTURE PLANNING REVIEW* 22 (Oct. 2005), pp. 633–642. DOI: 10.2208/journalip.22.633.

[114] Abu Sied. "Advancing Warehouse Management Systems: Optimizing Loading-Unloading, Conditioning, Packing and Marking Processes with Adaptive AI Technology". In: *International Journal of Scientific Research and Engineering Trends* 10 (June 2024), pp. 2395–566. DOI: 10.61137/ijsret.vol.10.issue3.164.

[115] Ki In Bang et al. "LiDAR system calibration using point cloud coordinates in overlapping strips". In: vol. 1. Mar. 2009, pp. 9–13.

[116] Alexander Kamann et al. "Automotive Radar Multipath Propagation in Uncertain Environments". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 859–864. DOI: 10.1109/ITSC.2018.8570016.

[117] Tzutalin. *LabelImg GitHub Repository*. https://github.com/HumanSignal/labelImg. Accessed: 2025-01-16. 2015.

[118] Christoph Sager, Patrick Zschech, and Niklas Kühl. *labelCloud: A Lightweight Domain-Independent Labeling Tool for 3D Object Detection in Point Clouds*. 2021. arXiv: 2103.04970 [cs.CV].

[119] Ultralytics LLC. *Ultralytics GitHub Repository*. Accessed: 2025-02-06. 2025. URL: https://github.com/ultralytics/ultralytics.

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

List of Figures

2.1 Principle of Frequency-Modulated Continuous Wave (FMCW) radar [22]. . . . 6

2.2 Simplified block diagram of an FMCW radar system [16]. 6

2.3 Transmitted and received chirps for range measurement [26]. 7

2.4 Illustration of velocity measurement with two chirps [16]. 8

2.5 Angle estimation in FMCW radar using multiple antennas [20]. 8

2.6 Texas Instruments IWR6843 ISK Evaluation Module[27]. 9

2.7 Texas Instruments MMWAVEICBOOST Board[28] 10

2.8 Block diagram of an FMCW sensor[16]. 11

2.9 Illustration of data capture based on FMCW radar[8]. 12

2.10 Range-Angle spectrum example showing distance and angular position of objects [30]. 12

2.11 Flowchart of point cloud generation using mmWave radar[8]. 13

2.12 Example of a point cloud generated by mmWave radar[8]. 13

2.13 A Simple Neural Network Diagram [38] 15

2.14 Simple Network with Feedback and Competition [38] 16

2.15 Supervised and Unsupervised Learning [44] 16

2.16 Workflow of Camera and Radar Sensor Fusion [37]. 17

3.1 Comparison of AP vs. inference time on V100 GPU for various real-time object detectors. YOLOv7 achieves state-of-the-art performance [18]. 21

4.1 Architecture of the YOLOv7 model 24

4.2 Extended Efficient Layer Aggregation Networks (E-ELAN) [18]. 25

4.3 RetinaNet Architecture. 26

4.4 Faster R-CNN detailed architecture. 27

4.5 Design pipeline for the offline model testing experiment. 27

4.6 Design pipeline for the online model testing experiment. 28

4.7 Assembly of the IWR6843ISK module and MMWAVEICBOOST board. 32

5.1 Illustrations of dataset objects: (a) Forklift, (b) Forklift with KLT, (c) Workstation, (d) Robotnik. 36

5.2 Dimensions and Vicon coordinates of the FLW logistics hall. 36

5.3 Representation of the dataset collection setup. 37

5.4 Example of a range-azimuth 2D histogram with a reference camera frame. . . . 39

5.5 Four possible radar sensor positions in the arena. 40

5.6 Boundary Condition: Comparison between raw input and filtered input with boundary conditions. Points outside the experiment area are excluded from further processing. 42

5.7 Multipath Reflection: Comparison between raw input and filtered input with multipath. The filtering process removes less reliable readings based on SNR values. 43

5.8 Labeling range-azimuth data using the LabelImg tool. 44

5.9 Labeling 3D point cloud data using the LabelCloud tool. 45

5.10 Object detection output for Forklift with KLT. 47

5.11 Object detection output for Forklift. 48

5.12 Object detection output for Workstation. 48

5.13 Object detection output for Robotnik robot. 48

5.14 Confusion matrix for static dataset-trained models evaluated on test data. 50

5.15 Confusion matrix for dynamic dataset-trained models evaluated on test data. . . 50

5.16 (a) Precision-recall (PR) curve of static model, (b) PR curve of dynamic model, (c) F1 curve of static model, and (d) F1 curve of dynamic model. 51

5.17 Training results over epochs showing precision, recall, mAP@0.5, mAP@0.95, box, objectness, and classification metrics. 52

5.18 Confusion matrix comparison of CNN, Faster R-CNN, and RetinaNet backbone algorithms 53

5.19 Overview of the latency components in online model. 54

5.20 Examples of online model testing outputs, showing processed sensor data and detected objects. Live visual representation through the camera frame illustrates the real environment and the model's detection performance. 54

5.21 Distribution of prediction scores for the online object detection model across different objects. 55

5.22 Confusion matrix for static and dynamic dataset-trained models evaluated on test data. 57

5.23 Precision-Recall and F1 curves for static and dynamic models. 59

5.24 3D point cloud model detection output for Forklift and Workstation. 60

5.25 3D point cloud model detection output for Forklift+KLT and Robotnik. 60

5.26 Comparison of radar and Vicon system detections in 2D (x, y coordinates). . . 61

5.27 3D comparison of radar and Vicon system detections. 62

List of Tables

2.1 Comparison of Millimeter-Wave Sensor Models [21] 9

2.2 IWR6843 ISK Evaluation Module Specifications[27] 10

3.1 Performance comparison of YOLOv7 variants on MS COCO dataset [18]. . . . 21

3.2 COCO Object Detection Baselines for Faster R-CNN [19]. 21

3.3 COCO Object Detection Baselines for RetinaNet [19]. 21

4.1 S1 Switch Configuration for Flash Mode 31

4.2 S1 Switch Configuration for MMWAVEICBOOST Mode 33

4.3 Sensor Configuration for mmWave Radar (`IWR6843ISK`) 33

5.1 Static Data Collection 38

5.2 Dynamic Data Collection 38

5.3 Comparison of key metrics for static and dynamic models. 58

List of Abbreviations

RADAR RAdio Detection And Ranging I

LiDAR Light Detection and Ranging I

KLT Small Load Carriers (Kleinladungsträger in German) I

IoT Internet of Things I

AGV Automated Guided Vehicle 1

FMCW Frequency Modulated Continuous Wave 1

CNN Convolutional Neural Network 3

RCNN Region-based Convolutional Neural Network 3

YOLO You Only Look Once 3

JCAS Joint Communication and Sensing 5

FOV Field of View 9

CFAR Constant False Alarm Rate 10

FFT Fast Fourier Transform 11

LPF Low Pass Filter 11

ADC Analog-to-Digital Converter 11

mAP Mean Average Precision 16

E-ELAN Enhanced Efficient Layer Aggregation Network 25

COCO Common Objects in Context 26

FPN Feature Pyramid Network 26

RPN Region Proposal Network 26

NLP Natural Language Processing 28

YACS Yet Another Configuration System 31

EVM Embedded Visual Model 31

IDE Integrated Development Environment 34

SNR Signal-to-Noise Ratio 37

DoF Degrees of Freedom 44

IoU Intersection over Union 47

FP False Positive 47

FN False Negative 47

AP Average Precision 47

NMS Non-Maximum Suppression 54

PCRCNN Point Cloud Region-based Convolutional Neural Network 56

SECOND Sparse Encoded ConvNet Detector 57

PV-RCNN PointVoxel Region-based Convolutional Neural Network 57

ROI Region of Interest 58

Eidesstattliche Versicherung

(Affidavit)

Rajpurkar, Sahil Sanjay
Name, Vorname
(surname, first name)

242976
Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Towards 6G-Driven Sensing: Development of Machine Learning-based Radar Object Detection for Logistics Entities

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 13.02.2025
Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 13.02.2025
Ort, Datum
(place, date)

Unterschrift
(signature)

*Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.