UiA University of Agder

Master

Target detection and localization using thermal camera, mmWave radar and deep learning

Authors

Sindre Beiermann & Andrea Tande

Supervisor

Linga Reddy Cenkeramaddi

University of Agder, 2023

Faculty of Engineering and Science

Department of Engineering and Sciences

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Vi erklærer herved at vår besvarelse er vårt eget arbeid, og at vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	**Vi erklærer videre at denne besvarelsen:** • Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • Ikke refererer til andres arbeid uten at det er oppgitt. • Ikke refererer til eget tidligere arbeid uten at det er oppgitt. • Har alle referansene oppgitt i litteraturlisten. • Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert.	Ja
5.	Vi er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	Nei

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).
Oppgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei

Acknowledgements

We would like to say thank you to Prof. Linga Reddy Cenkeramaddi for being our supervisor and for giving us the assignment. We would also like to say thank you to Wilson Ayyanthole Nelson for all the help with system setup and technical issues during our project. We would also like to say thank you to all of our friends, that has borrowed their cars, and stood still so that we could collect the data set. Also for all the discussion and support during the thesis.

Abstract

Reliable detection, and localization of tiny unmanned aerial vehicles (UAVs), birds, and other aerial vehicles with small cross-sections is an ongoing challenge. The detection task becomes even more challenging in harsh weather conditions such as snow, fog, and dust. RGB camera-based sensing is widely used for some tasks, especially navigation. However, the RGB camera's performance degrades in poor lighting conditions. On the other hand, mmWave radars perform very well in harsh weather conditions also. Additionally, thermal cameras perform reliably in low lighting conditions too. The combination of these two sensors makes an excellent choice for many of these applications. In this work, a model to detect and localize UAVs is made using an integrated system of a thermal camera and mmWave radar. Data collected with the integrated sensors are used to train a model for object detection using the yolov5 algorithm. The model detects and classifies objects such as humans, cars and UAVs. The images from the thermal camera are used in combination with the trained model to localize UAVs in the cameras Field of View(FOV).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an overview of the project, including its motivation and research objectives. It also presents the project management structure, roles, and responsibilities of the group members.

1.1 Background

The increasing usage of unmanned aerial vehicles (UAVs) in various applications, such as rescue operations [8] and deliveries [7], highlights the need for reliable detection, localization, and tracking of these aerial vehicles. However, detecting, localizing, and tracking UAVs remains a challenge, especially in harsh weather conditions such as snow, fog, and dust. Although traditional RGB camera-based sensing is frequently used for navigation, its performance suffers in low-light conditions. In contrast, millimeter-wave (mmWave) radar and thermal cameras can perform well in harsh weather conditions too. Thermal cameras can detect objects based on their heat signature independent of lighting conditions.

1.1.1 Motivation

The motivation behind this research is to address the challenge of detecting and localizing UAVs. As the usage of UAVs grows, so does the demand for reliable detection systems capable of identifying these vehicles from other types of objects. Existing detection systems may be affected by adverse weather conditions, making it difficult to detect and locate objects. By integrating thermal cameras and mmWave radar, this research aims to develop a system that can successfully integrate these sensors for this purpose. Our approach combines both sensors to provide a robust and accurate solution for UAV detection and localization.

1.2 Research Objectives

The research objectives of this thesis are as follows:

1. Integrate thermal camera and mmWave radar to enhance detection and locating capabilities.
2. Collect a data set to develop a system for detecting and localizing UAVs.
3. Differentiate UAVs from other objects, such as cars and humans, using the object detection algorithm YOLOv5.
4. Use the trained model to localize objects in the camera's (FOV).

1.3 Project Management

The project group consists of two participants, Sindre Beiermann and Andrea Tande, who both hold a bachelor's degree in computer engineering from Østfold University College and are currently pursuing a Master's degree in the field of communication and information technology at the University of Agder. The same educational basis provided them with a common understanding of the principles and concepts in the field of information and communication technology, which has been helpful for the development and execution of the project.

Roles and Responsibilities
Each project member has been assigned specific roles to ensure that all tasks are carried out as agreed. Andrea serves as the project manager, overseeing the project, meeting deadlines, and is primarily responsible for the design and writing of the thesis. Sindre takes on the role of technical manager, overseeing the technical aspects of the project, and ensuring that all data collection and analysis is done correctly. Documentation and project reporting will also be a significant responsibility.

Meetings
Regular meetings have been scheduled to facilitate progress and discuss questions and other matters that may arise. The project group has weekly meetings with the supervisor either digitally or physically, which could be changed at short notice.

Project structure
This project utilized an agile approach with task management and deadlines organized using Microsoft Teams' to-do list feature. This methodology provided structure, motivation, and effective communication through regular meetings with the supervisor. The project group divided tasks and set measurable goals for each sprint, following the principles of agile methodology[9].

An important benefit of adopting the agile methodology was the continuous feedback received from the supervisor. This valuable input played a vital role in providing guidance and making

necessary adjustments throughout the project.

Chapter 2

Background Theory

This chapter provides the theoretical foundation related to the project. It begins with an overview of the sensor and hardware components used in the system. Next, the chapter delves into the field of machine learning and deep learning.

2.1 Sensor and Hardware Components

2.1.1 Raspberry Pi

Created by the UK-based Raspberry Pi Foundation, the Raspberry Pi is a single-board computer known for both its powerful capabilities and small size. It was initially intended for promoting computer science education and to provide an affordable computing platform for students and hobbyists[15]. It released its first model back in 2012 which has since received several upgrades with advanced performance and feature additions. Raspberry Pi computers are built upon Advanced RISC Machine(ARM) architecture commonly used in mobile devices; employing System-on-Chip design incorporating CPU along with memory graphics & I/O Ports integrated onto a single chip.

The Raspberry Pi has numerous applications in education, robotics, and Internet of Things(IoT) projects. The Foundation provides tutorials on building projects such as weather stations using a Raspberry Pi and teaching programming with Minecraft[16]. In robotics, the Raspberry Pi can be used as the brain of a robot, controlling its movement and behavior. In IoT projects, the Raspberry Pi can be used to collect data and process data from sensors and other devices.

2.1.2 Thermal Camera

Thermal cameras also known as infrared cameras, detect and measure the heat emitted by objects with a temperature above zero[42]. By capturing the infrared radiation emitted by objects, these cameras convert thermal energy into a visual image.
The functioning of thermal cameras relies on a detector consisting of thousands of sensors called microbolometers. These sensors are particularly sensitive to variations in temperature[2].

When the infrared radiation from an object reaches the detector, it causes the temperature of each microbolometer to change. This temperature change generates an electronic signal proportional to the temperature of the object. These signals are then used to create a thermal image. Each pixel shows the temperature of a tiny area of the object's surface. This enables the detection of temperature variations that are otherwise undetectable to the human eye.

Furthermore, because thermal cameras use heat emitted by objects rather than visible light, they may be used in both daylight and low-light circumstances. As a result, they are suited for a wide range of applications in diverse environments and lighting conditions. Thermal cameras have a wide range of applications, including security and medical imaging. In security applications, thermal cameras can detect intruders in dark or smoke-filled environments [37]. In medical imaging, thermal cameras can detect skin cancer[44] or monitor the blood flow in veins and arteries.

2.1.3 Radar

Radar is a technology that employs radio waves to detect, locate and track nearby objects. To achieve this, radar systems project radio waves into the surroundings which bounce off any proximate objects before making their way back to the receiver. Calculating the timeframe it takes for this signal to bounce back can be used to determine the distance of an object. Analyzing the frequency and phase of the retrieved signal can find information about the object like its direction, velocity, and size[23].

The basic components of a radar system comprise a transmitter, receiver, antenna, and processor[13]. The transmitter sends out a radio signal which travels through an antenna. If this runs into anything solid instead of going through it like air or water, it will be reflected to the antenna and then picked up by the receiver. The "round trip time" is the amount of time it takes for a signal to reach an object and return. This interval determines how far away the object is. The received signal is usually very weak, as it has traveled a long distance and bounced off various objects before being picked up by the antenna. Therefore, the received signal must be amplified and filtered by the receiver before being sent to the processor.

Radar receivers typically have several components like a low-noise amplifier(LNA), mixer along with filter[13]. The LNA is the first component to encounter the received signal and it is responsible for amplifying the signal to a high enough level so that subsequent steps can be processed. Then, the mixer downconverts the signal to a lower frequency that is better suited for further processing. Finally, the filter removes unwanted noise and interference from the signal, such as noise from other electronic devices or atmospheric interference.

Once the received signal has been amplified and filtered, it is sent to the radar's digital signal

processor for further analysis. The processor uses advanced algorithms and signal processing techniques to extract information about the detected objects, such as their distance, speed, and direction of movement[13]. As an object approaches or moves away from the radar system, the frequency of the reflected signal shifts slightly due to the Doppler effect. By measuring this frequency shift, the radar system can determine the object's speed and direction of movement [23].

Radar technology is used in a variety of applications, from air traffic control [17] to weather forecasting [22]. Today, radar systems use advanced digital signal processing techniques, complex algorithms, and multiple sensors to achieve high accuracy, precision, and reliability. As a result, radar technology has become an important tool in many industries, contributing to safety, security, and efficiency in various applications.

2.2 Machine Learning

Machine learning is a subfield of Artificial Intelligence (AI) that focuses on developing tools and algorithms capable of learning complex relations in features from a set of data. These learned insights are then leveraged to perform various tasks such as classification or regression on similar data. Machine learning is often confused with AI and deep learning, as these terms are frequently used interchangeably. However, AI categorizes a wider range of programs and algorithms that perform anything that can be seen as intelligent, even encompassing relatively simple algorithms like the k-nearest neighbors algorithm (K-NN) [31].

Figure 2.1 provides an illustration of the relationship, showing that machine learning is a subset of AI rather than a synonym. On the other hand, deep learning refers to a particular field within machine learning that focuses on the utilization of deep neural networks, which have gained significant attention in recent years due to their impressive performance across diverse tasks and applications[4].

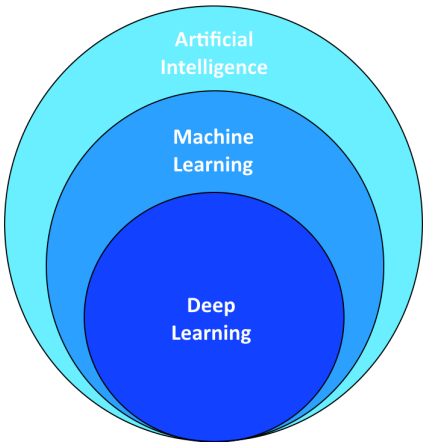

Figure 2.1: An illustration of the relation between artificial intelligence, machine learning, and deep learning.

Within the domain of machine learning, algorithms can be divided into three main types based on how they learn from data. In all cases, an algorithm, denoted as f, is provided with data in the form of a feature vector $x \in \mathbb{R}^m$ that is used to make predictions $f(x) = \hat{y} \in \mathbb{R}^v$. Here, m represents the number of attributes in the data, while v signifies the number of classes for prediction. This process can be interpreted as a mapping from the input space to a complex embedding function

$f : \mathcal{X} \to \hat{\mathcal{Y}}$. The end goal is then to train the model so that the predictions become as close as possible to the ground truth y, thus $f(x) = \hat{y} \approx y$ [4]. The difference between the three categories is how they use data to train.

Supervised Learning

In supervised learning, the true values for the entire data set are known. The algorithm works by having a supervisor during training. This is accomplished by providing the machine with labeled data, and the machine learns to recognize patterns and similarities among the labeled data[25]. After training, the machine can use this knowledge to make predictions or decisions on new, unlabeled data. Figure 2.2 shows a pipeline model of supervised learning, which involves selecting an appropriate model: either classification or regression algorithms[26]. **Classification** algorithms are used to identify different categories based on independent variables, such as "monkey" or "non-monkey". **Regression** algorithms are used to predict output for real or continuous values, such as estimating the weight of a monkey or the number of visitors to a zoo.

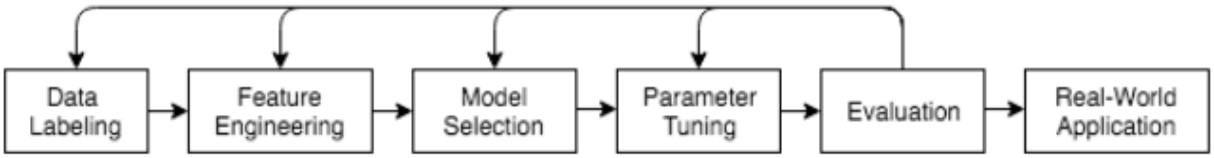

Figure 2.2: Supervised Learning Pipeline [39]

One critical step in supervised learning is data labeling, which involves annotating data to provide ground truth for machine learning algorithms. Data labeling is necessary for developing a machine learning system. It can be outsourced to third-party services or by developers themselves to get insight into their data set[39].

Unsupervised Learning

Unsupervised learning is trained on unlabeled data. It is assumed that there exists a ground truth, but it is not known. It involves grouping the data based on similarities, patterns, and differences without the need for a supervisor[19]. Unsupervised learning algorithms can be used for clustering or association. **Clustering** algorithms group similar data points into clusters, such as exotic animals in the zoo. **Association** algorithms identify rules that explain portions of data, such as people who buy product X often buy product Y[19].

Reinforcement Learning

Reinforcement learning is a type of machine learning where the algorithm learns by trial and error. The data does not come in the form of a set but rather is an interactive environment. The algorithm receives feedback in the form of rewards or penalties based on its actions, enabling it to improve its behavior over time[27]. The reinforcement signal can be positive, strengthening behaviors that lead to desirable outcomes, or negative, weakening behaviors that result in undesirable outcomes.

Imagine an intelligent animal in a zoo aiming to escape. The animal, acting as the reinforcement learning agent, explores its surroundings to discover an optimal path to freedom. As it explores, the animal received feedback. Finding an unlocked gate or an open door could result in a positive reward while encountering a dead-end or a locked gate could result in a negative penalty. Over time, as the animal continues to explore and learn from its environment, it refines its strategy and discovers the optimal path to escape the zoo with the highest cumulative reward.

2.3 Deep Learning and Neural Networks

Deep Learning is a subset of machine learning that focuses on the development and application of Artificial Neural Networks (ANNs), which are computational models inspired by the structure and functioning of the human brain. ANNs consist of interconnected nodes, called neurons, organized in layers. Each neuron receives input, applies an activation function to the weighted sum of those inputs, applied an activation function to the weighted sum of those inputs, and produces an output. The connections between neurons have associated weights that determine the strength of their influence on the output.

Deep Learning involved training deep neural networks with multiple hidden layers to learn complex representations and hierarchies of features from the data. This ability to automatically learn hierarchical representations makes deep learning highly effective for tasks such as image and speech recognition, natural language processing, and more.

Neural Networks

Neural networks represent a fundamental component of deep learning. They are composed of multiple layers of artificial neurons that process and transform input data[3].
A neural network consists of multiple layers, each comprising artificial neurons that hold scalar values. The first layer h_1, named the input layer, is where the feature vector $x \in \mathbb{R}^m$ is fed into the network. Therefore, the input layer needs to have m number of artificial neurons, each representing one attribute of the feature vector. The last layer h_n, named the output layer, provides the embedding $\hat{y} \in \mathbb{R}^v$, and therefore has to consist of v artificial neurons. The intermediate layers $h_2 \dots h_{n-1}$ are known as hidden layers, and serve to pass

information without interference[3].

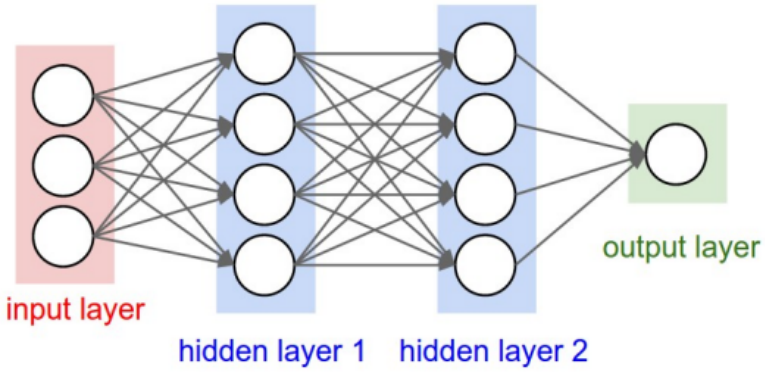

Figure 2.3: An illustration of a feed-forward fully connected neural network. [1]

The layers in a neural network are connected by edges, representing one-way connections between neurons in different layers. These edges are associated with scalar weights α, which represent the impact level of the connection. The most common architecture is feed-forward fully connected layers[3], where all neurons in adjacent layers are connected, and information flows in one direction.

The concept is that each neuron in the hidden layers learns some sort of complex feature about the input data, which is then utilized by the output layer to form a decision. The size and number of hidden layers depend on the specific application and are an essential aspect of the design process. A broad network with many neurons per layer can identify more features, while a deep network with many layers can develop more intricate features. As shown in Figure 2.3, the neural network consists of an input layer (orange), two hidden layers (red), and an output layer (blue) with a single neuron.

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a type of deep neural network that takes into account the order of the input data, inspired by the mathematical operation of convolution. This makes it possible for the network to analyze and identify local features within the input rather than an entire layer. This concept has been proven to provide good results in picture classifications, as seen by the ratings on the massive image classification benchmark ImageNet[11].

The structure of a CNN consists of three main parts: a convolutional layer, a pooling layer, and a fully connected layer. Figure 2.4 illustrates a simplified CNN architecture used for MNIST classification.

[1] https://brilliant.org/wiki/feedforward-neural-networks/

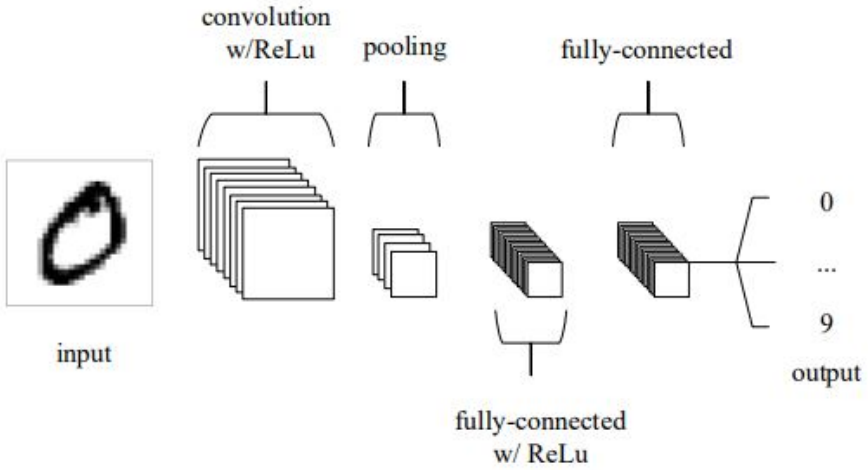

Figure 2.4: Illustration of a CNN architecture [2]

In this architecture, the input layer like in other types of neural networks, represents the pixel values of the image. The convolutional layer is responsible for extracting features from the input image. It does so by connecting neurons to local regions of the input and performing This is achieved by calculating the scalar product between the weight of the neurons and the corresponding region of the input volume. Each convolution kernel within this layer focuses on extracting a different aspect of the feature [21] via the formula:

$$E = f(MW + b) \tag{2.1}$$

where E represents the retrieved feature matrix after the convolution process, M represents the input, W represents the weight matrix (the convolution kernel), and b is the bias vector of the network.

The pooling layer performs downsampling along the spatial dimension of the input. By One advantage of CNNs is their shift invariance, which means that a small shift in the input does not significantly impact the output. This property allows CNNs to effectively handle relation-based data, such as images, without needing to learn all possible object variations and placements. Figure 2.4 illustrates the working principle of a CNN.

YOLOv5 Object detection network

YOLOv5(You Only Look Once version 5) is an advanced object detection algorithm that aims to achieve real-time and high-precision object detection in images and videos. It builds upon the success of its predecessors, YOLOv1, YOLOv2, YOLOv3, and YOLOv4. It is composed of four main components: the input processing, backbone network, feature fusion network, and prediction module.
The input module executes picture preparation tasks such as zooming, data augmentation, and initial prediction box modification. These steps help enhance the quality and variety

[2] https://arxiv.org/pdf/1511.08458.pdf

of input data[12]. The backbone network extracts scale features at different scales. It consists of multiple layers, with the shallow layers capturing low-level features like edges, colors, and texture, while the deep layers focus on extracting high-level semantic features. The feature fusion network combines the low-level and high-level features obtained from the backbone network. By fusing these features, it generates highly detailed and semantically rich representations. This fusion process enhances the ability of the network to capture intricate patterns and meaningful information from the input data. The prediction module utilizes the fused features to make predictions at multiple scales. It produces bounding box predictions and class probabilities for various objects present in the input image[12].

2.4 Robot Operating System

The Robot Operating System (ROS) is an open-source framework for developing robotic systems. The fact that it is open-source makes the integration of new libraries or tools easy to bring to the platform[46].

ROS Architecture

ROS uses a node-based architecture to communicate between software components and hardware devices. Each node performs specific tasks, and the messaging system is based on a publish-subscribe system[32]. This means that different parts of a robotic system can communicate by publishing and subscribing to messages. ROS topics are used to transmit data between nodes. Each node can either publish or subscribe to a topic depending on whether they are sending or receiving information. These topics hold data types called messages that are used to transmit information between nodes[32]. When a node publishes a message, all other nodes that have subscribed to that topic receive the message and can process it accordingly.

ROS Melodic

ROS Melodic is the twelfth release of ROS and is specifically designed for the Ubuntu 18.04 (Bionic) operating system[46]. ROS Melodic supports various programming languages such as C++, Python, and others, making it accessible to a wide range of developers. It introduces improved support for newer hardware, including better compatibility with sensors and robotic platforms.

2.5 Related Work

This subsection provides a comprehensive review of the existing research in this field, with a focus on identifying on relevant aspects.

2.5.1 Object Detection Models

Object detection models have made significant progress in meeting the challenges of processing speed and accuracy. Traditional two-stage architectures, such as R-CNN have limitations in terms of their processing speed[20]. This has led to the development of one-stage deep neural networks such as YOLO, RetinaNet, and SDD [35] [33]. YOLO can simultaneously perform classification and bounding box regression, which leads to faster inference times. The subsequent versions of YOLO, like YOLO-v3, YOLO-v4, and YOLO-v5 have further refined their accuracy and speed. Another one-stage model, SSD, utilized predefined default boxes and scale-invariant features to detect objects [35]. RetinaNet introduces a modified focal loss mechanism that assigns smaller weights to easily detectable objects while allocating larger weights to more challenging objects[30].

Although anchor box-based detectors have shown effectiveness, they can be sensitive to hyper-parameters. To address this, anchor-free methods like FCOS have been proposed[43]. However, FCOS performs pixel-wise bounding box prediction, which increases the execution time for the detection-then-classification task.

2.5.2 Radar Target Detection

In recent years, deep neural network (DNN) architectures such as ResNET[41], AlexNet[28], VGGNet [36], and GoogLeNet[40] have achieved remarkable success in several fields, including radar target detection (RTD). Researchers have proposed several approaches based on artificial neural networks (ANNs) and DNNs to address the challenges of RTD in complex scenarios.

Gandhi et al.[18] were among the pioneers in using ANNs to detect signals in non-Gaussion noise and differentiate targets from noise. Amores et al. [24] showed the potential of ANN-based methods to improve the robustness of radar detectors. They demonstrated the effectiveness of using ANNs to improve the detection performance in challenging noise environments.

Rohman et al. [34] introduced an adaptive ANN-CFAR detector to improve RTD performance in non-homogeneous noise. Constant False Alarm Rate (CFAR) detection is a common technique used in radar systems. By integrating an adaptive ANN into CFAR detection, a better performance in handling non-homogeneous noise scenarios was achieved.

While distinguishing targets from noise is a fundamental goal, detecting targets in cluttered environments is a more common but challenging task. Cheikh et al. [10] investigated RTD using various ANN architectures in the presence of K-distribution clutter. Akhtar et al. [1] proposed a general training strategy for extracting floating targets in cluttered environments. Additionally, Pan et al.[29] introduced a deep CNN approach for detecting small marine targets in strong sea clutter backgrounds. These studies addressed the binary classification problem of determining the presence or absence of targets. In addition to single input

modes, researchers have explored the use of multiple inputs in deep learning models for RTD. For example, Pan et al. [29] used pulse-range images as inputs to a deep CNN model, while Wand et al. [45] designed a CNN-based target detector using range-Dopper spectrums and compared it against traditional CFAR detectors. The range-dopper spectrum provides a two-dimensional representation of radar returns and captures both range and doppler information.

Brodeski et al.[5] proposed a CNN-based architecture for automotive radar detection using range-doppler data for target location. Gustav et al.[6] constructed a time-frequency block using the Wigner-Ville distribution (WVD) and developed a WVD-CNN detector for RTD. The WVD is a time-frequency analysis tool that provides a high-resolution representation of non-stationary signals. By combining the WVD with a CNN, this paper aimed to improve RTD performance by capturing time-frequency characteristics of radar signals.

Su et al. [38] applied the short-time Fourier transform (STFT) to preprocess IPIX-measured data and designed a CNN-based method for maritime target detection. STFT is a time-frequency analysis technique commonly used to extract signal features. Su et al. used STFT to preprocess measured data, preparing it for input to a CNN-based method for detecting maritime targets.

These previous studies provide valuable insights into the application of ANNs and DNNs for RTD, addressing various aspects such as noise, clutter, and multi-modal inputs. However, further research is still needed for improving the performance and applicability of RTD systems in real-world scenarios.

Chapter 3

Methodology

This chapter provides an overview of the approach and techniques employed in the project. It begins with a detailed description of the data collection process, including the hardware components used and the system configuration. The data set is introduced, highlighting the measurement plan and the data labeling process. The chapter concludes with an overview of the data processing steps, including data extraction and model training using the YOLOv5 algorithm.

3.1 Data Collection

The initial phase of the project involved the process of collecting data, which is a critical aspect of any machine learning project. The quality of the collected data directly affects the subsequent analysis and model performance. This section presents the hardware and system setup used for data collection.

3.1.1 Hardware

The data collection process relied on several hardware components to ensure accurate and reliable data capture. The following components were utilized:

Raspberry Pi 4 Model B
The Raspberry Pi 4 Model B is a powerful computer with a 1.5 GHz quad-core ARM Cortex-A72 processor and 4 GB of RAM[16]. It provides enough processing power for data collection. Additionally, it is highly programmable and can easily be integrated with other hardware components. The Raspberry Pi's GPIO pins make it easy to connect to other sensors and hardware devices. The built-in WiFi capability enables wireless connection.

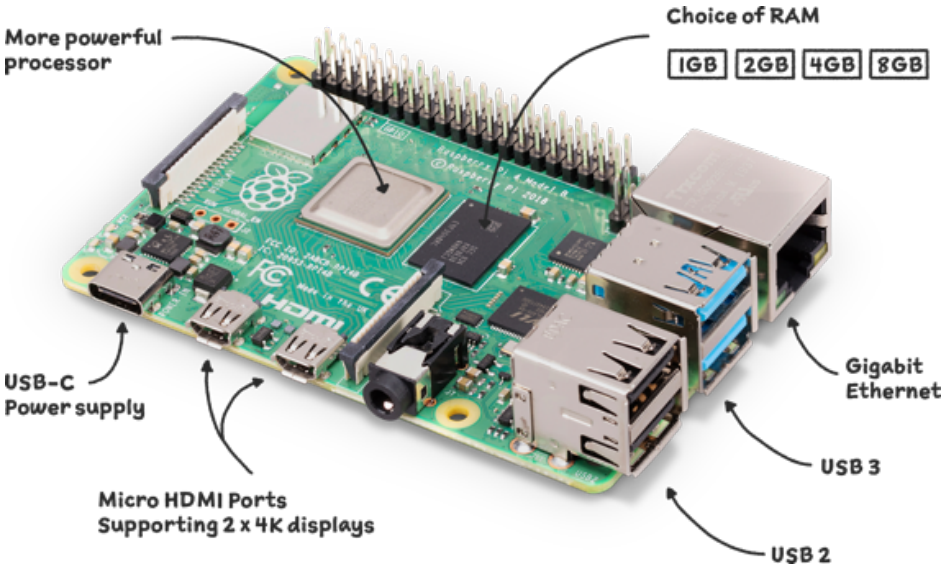

Figure 3.1: Raspberry Pi 4 Model B. The figure is taken from their webpage.[1]

FLIR Lepton 3.5

The FLIR Lepton 3.5 thermal camera presented in Figure 3.2, offers the highest resolution among Leptons, featuring arrays with 160x120 pixels with a 57-degree field of view[14]. It can detect temperature differences as small as 0.005 °C [14], making it effective in detecting heat signatures from a distance. The operating temperature range of the camera is from -10 to +65 °C.

Figure 3.2: FLIR Lepton 3.5. The figure is taken from their webpage.[2]

Texas Instruments IWR1642

The Texas Instruments IWR1642 radar sensor, figure 3.3 operates at a frequency of 76-81 GHz and has a range of up to 60 meters with an accuracy of +/-7.5 cm[23]. It provides the ability to detect the presence and movement of objects in the environment.

[1] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[2] https://www.flir.eu/products/lepton/?model=500-0771-01&vertical=microcam&segment=oem#mz-expanded-view-114538616485

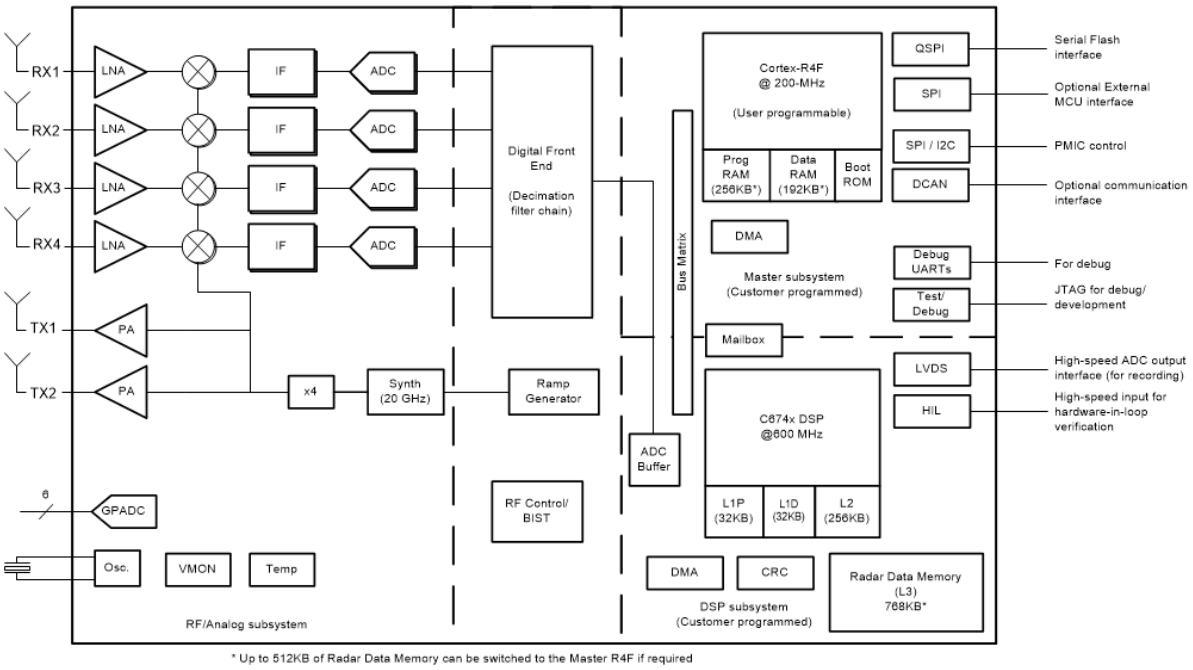

Figure 3.3: Texas Instruments IWR1642 Radar Sensor Block Diagram. The diagram is taken from the mmWave product page.[4]

3.1.2 Hardware Setup

The Raspberry Pi, radar, and power supply were placed in a PLA 3D-printed case, shown in Figure 3.4. This custom case provides stability and allows for mounting on a tripod. The FLIR Lepton 3.5 thermal camera was fitted into a separate PLA 3D-printed case, shown in Figure 3.5. This case was mounted directly above the radar so that simultaneous measurements could be taken from the same angle.

Figure 3.4: Case for Raspberry Pi, radar, and Power supply.[6]

Figure 3.5: Custom 3D-printed case for thermal camera.[8]

[4] https://www.ti.com/product/IWR1642
[5] https://www.thingiverse.com/thing:3585235
[7] https://www.thingiverse.com/thing:4642813

3.1.3 Sensor Configuration

The Raspberry Pi Model B served as the main processing unit, responsible for collecting data from the thermal camera and the radar. It was equipped with Linux Ubuntu 18.04 and ROS Melodic. The thermal camera and radar sensor were connected to the Raspberry Pi, allowing simultaneous data capture. In figure 3.6 the full setup is shown. To initiate the data collection process, the system was powered by a reliable power source, in this case a power bank.

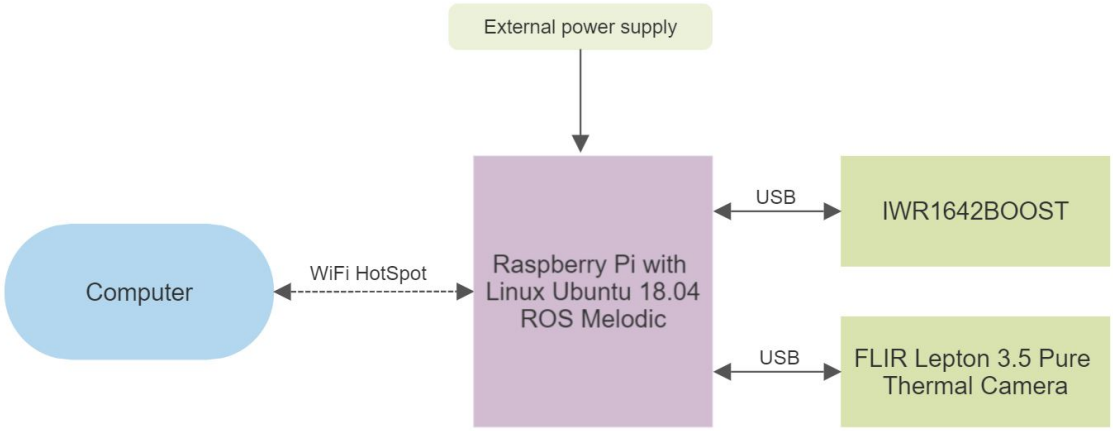

Figure 3.6: Flow diagram of the setup

Thermal Camera setup

The thermal camera was accessed using the **capra_thermal_cam** [9] ROS package, which provides a ROS interface for the camera. The **capra_thermal_cam** package relies on another package called **usb_cam** [10]. Additionally, the **purethermal1-uvc-capture**[11] package was installed to obtain images from the thermal camera.

The launch file for the **capra_thermal_cam** package was modified to adjust the frames per second (FPS) to match the capabilities of the camera and the USB port that the camera was connected with.

Radar setup

The radar was accessed using the **ti_mmwave_rospkg**[12] package. This package offers ROS support for Texas Instruments IWR mmWave radar sensors with several launch files. For this case the launch file 1641esp1_short_range was used.

Launch Script

A custom ROS package and launch script were created to collect the samples simultaneously

[9] https://github.com/clubcapra/capra_thermal_cam
[10] https://github.com/ros-drivers/usb_cam
[11] https://github.com/groupgets/purethermal1-uvc-capture
[12] https://github.com/radar-lab/ti_mmwave_rospkg

from both the thermal camera and radar. A Launch script is a script that starts up all the necessary things with the correct settings for the camera or radar two work. But to make the thermal camera and radar collect at the same time it was necessary to create a script that would launch both packages while also saving the samples in a bag file with a specific name using ROS bag commands [13].

Remote Access setup

A WiFi hotspot was set up on the Raspberry Pi To enable remote access while performing measurements. This lets users connect to the system through SSH via the WiFi hotspot. Since SSH requires a static IP, the WiFi hotspot starts automatically when the Raspberry Pi is turned on. Any user who wishes to log in can connect to this hotspot, connect to the Raspberry Pi through SSH, and execute commands or debug. This is useful for data collection, as it allows users to easily connect to the system and collect data without having to be physically present at the system.

3.1.4 Data Set

The data set is a collection of images that captured humans, cars, and UAVs placed in various positions and distances. With a desired number of 300 instances for each object category, the data set encompasses 150 different cases, utilizing two DJI mini SE Drones, two cars, and two humans in each case.

Measurement Plan

A measurement plan was devised to ensure precise object tracking in each case. The plan involved drawing up a designated area on the ground where the objects were precisely positioned. Initially, the area spanned from 60 degrees to 120 degrees, but later it was converted for the purpose of comparison. The area was also divided into distinct lengths, creating a grid-like structure for the precise placement of the objects. The measurement plan is illustrated in Figure 3.7, and is sourced from Appendix A.

Range (m) →													
Cases ↓	1	2	3	4	5	7	9	11	13	15	17	19	21
1						C1(15)	U1(-15)	H1(0)		H2(-10)		C2(5)	
2							U2(-10°)		H1(15)	C1(-20°)			C2(5°)
3						C1(15)	U1(-15)			H1(30)	H2(-5)	C2(10)	
4		U2(-20)		U2(-10°)					C2(10°)		H1(0°)		C1(-15°)
5							U2(-15°) H1(15°)		C2(10°)				C1(-20°)

Figure 3.7: Measurement Plan

Despite the camera having a FOV of 28.5, the objects were captured within a range from -30 to 30 degrees relative to the position of the camera. This specific range was chosen to place the object with more accuracy during data collection. Additionally, the objects were

[13] http://wiki.ros.org/rosbag/Commandline

positioned at various distances, ranging from 1 to 21 meters from the camera.

3.2 Data Processing

The data processing phase of the project involves processing the collected data and preparing for further analysis and model training. This section presents the data processing steps and techniques employed.

3.2.1 Data extraction

Data extraction is the initial step in the data processing pipeline, aimed at retrieving the relevant information from the collected sources.

Thermal Images
The thermal images are initially stored in a ROS bag file. To process these images, they need to be extracted from the bag file and converted into OpenCV format for further processing. This is done using the bags2Images.py from Appendix A.0.1. The Python script reads messages from the specified image topic in the ROS bag file. A loop iterates through each message on the topic and uses the CvBridge package to convert the image message into an OpenCV image in cv_img format.

Once the thermal images have been extracted, they can be used for subsequent steps in the data processing pipeline, such as model training.

Data Labeling
Data labeling is a fundamental and crucial step in training a supervised machine learning model. It involves assigning appropriate labels to each data case, indicating the class or category of the object present. Accurate and reliable labeling is essential for teaching the model to recognize and differentiate objects during the training process.

In this project, the labelImg[14] tool was chosen for efficient data labeling. LabelImg provides a user-friendly graphical interface that simplifies the labeling process. It allows users to mark the objects of interest in images and assign them the corresponding labels. The software creates bounding boxes around the objects, as shown in Figure 3.8, and associates them with the respective class labels.

[14] https://github.com/heartexlabs/labelImg

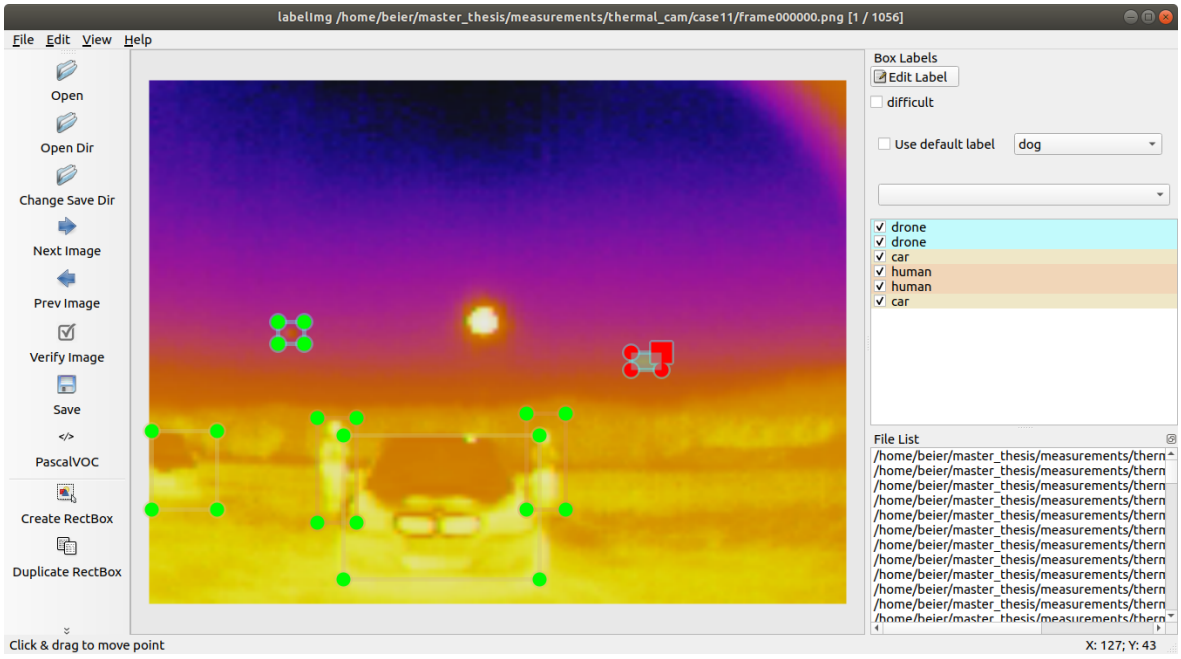

Figure 3.8: labelImg Gui showing two drones. two cars and two humans

The labeled data is then saved in Pascal VOC file format, containing information about the bounding boxes and labels for each object in the image. This labeled data serves as the ground truth for training the machine learning model and evaluating its performance.

3.2.2 Model Training

To train the YOLOv5 model a Python notebook file is utilized, following a guide[15] for custom object detection training using YOLOv5. The data set is prepared following practices in machine learning. This section describes the steps involved in training the model.

Configuring the Training Process

The training process of the YOLOv5 model requires setting several important parameters that govern its behavior and performance. These parameters determine how the model learns from the data and how it makes predictions. Table 3.1 presents the configuration parameters used in training the YOLOv5 model.

Table 3.1: Yolov5 configurations Parameters

Parameter	Value
Batch Size	16
Epochs	100
Number of classes	3
Momentum	0.937
Decay	0.005
GPU version	Tesla V100-SXM3-32GB

[15] https://learnopencv.com/custom-object-detection-training-using-yolov5/

The batch size determines the number of training samples processed in each iteration. Epochs represent the number of times the entire data set is passed through the model during training. The number of classes specifies the total number of object classes to be detected by the model. Momentum is a parameter that influences the optimization process by controlling the rate at which the model learns from previous updates. Decay represents the regularization term that helps prevent overfitting. Lastly, the GPU version indicates the specific GPU hardware used for training, in this case, a Tesla V100-SXM3-32GB.

Data Preparation

Before training the YOLOv5 model, the custom data set needs to be structured appropriately. The dataset is organized into three subsets: the training set, the validation set, and the testing set. These subsets play distinct roles in training and evaluating the model.

The training set contains 80% of the data and is used to train the YOLOv5 model. It provides the necessary examples for the model to learn the patterns and characteristics associated with each object class. The validation set, comprising 10% of the data, is used to fine-tune the model and optimize its performance. By evaluating the model's performance on this set, adjustments can be made to enhance its accuracy and generalization capabilities. The remaining 10% of the data is allocated to the testing set, which evaluates the model's performance on unseen data and assesses its ability to make predictions in real-world scenarios.

Splitting the data

The data set is split by using a Python script called moveRandom.py given in appendix A.0.1. This script randomly selects images and their corresponding label files and moves them into the correct folder. It begins by declaring the number of files to be moved and then determines which of the three different folders (train, validation, or test) each file should be placed in. Within each folder image files will be placed in the image folder and label files in the label folder. This organization allows the YOLOv5 algorithm to locate and process the necessary data during training and evaluation.

Converting Label Files

The label files in the original Pascal VOC format need to be converted to the YOLO format which is the required format for training the YOLOv5 model. A Python script is used to perform this conversion, following a guide that outlines the process[16]. The script handles the transformation of the bounding box coordinates and object labels into the YOLO format, enabling the YOLOv5 algorithm to effectively train and make predictions on the data.

Yaml file

A YAML file is created to provide the necessary information for the training script. This file contains the paths to the test and validation folders, enabling the training script to locate

[16] https://towardsdatascience.com/convert-pascal-voc-xml-to-yolo-for-object-detection-f969811ccba5

the data for training and evaluation. Additionally, the YAML file includes the names of the different classes present in the dataset, specifying the objects or entities that the model is expected to detect. In this case, the classes are 'human', 'car', and 'drone'.

```yaml
train: ../fullDataSet_yoloformat/train/images
val: ../fullDataSet_yoloformat/valid/images
test: ../fullDataSet_yoloformat/test/images

nc: 3
names: ['human','car','drone']
```

3.2.3 Model Evaluation

After training the model, the same notebook for training the data contains functions to perform inference on the data set and evaluate the model.

When this evaluation script is executed, the trained model processes the images in the test folder and generates a results folder. The results folder contains various outputs, including a confusion matrix, different graphs, and label files in YOLO format for the detected objects.

K-fold Cross Validation

To validate the model, a technique called K-fold cross-validation is employed. The model is trained where K is set to 5 in this case. This technique involves dividing the data set into K equally sized folds and performing training and validation K times, as illustrated in figure 3.9. The main idea behind K-fold cross-validation is that it can be measured as an unbiased estimation error.

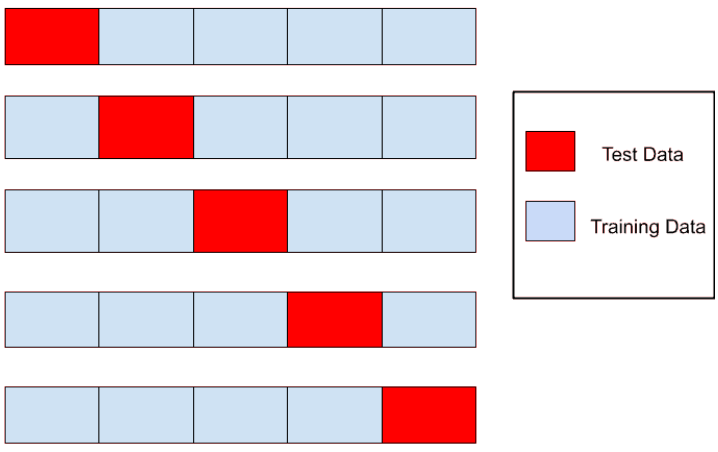

Figure 3.9: K-fold cross-validation. Image is taken from mltut.com [17]

[17] https://www.mltut.com/k-fold-cross-validation-in-machine-learning-how-does-k-fold-work/

3.2.4 Estimating position of object in the FOV from the camera

Once the YOLOv5 model has detected objects, it is possible to estimate their location within the camera's FOV. The results folder generated by the evaluation script contains label files with the detected objects in the YOLO format. To estimate the location within the FOV, a Python file is created, which includes three functions.
The 'result' folder contains label files with the objects found in YOLO format. master_functions.py A.0.1 is made to estimate the location of the object according to the camera's FOV. The Python file contains three functions. The image is 160x120, which means there is a 160-pixel split of 57 degrees in the x direction.

$$Onepixel = 160/57 = 2.8070° \tag{3.1}$$

Algorithm 1 Calculating the estimated position of the object according to the cameras FOV

1: **procedure** CONVERTING LABEL FILES
2: Receieves label file in YOLO format.
3: Calculates the center point of the bounding box into pixel coordinates.
4: Add the type of object and its pixel coordinates into an array.
5: **return** Array containing type of object and its corresponding pixel coordinates.
6: **end procedure**
7: **procedure** CONVERTING IMAGE
8: Receives image.
9: Converts the image into an array of pixel coordinates.
10: **return** Array containing pixel coordinates of Image.
11: **end procedure**
12: **procedure** CALCULATING POSITION OF OBJECT
13: Receives arrays from the above functions.
14: **procedure** ITERATES THROUGH THE ARRAY OF PIXEL COORDINATES.
15: **procedure** IF THE OBJECT'S CENTER POINT IS FOUND.
16: Checks if the center point is on the left or right side(Using the X value from the object).
17: Multiplies the number of pixels from the middle pixel with the pixel value.
18: **end procedure**
19: **end procedure**
20: **return** Position and type of object according to the camera's FOV.
21: **end procedure**

Chapter 4

Results and Analysis

This chapter presents the results and findings obtained during the project. It presents a pilot test and the analysis of the collected data. The chapter focuses on the performance evaluation of the system, including the classification and localization of objects.

4.1 Pilot test

A pilot test was conducted to assess the performance of the system and determine the sufficiency of the collected data. This was done by conducting a trial run of the measurement plan, with only a few cases collected. The objectives of the pilot test were to identify system limitations, address challenges, and make necessary improvements. This section provides a detailed examination of the pilot test, including the setup, observations, and implemented improvements.

Experimental Setup
The pilot test was conducted using two DJI Mini SE drones with a flight time of up to 30 minutes and a weight of 249 grams. The positioning of cars, humans, and drones followed the measurement plan described in Section 3.1.4. To ensure more accurate placement of objects, a FOV is marked on the ground, shown in Figure 4.3.

Figure 4.1: Experimental set-up for the pilot test.

Observations

The data collected during the pilot test provided valuable insights into the system's performance and identified specific challenges. The key observations are as follows:

1. The thermal camera encountered difficulties in detecting the drones due to the heat signature of the environment in the background. As an example in Figure 4.2 the location of the drone is marked.

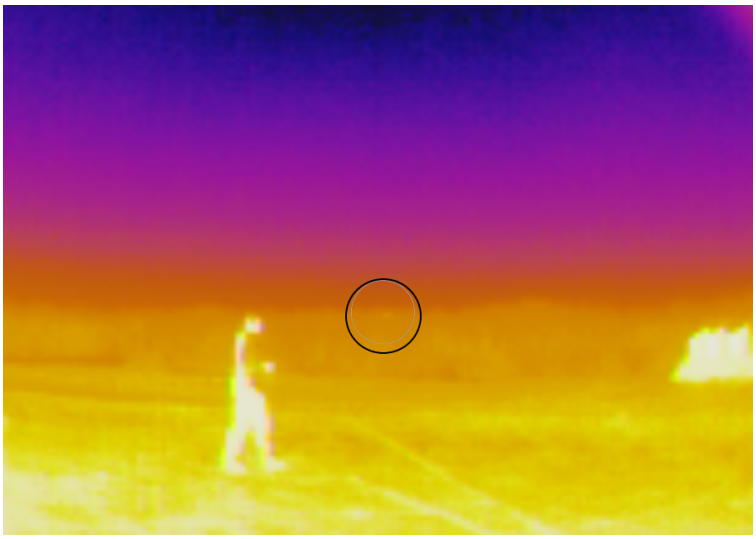

Figure 4.2: Thermal camera failing to detect drone

2. Drones positioned at a distance of 17-21 were challenging to detect.

3. The limited battery life of the drones was identified as a constraint, allowing only 8 measurements to be performed on a single battery cycle. Attempts were made to save battery life by turning off the drones during each case, keeping them in the air while object repositioning, and landing them on the ground while still powered on. However, these changes did not result in any improvements.

Implemented Improvements

Figure 4.3 illustrates the updated setup. Based on the observations made during the pilot test, several improvements were identified and implemented in the final system configuration:

1. Adjusting the camera view to prevent capturing heat signatures from nearby buildings that could interfere with detection.
2. Limiting the positioning of drones to a maximum distance of 15 meters to ensure optimal visibility and minimize detection challenges.
3. Increased the altitude of drones as they moved further back, ensuring they were always higher than any other objects.
4. Expanding the system to include two batteries, effectively doubling the number of measurements per session.

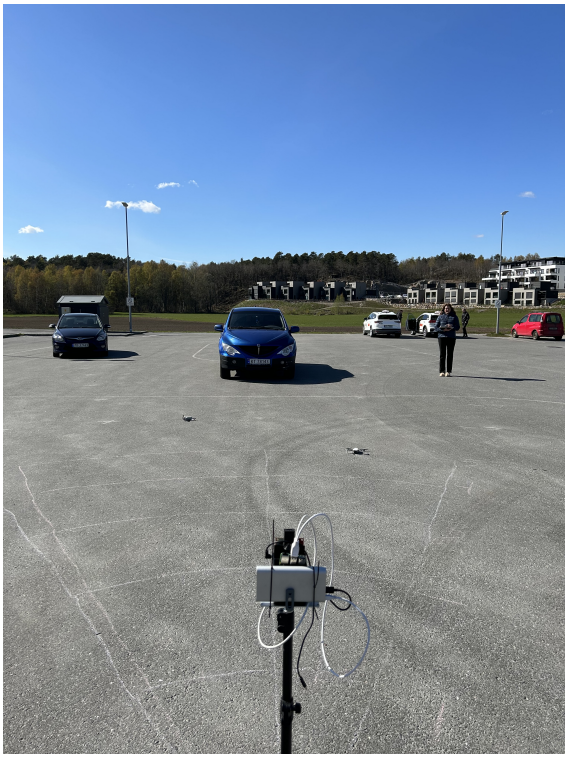

Figure 4.3: Final setup

4.2 Data set

The data set used for the model training consisted of 151.240 images, spread over 150 cases. In all cases, there was a count of 240 drones, 260 humans, and 298 cars.

4.3 Data Analysis

This section presents the main results obtained from training the YOLOv5 model with a custom data set. The analysis focuses on the classification and localization performance of

objects.

4.3.1 Classification Performance

The performance of the trained model using was assessed based on the detection accuracy and the ability to differentiate between the classes human, car, and drone. The evaluation of the detection performance was conducted using confusion matrices and F1 curves generated from the YOLOv5 algorithm, providing an overview of the true positive (TP), false positive (FP), true negative (TN), and false negative(FN) detections for each object class.

- TP: Refers to a situation where a predicted class detection is indeed a true class, and it is correctly identified.

- FP: Occurs when a predicted class detection is actually a false class, meaning it is incorrectly identified as a true class.

- TN: Describes a scenario where a predicted object detection is classified as a false class, and it is a false class.

- FN: Represents a situation where a predicted class detection is a false class, but it is mistakenly identified as a true class.

In Figure 4.4 the classification is visualized using the third fold. The visualization shows that the trained model is well able to detect and distinguish the different classes.

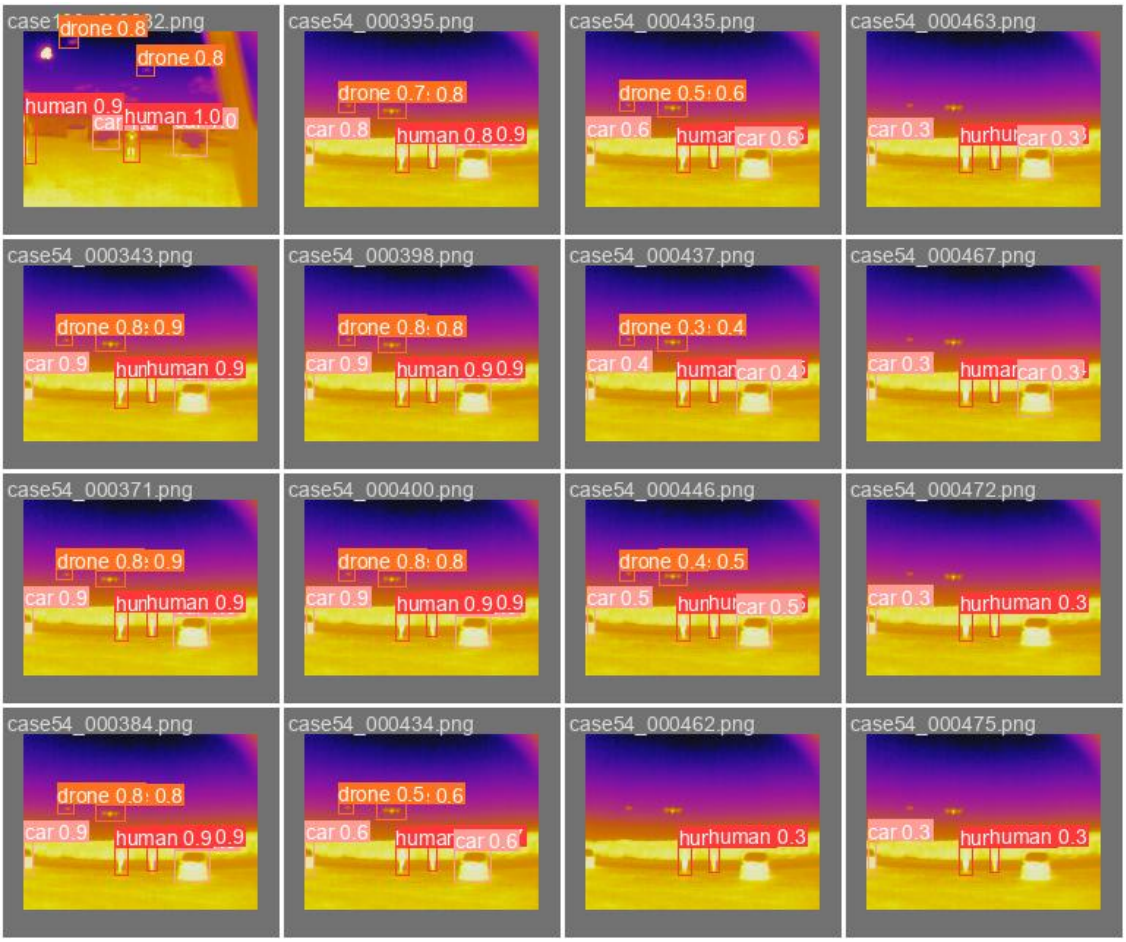

Figure 4.4: Validation batch for fold 3

A confusion matrix is represented with multiple classes from the first fold. This contains information about the precision of the classification. From the confusion matrix in Figure 4.5 it can be observed that the system achieved a high true positive rate for detecting humans, with 98% of true human detection's classified correctly. The FP rate for humans misclassified as cars were low and at 3%, indicating good differentiation between these object classes. However, there was a higher FP rate of 46% for humans classified as background. The FN for an object being both a car or a background is at 1%.

The performance of car detection has a true positive rate of 97%. The chances for a false positive rate for misclassified as human at 1% and background 42%. The false negative has a 3% chance of being classified as human.

For drones, the system achieved a true positive rate of 99%. The false positive rate for drones misclassified as the background was at 13% and a false negative at 1% for being the background.

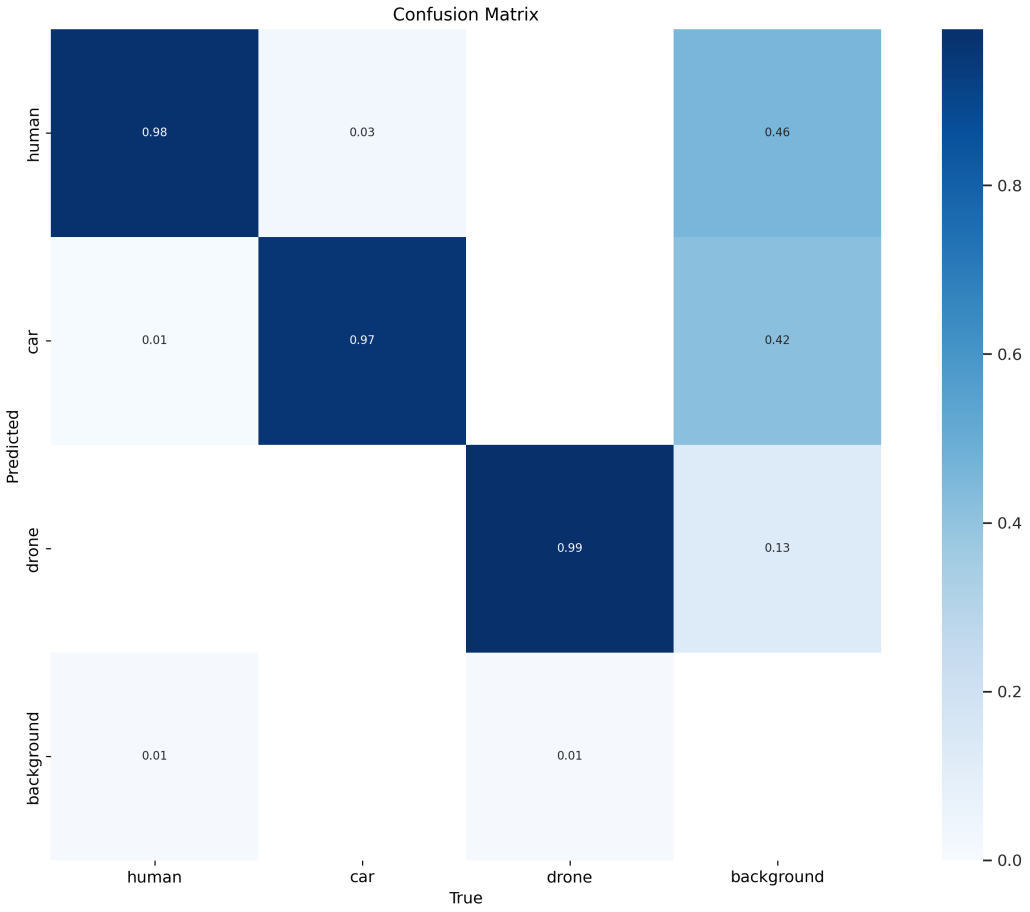

Figure 4.5: Confusion matrix for the first fold.

In addition to the confusion matrices, F1 curves were generated to visualize the performance of the system across different object classes. These curves are presented in Figure 4.6 and combine the precision and recall value of the classification. The classification keeps a steady F1 value with good confidence. The F1 score for drones falls a bit earlier around a confidence of 0.7 and the F1 score keeps more steady when it is either a car or a human.

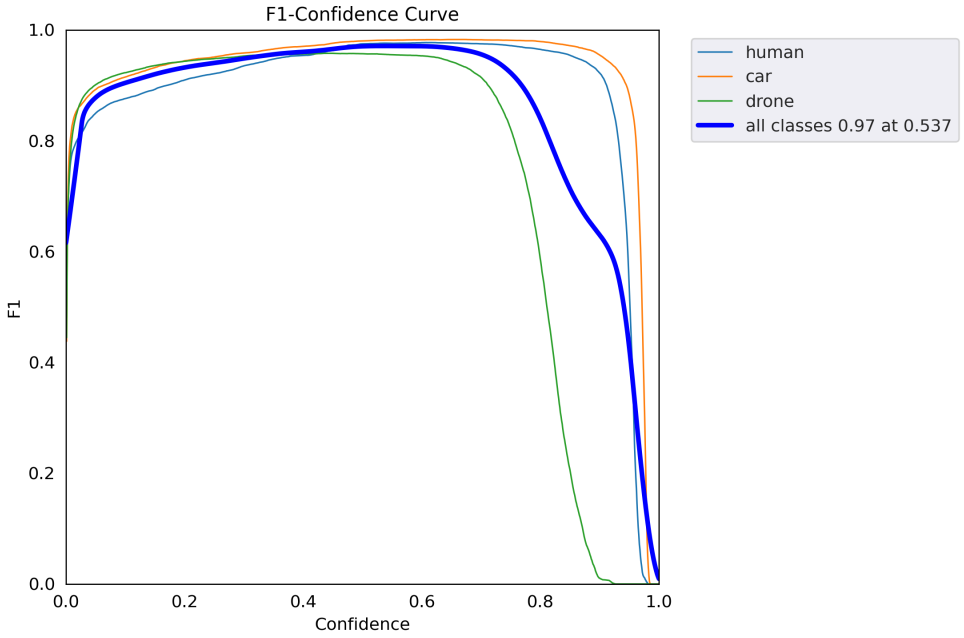

Figure 4.6: F1 Curve for the first fold.

The figures used in this section are just the results from the training of the first fold. While looking at the results from the other four folds it was noticed that it is small differences in the FP and the FN rates.

4.3.2 Localization Performance

The localization performance is evaluated based on the estimation of object positions within the camera's FOV (-28.5 to 28.5°).

To assess the performance, the five-folds used in training are used to compare the model's accuracy, and the results are presented in Tables 4.1, 4.2, 4.3. The cases have been randomly selected and compared against the measurement plan for the ground truth. To represent the accuracy of the predicted localization the error is calculated using Root Mean Square Error(RMSE). RMSE measures the difference between the estimated position of the object with the true position.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{d_i - f_i}{\sigma_i}\right)^2} \tag{4.1}$$

In Table 4.1, the localization of humans is evaluated for five different cases using the trained model from each of the five folds. The table shows that the error in estimating the positions of humans ranges from 1° and 4°.

Table 4.1: K-fold cross-validation for detected Humans

K-fold→ Case↓	$\theta_{\mathbf{est1}}$	$\theta_{\mathbf{est2}}$	$\theta_{\mathbf{est3}}$	$\theta_{\mathbf{est4}}$	$\theta_{\mathbf{est5}}$	$\theta_{\mathbf{true}}$	RMSE
Case 11	-12.82°	-12.82°	-12.82°	-13.18°	-12.82°	-15.00°	2.11°
Case 13	4.28°	3.92°	3.92°	3.56°	3.92°	5.00°	1.10°
Case 15	11.04°	11.04°	11.04°	11.04°	11.04°	15.00°	3.96°
Case 40	-16.74°	-16.74°	-16.74°	-16.74°	-16.74°	-20.00°	3.26°
Case 83	-2.85°	-3.21°	-2.85°	-2.85°	- 2,85°	-5.00°	2.09°

Table 4.2 presents the results for object localization of cars. Similarly to human localization, the estimated positions of cars exhibit an error ranging from 1° and 2.5°, indicating a slightly higher variability compared to humans.

Table 4.2: K-Fold cross-validation for detected Cars

K-fold→ Case↓	$\theta_{\mathbf{est1}}$	$\theta_{\mathbf{est2}}$	$\theta_{\mathbf{est3}}$	$\theta_{\mathbf{est4}}$	$\theta_{\mathbf{est5}}$	$\theta_{\mathbf{true}}$	RMSE
Case 14	3.92°	3.92°	3.92°	3.92°	3.92°	5.00°	1.08°
Case 50	9.62°	9.62°	9.62°	9.62°	9.62°	10.00°	0.38°
Case 16	4.28°	4.28°	4.28°	4.28°	3.92°	5.00°	0.81°
Case 83	-24.58°	-24.58°	-24.58°	-24.58°	-24.58°	-30.00°	0.42°
Case 135	-17.46°	-17.46°	-17.1°	-17.46°	-17.1°	-15.00°	2.32°

In Table 4.3, the localization performance for drones is shown. The estimated positions of drones exhibit a higher error ranging from 1° to 4°, indicating a larger discrepancy between the predicted and true positions.

Table 4.3: K-Fold cross-validation for detected Drones

K-fold→ Case↓	$\theta_{\mathbf{est1}}$	$\theta_{\mathbf{est2}}$	$\theta_{\mathbf{est3}}$	$\theta_{\mathbf{est4}}$	$\theta_{\mathbf{est5}}$	$\theta_{\mathbf{true}}$	RMSE
Case 10	12.47°	11.76°	12.11°	12.47°	12.11°	15.00°	2.83°
Case 11	-16.64°	-16.64°	-16.64°	-16.64°	-16.74°	-20.00°	3.36°
Case 13	11.76°	11.04°	11.4°	11.76°	11.76°	15.00°	3.47°
Case 38	-5.7°	-6.06°	-6.41°	-6.41°	-6.06°	-10.00°	3.88°
Case 61	-8.91°	-8.91°	-8.91°	-8.91°	-8.91°	-10.00°	1.09°

Chapter 5

Discussions

This chapter provides an analysis and discussion of the system's performance in terms of classification accuracy and localization of objects within the camera's field of view (FOV).

5.1 Classification Performance

Confusion Matrix
The confusion matrix in Figure 4.5 was used as a tool to measure the accuracy and precision of the trained model. It revealed that the system had some FP detections when classifying humans and drones, with 46% change of human beings classified as background and 13% for drones. This is mainly because of the background noise and disturbance in the parking lot where the collection took place. Inaccuracies in data labeling can contribute to misclassification, such as mistaking clouds for drones.

F1 Curve
The F1 curve in Figure 4.6 visualized the confidence and precision of the trained model for each class. It showed that the precision dropped faster with drones compared to humans and cars. This emphasized the challenges encountered in accurately classifying drones, particularly when they were mixed with clouds in the background. This affects the model's precision.

In some cases humans and cars are placed close to each other, resulting in one of the other block parts of each other. When there are two or more persons labeling the data, labeling might be solved differently. This will have an effect on the precision of the model and is an area where consistency is important for the training model. It is important if there is more people labeling, that situations like this are clarified upfront. This will be taken as a lesson for later work.

5.2 Localization Performance

The obtained results of localization of classes were at a satisfactory accuracy, as indicated by the RMSE values presented in Tables 4.1, 4.3, and 4.2. The RMSE values in degrees were relatively small for all classes, suggesting good accuracy in localizing objects within the FOV

However, there is still room for improvement in terms of further reducing the RMSE and overall localization accuracy. One factor contributing to the variation in RMSE is the size of the detected class. Cars, being relatively larger in size, exhibited smaller errors in estimation compared to humans and drones. When labeling the bounding boxes should be as close to the objects as possible, which made it harder to clearly label the drones. It is also important to note that the manual creation of the grid markings on the ground for the measurement plan is not perfectly accurate in degrees and distance. Additionally, the environmental factors and rotation in the setup during data collection caused errors in the accuracy of the measurement plan. Figure 5.1 illustrates an example where the car on the right was originally placed at the edge (30 degrees) according to the measurement plan, but due to rotation, it was captured at a different angle.

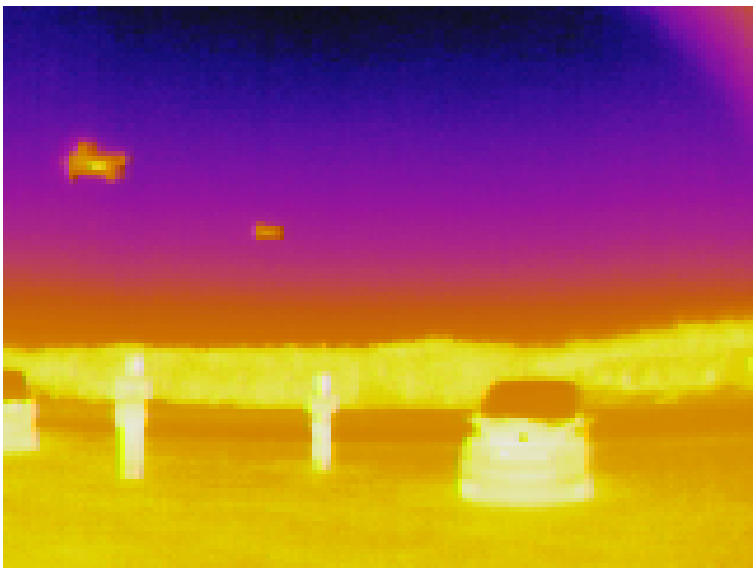

Figure 5.1: Case 49, all objects detected

All images were carefully inspected and compared with the measurement plan. In cases such as case 49, where rotations occurred, adjustments were made to the measurement plan, but there will still be some errors in the accuracy.

5.3 Detection and Localization implications

In addition to the system's performance analysis, it is essential to consider the implications encountered during the data collection.

5.3.1 Number of classes per case

One significant challenge was the loss of instances, particularly in the drone class. During the pilot test drones were observed to be the most challenging class to detect. They could be mixed up with clouds on overcast days. They could get lost in the heat signature by houses and trees in the background. In an attempt to prevent the inference of heat signature, drones were flown at a higher altitude, which in some cases resulted in them being placed outside the FOV. According to the data set, described in Section 4.2, there was a total loss of 60 drone instances.

Humans also experienced a noticeable loss of instances even though it is not as big as with drones. This loss occurred when the human was placed at +/- 30° degrees, and the setup had been rotated. The mounting of the sensors was not always static, especially during windy conditions. This caused slight rotations of the setup, resulting in classes positioned at the edge of the FOV being partially or fully outside the captured images, leading to the loss of 40 human instances. Two cars were also lost due to similar reasons.

5.3.2 Quality of the Collected Data

Despite these limitations, the data collected still possessed valuable qualities. The thermal images obtained from the parking lot demonstrated clear visibility and high contrast, as shown in Figure 5.1. The figure also shows the issue mentioned about background noise. Going back to the confusion matrix in section 4.3.1 that says it is a 46% chance of an FP, this is a good example that shows how the human can be mixed in the background.

Chapter 6

Conclusion

The thesis explored four research objectives, and the first objective is to integrate a thermal camera and mmWave radar. This has been successfully integrated using a Raspberry Pi 4 model B and ROS melodic with the necessary packages. The integrated system was then used to reach the second objective by collecting a data set. When the data was collected the data has been labeled and fed into the YOLOv5 algorithm for training. The trained model completes the third objective by being able to differentiate UAVs from humans and cars. In combination with the trained model, a Python script is used to complete the fourth and final objective by localizing objectives in the FOV of the camera.

6.1 Implication

While the system demonstrated good accuracy in detecting objects and classifying them, there are certain limitations that should be considered:

1. **Detection Range:** The system's effectiveness in detecting objects may be limited at longer distances. In the pilot test, drones positioned beyond 15 meters proved to be challenging to detect.
2. **Thermal Camera Limitations:** The thermal camera used in the system had some difficulties in detecting objects with similar heat signatures to the background environment. This limitation was observed in the pilot test and was confirmed in the confusion matrix.

6.2 Future Work

Currently, the radar data collected remain unused. The incorporation of this data into the existing system has the potential to bring significant improvements and advancements. One theoretical aspect to explore is the use of radar data for filtering out background noise. By combining the information from the thermal camera and radar, it becomes possible to distinguish genuine objects from false positives. E.g. if the clouds are identified as a drone, the radar would say that there is no object at that particular position. This would make the system more reliable.

Furthermore, the radar data can enable distance estimation making it possible to determine the distance between the system and the detected object. This information makes a better understanding of the relationship between objects and extends the detection range.

Appendix A

Appendix A: PDF

A.0.1 Measurement_Plan.pdf

Appendix B: Code

A.0.2 master_functions.py

A.0.3 move_random.py

A.0.4 prepareDataset.py

A.0.5 xml2yolo.py

A.0.6 bags2Images.py

Appendix C: Result

In the following folders, the figures given from training the different folds are given.

A.0.7 Fold 1

A.0.8 Fold 2

A.0.9 Fold 3

A.0.10 Fold 4

A.0.11 Fold 5

Bibliography

[1] Jabran Akhtar and Karl Erik Olsen. "GO-CFAR trained neural network target detectors." In: *2019 IEEE Radar Conference (RadarConf)*. IEEE. 2019, pp. 1–5.

[2] RK Bhan et al. "Uncooled infrared microbolometer arrays and their characterisation techniques." In: *Defence Science Journal* 59.6 (2009), p. 580.

[3] Chris M. Bishop. "Neural networks and their applications." In: *Review of Scientific Instruments* 65.6 (June 1994), pp. 1803–1832. ISSN: 0034-6748. DOI: 10.1063/1.1144830. eprint: https://pubs.aip.org/aip/rsi/article-pdf/65/6/1803/8387807/1803_1_online.pdf. URL: https://doi.org/10.1063/1.1144830.

[4] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[5] Daniel Brodeski, Igal Bilik, and Raja Giryes. "Deep radar detector." In: *2019 IEEE Radar Conference (RadarConf)*. IEEE. 2019, pp. 1–6.

[6] Daniel Brodeski, Igal Bilik, and Raja Giryes. "Deep radar detector." In: *2019 IEEE Radar Conference (RadarConf)*. IEEE. 2019, pp. 1–6.

[7] GA Cardona, D Tellez-Castro, and E Mojica-Nava. "Cooperative transportation of a cable-suspended load by multiple quadrotors." In: *IFAC-PapersOnLine* 52.20 (2019), pp. 145–150.

[8] Gustavo A Cardona and Juan M Calderon. "Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations." In: *Applied Sciences* 9.8 (2019), p. 1702.

[9] Frank K.Y. Chan and James Y.L. Thong. "Acceptance of agile methodologies: A critical review and conceptual framework." In: *Decision Support Systems* 46.4 (2009). IT Decisions in Organizations, pp. 803–814. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2008.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0167923608002133.

[10] K Cheikh and Faozi Soltani. "Application of neural networks to radar signal detection in K-distributed clutter." In: *IEE Proceedings-Radar, Sonar and Navigation* 153.5 (2006), pp. 460–466.

[11] Jia Deng et al. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[12] "Detection of tiger puffer using improved YOLOv5 with prior knowledge fusion." In: *Information Processing in Agriculture* (2023). ISSN: 2214-3173. DOI: https://doi.org/10.1016/j.inpa.2023.02.010.

[13] Mohammad Emadi. "Radar Technology." In: *Advanced Driver Assistance Systems and Autonomous Vehicles: From Fundamentals to Applications*. Ed. by Yan Li and Hualiang Shi. Singapore: Springer Nature Singapore, 2022, pp. 265–304. ISBN: 978-981-19-5053-7. DOI: 10.1007/978-981-19-5053-7_9. URL: https://doi.org/10.1007/978-981-19-5053-7_9.

[14] FLIR. *FLIR LEPTON® Engineering Datasheet*. https://cdn.sparkfun.com/assets/f/6/3/4/c/Lepton_Engineering_Datasheet_Rev200.pdf. 2018.

[15] Raspberry Pi Foundation. *About Us*. https://www.raspberrypi.org/about/.

[16] Raspberry Pi Foundation. *Empowering young people to use computing technologies to shape the world*. https://www.raspberrypi.org/. 2023.

[17] Gaspare Galati, Fausto Marti, and Fabrizio Rocci. "Generation of Radar Images of Aircraft for Design and Test of Image Processing Algorithms in Radar Applications." In: *Advanced Video-Based Surveillance Systems*. Ed. by Carlo S. Regazzoni, Gianni Fabri, and Gianni Vernazza. Boston, MA: Springer US, 1999, pp. 213–222. ISBN: 978-1-4615-5085-3. DOI: 10.1007/978-1-4615-5085-3_19. URL: https://doi.org/10.1007/978-1-4615-5085-3_19.

[18] Prashant P Gandhi and Viswanath Ramamurti. "Neural networks for signal detection in non-Gaussian noise." In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2846–2851.

[19] Zoubin Ghahramani. "Unsupervised Learning." In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112. ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_5. URL: https://doi.org/10.1007/978-3-540-28650-9_5.

[20] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[21] Jiuxiang Gu et al. "Recent advances in convolutional neural networks." In: *Pattern recognition* 77 (2018), pp. 354–377.

[22] Malcolm L. Heron, William G. Pichel, and Scott F. Heron. "Radar Applications." In: *Coral Reef Remote Sensing: A Guide for Mapping, Monitoring and Management*. Ed. by James A. Goodman, Samuel J. Purkis, and Stuart R. Phinn. Dordrecht: Springer Netherlands, 2013, pp. 341–371. ISBN: 978-90-481-9292-2. DOI: 10.1007/978-90-481-9292-2_13. URL: https://doi.org/10.1007/978-90-481-9292-2_13.

[23] Texas Instruments. *IWR1642 Evaluation Module (IWR1642BOOST) Single-Chip mmWave Sensing Solution*. https://www.ti.com/lit/ug/swru521c/swru521c.pdf?ts=1679365553662&ref_url=https%253A%252F%252Fdev.ti.com%252F. 2020.

[24] P. Jarabo-Amores et al. "A neural network approach to improve radar detector robustness." In: *2006 14th European Signal Processing Conference*. 2006, pp. 1–5.

[25] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects." In: *Science* 349.6245 (2015), pp. 255–260. DOI: 10.1126/science.aaa8415. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8415. URL: https://www.science.org/doi/abs/10.1126/science.aaa8415.

[26] Krati Joshi. *What is Classification in Machine Learning and Why is it Important?* https://emeritus.org/blog/artificial-intelligence-and-machine-learning-classification-in-machine-learning/. 2023.

[27] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey." In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[30] Tsung-Yi Lin et al. "Focal loss for dense object detection." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[31] Yeray Mezquita et al. "A Review of k-NN Algorithm Based on Classical and Quantum Machine Learning." In: *Distributed Computing and Artificial Intelligence, Special Sessions, 17th International Conference*. Ed. by Sara Rodríguez González et al. Cham: Springer International Publishing, 2021, pp. 189–198.

[32] Morgan Quigley et al. "ROS: an open-source Robot Operating System." In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[33] 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*. 2016, pp. 779–788.

[34] Budiman P.A. Rohman, Dayat Kurniawan, and M. Tajul Miftahushudur. "Switching CA/OS CFAR using neural network for radar target detection in non-homogeneous environment." In: *2015 International Electronics Symposium (IES)*. 2015, pp. 280–283. DOI: 10.1109/ELECSYM.2015.7380855.

[35] Seungbo Shim and Gye-Chun Cho. "Lightweight semantic segmentation for road-surface damage recognition based on multiscale learning." In: *IEEE Access* 8 (2020), pp. 102680–102690.

[36] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[37] Tomasz Sosnowski, Grzegorz Bieszczad, and Henryk Madura. "Image Processing in Thermal Cameras." In: *Advanced Technologies in Practical Applications for National Security*. Ed. by Aleksander Nawrat, Damian Bereska, and Karol Jędrasiak. Cham: Springer International Publishing, 2018, pp. 35–57. ISBN: 978-3-319-64674-9. DOI: 10.1007/978-3-319-64674-9_3. URL: https://doi.org/10.1007/978-3-319-64674-9_3.

[38] Ningyuan Su et al. "Deep CNN-based radar detection for the real maritime target under different sea states and polarizations." In: *Cognitive Systems and Signal Processing: 4th International Conference, ICCSIP 2018, Beijing, China, November 29-December 1, 2018, Revised Selected Papers, Part II*. Springer. 2019, pp. 321–331.

[39] Yunjia Sun, Edward Lank, and Michael Terry. "Label-and-Learn: Visualizing the Likelihood of Machine Learning Classifier's Success During Data Labeling." In: *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. IUI '17. Limassol, Cyprus: Association for Computing Machinery, 2017, pp. 523–534. ISBN: 9781450343480. DOI: 10.1145/3025171.3025208. URL: https://doi.org/10.1145/3025171.3025208.

[40] Christian Szegedy et al. "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[41] Sasha Targ, Diogo Almeida, and Kevin Lyman. "Resnet in Resnet: Generalizing Residual Architectures." In: *arXiv e-prints*, arXiv:1603.08029 (Mar. 2016), arXiv:1603.08029. DOI: 10.48550/arXiv.1603.08029. arXiv: 1603.08029 [cs.LG].

[42] *Thermal Imaging Camera*. https://www.omega.com/en-us/resources/thermal-imagers.

[43] Zhi Tian et al. "Fcos: Fully convolutional one-stage object detection." In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9627–9636.

[44] Jan Verstockt et al. "Skin Cancer Detection Using Infrared Thermography: Measurement Setup, Procedure and Equipment." In: *Sensors* 22.9 (2022). ISSN: 1424-8220. DOI: 10.3390/s22093327. URL: https://www.mdpi.com/1424-8220/22/9/3327.

[45] Li Wang, Jun Tang, and Qingmin Liao. "A study on radar target detection based on deep neural networks." In: *IEEE Sensors Letters* 3.3 (2019), pp. 1–4.

[46] *Why ROS?* https://www.ros.org/blog/why-ros/.