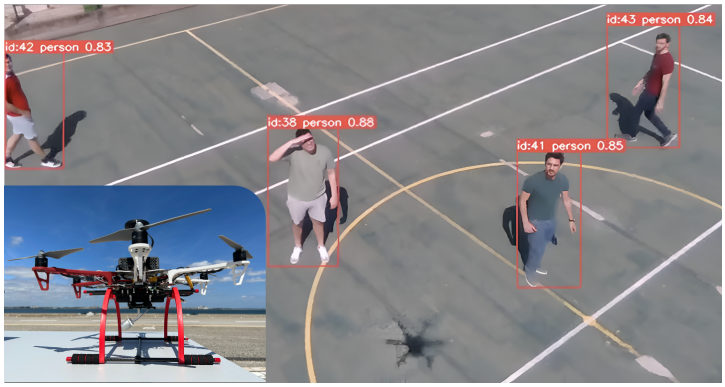
TÉCNICO LISBOA



Leveraging Multi-Object Tracking in Vision-based Target Following for Unmanned Aerial Vehicles

Diogo Costa Ferreira

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor: Dr. Meysam Basiri

Examination Committee

Chairperson: Prof. Paulo Sérgio De Brito André
Supervisor: Dr. Meysam Basiri
Member of the Committee: Prof. Alberto Manuel Martinho Vale

July 2024

”Always tell the truth. Always take the high road.
Live each day like it could be your last. Drink it in.
Be adventurous, be bold, but savor it. It goes fast.”
- Ben, Captain Fantastic

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First, I would like to express my appreciation to my advisor Dr. Meysam Basiri for the opportunity to work in such a challenging hands on project related to UAVs, one of my passions that I discovered in IST. I am grateful for the continuous encouragement to push the limits of my knowledge. I am also thankful for the opportunity to participate in ICARSC2024 where I was able to publish some of this work, engage with peers and learn a great deal about the world of robotics. I thank LARSyS (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIDP/50009/2020 and 10.54499/UIDB/50009/2020) and Aero.Next project (PRR - C645727867-00000066) that funded part of this research.

Second, I'm extremely grateful to the Portuguese Navy, namely Comandante Mourinha, Capitão-tenente Veloso e Segundo Tenente Gonçalves from CEOM and to CEOV. Their help was essential to be able to fully test the developed system in several real-world scenarios. I am also thankful to AEIST for giving me access to the football field where the remaining of the experiments were performed.

Third, I want to give recognition to AeroTéc and the project ATLAS for not only letting me borrow several batteries and the battery charger, but also for the incredible 3 years that I spent there as an associate. As a young student in IST I joined an emerging project called "UAV-ART" that would spike my interest in UAVs and lead me to where I am today. The opportunity to see that small project grow into ATLAS - the most successful aircraft student project in Portugal and being able to lead it while doing this thesis was a great challenge but nonetheless, the most incredible learning opportunity in my academic journey. Thank you to Pinto, Luís, Afonso and Simão for the trust and mentorship.

Fourth, I want to thank everyone that helped me in this thesis carrying things or just being humans for my tests. In particular to my friend Afonso Vale for not only joining me in the crazy adventure that was our journey in AeroTéc, through UAV-ART and ATLAS, but also for providing support for this thesis as my flight safety pilot. His unending availability and expertise were invaluable for this work.

This endeavor would not have been possible without my friends that made Lisbon a second home, when my first one was very far away. Thank you João, Jorge, Bárbara, Chaymaa, Tati, Larissa, Sara, Marcus and Luís and to my cousin Filipa for the good times we shared. To my friend Filipe that showed me how fun Lisbon can be. Special notes to André and Cochicho for the incredible moments we spent together that turned my stay in IST into a never-ending comedy show filled with joy and caffeine. I am really glad to have met you and shared this experience with you. To my girlfriend Mariana, thank you for the companionship, for sharing with me the best moments and lifting me up in my lowest ones; for being there when I doubted me and providing me reassurance, I am deeply grateful to you. Also, my appreciation to my SuperTux friends from back home Pedro, Helena, Beatriz, Rocha, Tomás, Jessica, Gil, Luís, Miguel and Ricardo, that never forgot me despite my rare appearances.

Finally, I want to give my greatest thanks to my family for believing in me throughout this entire journey. To my grandparents who always give me the most praise for the smallest of things and to my godmother and my cousins that are always very happy to see me. To my parents that made me who I am today by always working the hardest to provide me with every opportunity possible. And lastly to my little sister Joana, for giving me a reason to strive for being the best role model possible for her.

Resumo

Esta dissertação apresenta um sistema autónomo de rastreio e seguimento de alvos baseado em visão, concebido para Veículos Aéreos Não Tripulados (UAVs). A aplicação de métodos recentes de Rastreamento de Objetos Múltiplos (MOT) em cenários de seguimento de alvos é testada, em vez dos algoritmos tradicionais de Rastreamento de Objeto Único (SOT), explorando esta lacuna no estado-da-arte. O sistema integra o modelo de deteção de objetos em tempo real, You Only Look Once (YOLO)v8, com algoritmos MOT BoT-SORT e ByteTrack, extraindo informações sobre múltiplos alvos. Com estas informações, as capacidades de redetecção são melhoradas, resolvendo os maiores desafios destes algoritmos como alterações de ID dos alvos e oclusões. Um módulo de medição de profundidade é incorporado para melhorar a estimativa da distância, quando disponível. Quando não há informação de profundidade, o tamanho do alvo na imagem é utilizado. Para o seguimento, é proposto um sistema de controlo em 3D, capaz de reagir às mudanças de velocidade e direção do alvo, mantendo-o em vista. O sistema é testado em simulação e implementado em cenários reais num UAV montado de raiz nesta tese. Os resultados mostram um seguimento preciso do alvo, resistente a oclusões e distinguindo eficazmente o alvo seguido dos restantes. Comparações entre os algoritmos BoT-SORT e ByteTrack revelam um compromisso entre precisão e eficiência computacional, favorecendo o BoT-SORT devido à redução significativa do número de alterações de ID. Com as contribuições apresentadas, esta tese possibilita novas aplicações de seguimento de alvos a partir de UAVs, aproveitando informação sobre múltiplos alvos.

Palavras-chave: Veículo Aéreo Não Tripulado, YOLOv8, Rastreamento de Objetos Múltiplos, BoT-SORT, ByteTrack, Seguimento de Alvos Móveis.

Abstract

This thesis presents an autonomous vision-based mobile target tracking and following system designed for Unmanned Aerial Vehicles (UAVs) leveraging multi-target information. It explores the research gap in applying the most recent Multi-Object Tracking (MOT) methods in target following scenarios over traditional Single-Object Tracking (SOT) algorithms. The system integrates the real-time object detection model, You Only Look Once (YOLO)v8, with the MOT algorithms BoT-SORT and ByteTrack, extracting multi-target information. The system leverages this multi-target information to improve redetection capabilities, addressing key challenges such as target miss-identifications (ID changes), and partial and full occlusions in dynamic environments. A depth sensing module is incorporated to enhance distance estimation when feasible. When depth information is not available, the target size in the image is used instead. A 3D flight control system is proposed for target following, capable of reacting to changes in target speed and direction while maintaining line-of-sight. The system is initially tested in simulation and then deployed in real-world scenarios within a UAV platform assembled from scratch for this thesis. Results show precise target tracking and following, resilient to partial and full occlusions in dynamic environments, effectively distinguishing the followed target from bystanders. A comparison between the BoT-SORT and ByteTrack trackers reveals a trade-off between computational efficiency and tracking precision, with a preference for BoT-SORT due to its substantial decrease in ID changes in target tracking. In overcoming the presented challenges, this thesis enables new practical applications in the field of vision-based target following from Unmanned Aerial Vehicles (UAVs) leveraging multi-target information.

Keywords: Unmanned Aerial Vehicle, YOLOv8, Multi-Object Tracking, BoT-SORT, ByteTrack, Mobile Target Following;

Contents

List of Tables

List of Figures

Acronyms

C2f	CSP Bottleneck with 2 Convolutions - Fusion
CNN	Convolutional Neural Network
CSP	Cross Stage Partial
D-SSD	Deconvolutional Single Shot Detector
DNN	Deep Neural Networks
FCU	Flight Control Unit
HOG	Histogram of Oriented Gradients
IBVS	Image-Based Visual Servoing
IoU	Intersection over Union
KCF	Kernelized Correlation Filter
MOT	Multi-Object Tracking
PANet	Path Aggregation Network
PI	Proportional Integrative
PID	Proportional Integrative Derivative
PN	Proportional Navigation
ROS	Robot Operating System
SIFT	Scale-Invariant Feature Transform
SOT	Single Object Tracking
SPP	Spatial Pyramid Pooling
SPPF	Spatial Pyramid Pooling - Fast
SSD	Single Shot Detector
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once

Chapter 1

Introduction

In this chapter, an overview of the potential of Unmanned Aerial Vehicles (UAVs) in vision-based mobile target tracking and following is provided along with the motivation for continuous improvements in this field of research. The proposed solution is shown as well as the contributions that this thesis presents for the Aerospace Engineering field.

1.1 Context and Motivation

Airborne information is one of the best and fastest ways to assess an open environment and, more than ever, UAVs are becoming an essential, valuable and reliable tool to quickly and efficiently provide it. Recently, the ability of UAVs to perform autonomous tasks has increased the need for higher quality and quantity of information, which can be a determining factor in critical decision making during operations. This work will focus on one of these autonomous tasks: vision-based mobile target tracking and following using UAVs.

The ability to track and follow moving targets from the air has numerous applications, including search and rescue [1], human pursuit in police cases [2], aerial photography [3], ship tracking [4], and vehicle tracking [5]. In these applications, dynamic scenarios with multiple visible targets are expected. Being able to retrieve and process this information can be crucial to enhance the capacity to continuously follow the desired target or even dynamically determine which target is most suitable to follow for the success of the mission. The ability to retrieve multi-target information enables many of these potential applications using mobile target tracking and following from a UAV. Therefore, it is vital that the systems deployed have the capability of not only tracking and following the target of interest, but are also able to be aware of other objects of interest in the image.

The problem of vision-based mobile target tracking and following usually encompasses the detection of the object in the image, followed by a target tracking algorithms that monitors the position of the object over time in the image. Finally, the relative position of the object in relation to the UAV is computed to give feedback to the controller, in order to allow target following. Currently there is a divide between two different strategies that perform target tracking in the image: Single Object Tracking (SOT) approaches,

where only the object of interest is detected and tracked in the image and Multi-Object Tracking (MOT) approaches, where the system detects and tracks all the objects of interest in the image and assigns a specific ID to each of them.

Many works have been developed in the field of vision-based mobile target tracking and following by UAVs, however, the most accepted approach is to tackle the tracking step of the problem by using SOT. These approaches only perform the detection step in the initial frame, which is usually done manually. Although they provide an efficient way to perform the target tracking and following task, they severely limit the system's knowledge of other objects of interest in the image which can harm the redetection capabilities if the target is under occlusions in a dynamic environment. Hence, they show clear limitations in terms of accurate position estimation of the target, handling dynamic environments, and managing occlusions.

To take advantage of multi-target information, it is necessary to apply MOT methods. These algorithms handle dynamic multi-target environments but struggle with camera motion and view changes provoked by the UAV movement. Recently, some works attempt to address this issue by improving camera motion models [6, 7]. Additionally, MOT algorithms are more computationally demanding, having to process multiple detections in each frame. Recent improvements in deep learning methods show that they are becoming more reliable for real-time applications, making them appealing for the detection step [8]. In particular, the detection algorithm You Only Look Once (YOLO)v8 is the new iteration of the YOLO family and proves to be faster and more accurate than the previous versions [9]. This deep-learning approach can be an improvement on existing approaches as suggested per Liu et al. [10].

Despite these advancements, real-world experimentation with MOT algorithms has predominantly been confined to fixed-area monitoring without camera movement. Consequently, there is a significant research gap regarding the practical application of MOT algorithms in UAV-based target tracking and following. This thesis aims to bridge this gap through rigorous real-world validation. New state-of-the art MOT algorithms show improvement in camera motion corrections, enabling exciting new applications in target following scenarios. MOT applications also have the advantage of working under a tracking-by-detection approach which performs a detection step followed by a tracking step, in every frame. This approach allows for the consistent use of new the new deep-learning based detection methods such as YOLOv8 in order to increase the reliability of the system as a whole.

Taking into consideration the recent developments in detection and tracking algorithms, this thesis proposes the use of MOT algorithms and the multi-target information they provide in a vision-based target tracking and following Unmanned Aerial System (UAS). The system combines a state-of-the-art detector, YOLOv8, with the recent MOT algorithm BoT-SORT and compares the results with its predecessor ByteTrack. It is believed that by leveraging this new information, new and exciting applications for these systems can be unlocked, thus further propelling the research in this field.

Target following algorithms still require improvements in the estimation of the distance to the target. The limitation of precise target positioning is still present in many works, which could benefit from a more accurate position estimation method. Incorporating a depth estimator module is an interesting improvement that can solve this issue [10]. Hence, in this work a two step distance estimation approach

will be taken by combining the visual information with depth information from an RGB-D camera for better distance estimation accuracy in dynamic environments.

1.2 Proposed solution

First, this work integrates a detection and tracking algorithm combining the YOLOv8 detection algorithm with both the BoT-SORT and the ByteTrack MOT algorithms into a comprehensive system that provides multi-target information for vision-based mobile target tracking and following using a UAV. By keeping track of the identities of all possible targets in the image, the system is able to accurately differentiate between the followed target and similar objects. The additional information is used to improve redetection algorithms in dynamic scenarios, making the system robust to partial and full occlusions.

Second, an estimation method is implemented to calculate the distance from the UAV to the target. In addition to an estimation method based on visual information, a depth camera is installed to provide more reliable estimations, when the information is available. Based on this information, a 3D controller is implemented for the UAV, that can effectively respond to the movement of the target while keeping it in line-of-sight at a pre-determined distance.

Third, extensive simulation testing is conducted to evaluate the performance and reliability of the integrated system and the controller under various conditions and scenarios.

Fourth, a UAV platform is developed by integrating and assembling every necessary component for a fully functional experimental setup capable of performing real-world tests. With this, real-world tests are conducted to validate the performance of the developed system through real-world testing using the developed platform. Finally, the performance of the developed system is analysed, by comparing both the YOLOv8+BoT-SORT and the YOLOv8+ByteTrack setups in terms of detection accuracy, tracking consistency, computational load, control precision, and overall system robustness.

1.3 Contributions

The objectives of this thesis aim to push the boundaries of vision-based target tracking and following using UAVs, particularly in dynamic and uncooperative environments, by leveraging multi-target information. By proposing new redetection methods to tackle the problems associated with integrating MOT algorithms in target following, this work seeks to bridge the gaps in current state-of-the-art solutions and allow new applications in the field.

The main contributions of this thesis are:

1. **Development of a Visual Detection and Tracking System**: Implementation of a state-of-the-art visual detection and tracking system utilizing the YOLOv8 detection algorithm in combination with two MOTs algorithms: BoT-SORT and and its predecessor ByteTrack. This system is designed for target following scenarios, providing robust multi-target information.

2. **Innovative Redetection Algorithm**: Creation of a novel redetection algorithm that leverages

multi-target information to effectively handle partial and full occlusions of the target in dynamic environments. This enhances the system's ability to maintain continuous tracking despite visual obstructions.

3. **3D Flight Control Algorithm Design**: Design and implementation of a 3D flight control algorithm that uses both RGB and depth information to accurately follow a target. This approach improves the precision of distance estimation and target positioning.

4. **Comprehensive Simulation Testing**: Extensive testing of the integrated system and control algorithms within the MRS-UAV System [11] simulation environment. These tests evaluate the system's performance, reliability, and robustness under various conditions and scenarios.

5. **Real-World Validation and Data Collection**: Development of a fully functional UAV platform to perform real-world tests, validating the system's performance in near-limit conditions. These tests provide valuable data to support the findings and demonstrate the system's capabilities and limitations in practical applications.

These contributions aim to enhance the effectiveness and reliability of vision-based target tracking and following systems using UAVs in dynamic scenarios, paving the way for for new and exciting applications with the use of multi-target information.

This work was partially published in the 2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC2024) [12]. The paper presents the proposed method and simulation results using the YOLOv8+BoT-SORT setup with slight modifications.

- D. Ferreira and M. Basiri. Leveraging multi-object tracking in vision-based target following for unmanned aerial vehicles. In 2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pages 88–93, 2024. doi: 10.1109/ICARSC61747.2024.10535936.

1.4 Thesis Outline

The structure of this thesis is as follows:

Chapter 1 - Introduction: This is the current chapter which provides the context and motivation behind the proposed topic, glances over the current dilemma in the literature, exposing the research proposition. It also outlines the proposed solution and major contributions of this thesis in the context of vision-based mobile target tracking and following by UAVs.

Chapter 2 - Literature Review: This chapter explores the state-of-the-art regarding this field of research. It begins by analysing the current object detection methodologies and explaining the YOLO family. Next, it reviews the literature on target tracking and following approaches using SOT and explains the current MOT algorithms. The chapter also explores distance estimation and control methods used in similar systems. Finally, it concludes with an overall analysis of the available approaches to the problem,

highlighting their advantages and limitations. The analysis culminates in the identification of a noticeable research gap in the current literature that this thesis aims to investigate.

Chapter 3 - Theoretical Background: This chapter explores some mathematical concepts related to this thesis that will be useful for the explanation of the proposed approach.

Chapter 4 - Proposed Approach: This chapter describes the full system, starting with the architecture, the implementation of the Detection, Tracking and Following algorithms followed by an overview on the Robot Operating System (ROS) implementation.

Chapter 5 - Results: This chapter first dives into the simulation experiments, explaining the setup and experiments. It presents the performance of the system and compares the differences obtained for both trackers. Next, it transitions into the real-world results starting with the test environments and the developed UAV platform. Then, it shows an analysis on the redetection capabilities and robustness of the system, followed by some long-term following experiments in an open-field. Finally, it ends with some additional tests in limit conditions.

Chapter 6 - Conclusions: This chapter presents the overall achievements of the developed system and discusses the limitations and possible improvements for future works.

Chapter 2

Literature Review

The research in the field of vision-based mobile target tracking and following from a UAV can be broadly categorized into three distinct steps:

1. Object Detection in an image;
2. Mobile Target Tracking in an image;
3. Vision-based Mobile Target following.

Each of these steps plays a vital role in the overall process and has been the subject of extensive research. This literature review aims to dive into these topics, exploring the various methodologies and techniques that have been proposed, their advantages and limitations, and the current state-of-the-art. The goal is to provide a comprehensive understanding of the existing body of knowledge in this field and identify potential areas for further research and development.

2.1 Object Detection in an Image

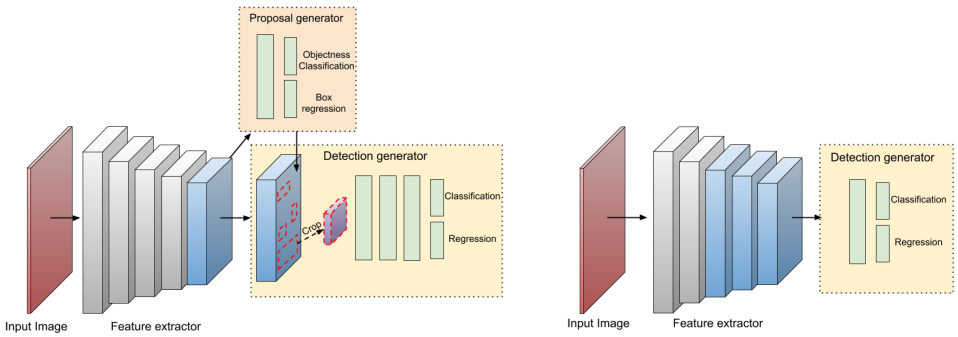
Identifying an object in an image can be defined as the act of surrounding the object with a bounding box and classifying which type of object it is. Traditional object detection algorithms are mainly composed of three main steps: informative region selection, feature extraction, and classification [13]. Informative region selection involves identifying potential areas in the image that may contain the object of interest, thereby reducing computational complexity. The second step, feature extraction, involves analysing these regions and extracting a set of distinctive attributes or properties, such as edges or textures, that best represent the object. For this second step, different sets of techniques can be used, such as: Histogram of Oriented Gradients (HOG) [14], which calculates and encodes local shape information from regions within an image; Scale-Invariant Feature Transform (SIFT) [15], which detects and describes local features in images, making them invariant to image scaling, translation, and rotation; and Haar-like feature extractors [16], which uses a filter to perform feature extraction from an image, summing the pixel intensities in each region and calculating the differences. In some deep-learning approaches a Convolutional Neural Network (CNN) is used in the feature extraction step. The final step, Classification,

involves using these features to categorize the object within a given set, with the help of a classifier which could be a traditional machine learning model or a deep learning model.

2.1.1 Deep Learning-based Detection

Deep Neural Networks (DNN) models can be referred to as neural networks with deep structures and the first implementations can date back as far as the 1940s [17], inspired by human brain nervous connections. The first goal was to create an algorithm capable of classifying information according to useful common characters and learn features directly from the data without the need for manual feature extraction. During the 1980s and 1990s, it kept increasing its' popularity, losing some relevance in the 2000s in favor of other machine learning methods [18]. After 2006, new technologies in parallel computing, such as GPU clusters, began to emerge. This, along with the availability of large scale annotated training data like ImageNet [19], and improved design of network structures and training strategies, led to DNN gaining new popularity and relevance [18].

Deep learning-based object detection algorithms can be divided into two categories: single-stage or two-stage detectors as shown in Figure 2.1. In short, one-stage detectors have high inference speeds and two-stage detectors have high localization and recognition accuracy [20]. Two-stage detectors first generate regions of interests in the image and send the region proposals down the pipeline to obtain the object bounding-boxes and classification [21]. Post-processing is then applied to fine-tune the bounding box and remove any duplicate detections. Although this two-stage architecture has the advantage of higher accuracy, it is slower and not entirely fit for real-time deployment onto constrained edge devices [9]. To allow faster speeds, single-stage detectors treat object detection as a simple regression problem by taking an input image and learning the class probabilities and the bounding box coordinates in one single step [21] without using pre-generated region proposals. Some examples of single-stage detectors are Single Shot Detector (SSD) [22], Deconvolutional Single Shot Detector (D-SSD) [23], RetinaNet [24] and YOLO [25].



(a) Two-stage Faster R-CNN (b) One-stage RetinaNet

Figure 2.1: Deep learning object detection meta-architectures from [26]. Comparison between one-stage and two-stage object detectors

2.1.2 YOLO

YOLO was first introduced in 2016 as the first detector to frame object detection as a simple regression problem, allowing for real-time object detection with a single pass through the network. This made it faster and more efficient compared to many of its predecessors. Currently YOLO is a family of architectures, very popular due to its high accuracy, lightweight, and capabilities of edge deployment with the most recent introduced in 2022 as YOLOv8 [9].

Since YOLO fits the category of single-stage object detector, it performs the three steps of object detection in one evaluation: region selection, feature extraction, and classification. This is done using a CNN to analyze the entire image in one pass. The CNN divides the input image into a SxS grid, with each grid cell responsible for predicting the objects in that cell returning the bounding boxes along with the associated confidence scores reflecting how confident the model is that the box contains an object. In addition, the CNN also returns the class probabilities.

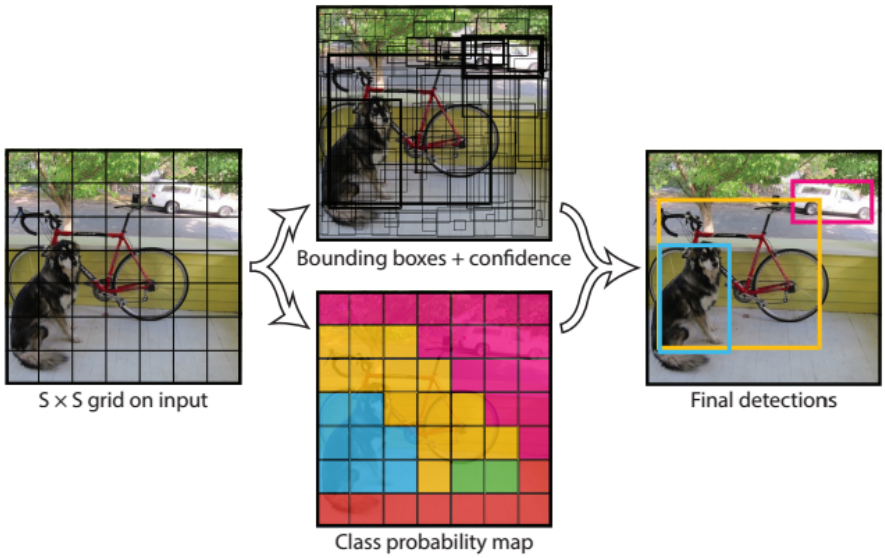


Figure 2.2: YOLOv1 Model. The input image is divided into an SxS grid and for each cell there are B predicted bounding boxes with correspondent scores, and C class probabilities. Taken from [25].

The network consists of multiple layers that extract features from the image. These can be extracted by identifying various details from simple ones like edges and corners in the initial layers, to more complex ones like shapes and patterns in the deeper layers. The output of the CNN acts as a feature extractor, and the extracted features are then used for predicting the probabilities and coordinates of objects. Since the YOLO detection pipeline is composed of a single network, it can be optimized on detection performance, making it exceptionally fast and efficient for real-time object detection.

2.1.3 YOLOv5

YOLOv5 [27] was introduced by Ultralytics in 2020 and has become very popular in computer vision applications that need real-time capabilities [28–30], showing improvements over the previous YOLOv4 model. In addition, YOLOv5 was the first of the YOLO family to be developed in PyTorch instead of the

Darknet framework of the other variants. Pytorch [31] provided a friendlier approach with emphasis on usability on top of speed, which made it popular within the computer vision community.

Similar to the previous YOLOv4 version, the YOLOv5 architecture is composed of 3 components: the backbone, the neck and the head shown in Figure 2.3.

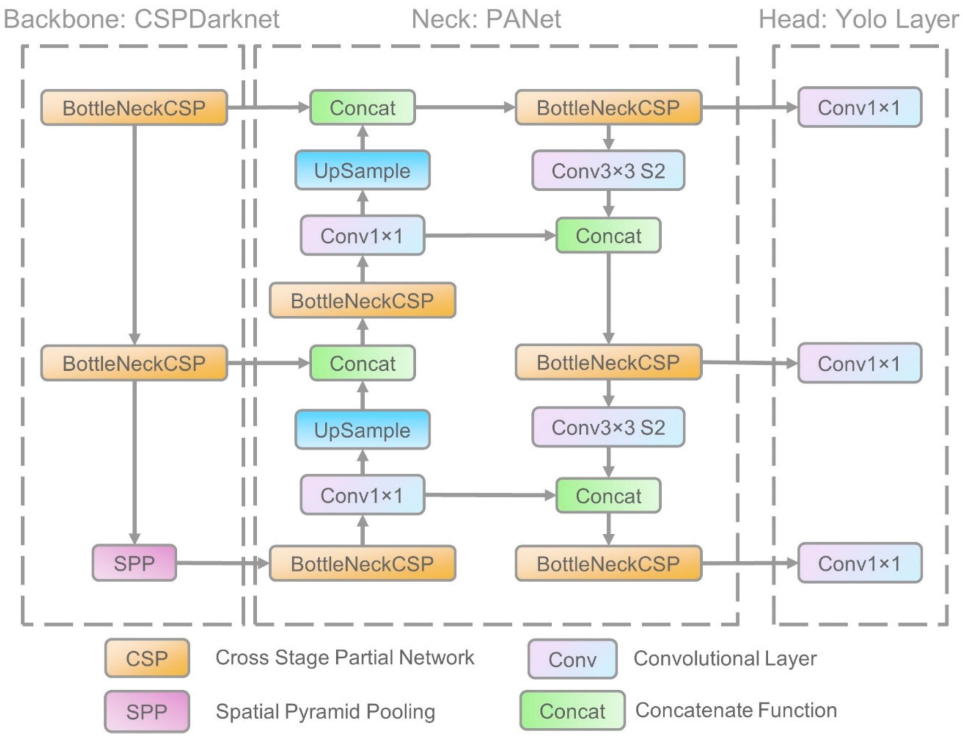


Figure 2.3: The network architecture of YOLOv5. It consists of three parts: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head: YOLO Layer. The data are first input to CSPDarknet for feature extraction, and then fed to PANet for feature fusion. Finally, YOLO Layer outputs detection results (class, score, location, size). Taken from [30].

The backbone serves as the main body of the network and extracts rich feature representations from the input image. The backbone is designed using the CSP-Darknet53 structure which essentially combines the convolutional network Darknet53 with the Cross Stage Partial (CSP) network strategy [32]. Using this method, the backbone is able to decrease the spatial resolution of the image while enhancing important features, making the model more compact which in turn increases the processing speed for the next steps.

The neck connects the backbone and the head and applies a Path Aggregation Network (PANet) [33] to extract feature pyramids. Feature pyramids are a crucial step in computer vision than enable the detection of objects at various spatial resolutions within the same image. By combining features from different scales, YOLOv5 can handle objects of different sizes effectively. The neck bridges the gap between the low-level features extracted by the backbone and the high-level features needed for accurate predictions.

The head is responsible for the final stage operations in YOLOv5. It uses the features from the neck and scans the image for possible objects. It applies anchor boxes on feature maps and generates the final output, which includes the predicted class labels for detected objects, the confidence scores

indicating the presence of an object and finally the bounding boxes, which specify the location of the detected object.

2.1.4 YOLOv8

YOLOv8 [34] is one of the most recent additions to the YOLO family by Ultralytics on 10 January 2023, the original creators of YOLOv5. The YOLOv8 architecture shown in Figure 2.4 is very similar to YOLOv5 which a couple of major improvements leading to better speed and accuracy.

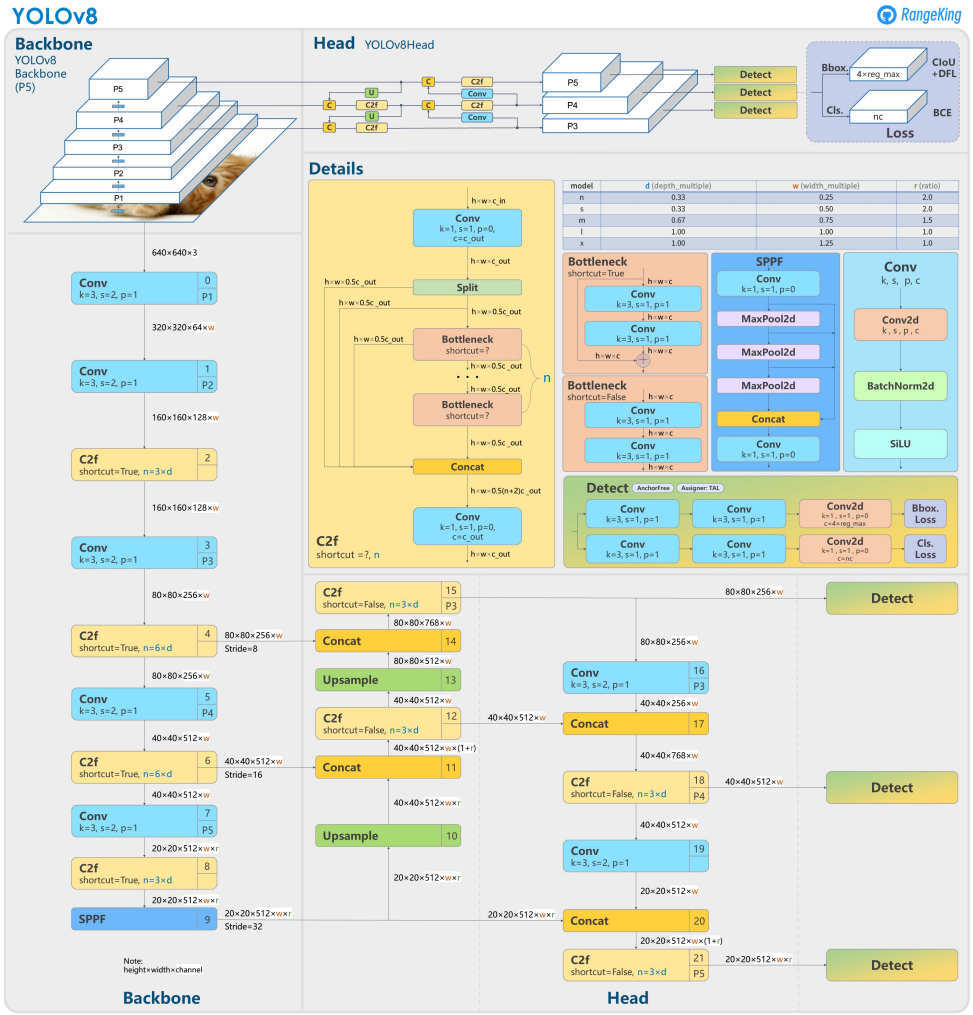


Figure 2.4: The network architecture of YOLOv8 by [35].

By exploring the YOLOv8 architecture, some other notable changes can be pointed out.

As mentioned in the YOLOv5 explanation, the backbone serves the purpose of extracting features from the image. In YOLOv8, the CSPDarknet53 is replaced by the CSP Bottleneck with 2 Convolutions - Fusion (C2f) module. The bottleneck improves efficiency by dividing the input into two parts, performing a convolution step in one of them and merging both paths together again. After four C2f steps, the algorithm utilizes a Spatial Pyramid Pooling - Fast (SPPF), an upgrade from the Spatial Pyramid Pooling (SPP) in YOLOv5. This module together with the convolution layers processes the features at varying scales while the Upsample layers increase the resolution of feature maps. The C2f module combines

the high-level features with contextual information to improve detection accuracy.

Finally, the high-dimensional features are mapped into bounding boxes and classes by a set of convolutional and linear layers in the Detection module. Since YOLOv8 is an anchor-free model, it directly predicts the center of the object rather than the distance to a predefined anchor box. This speeds up the process by lowering the amount of box predictions and consequently shortening the time taken in post-processing [36]. Also, the YOLOv8 head performs class assignment and bounding box assignment separately, thereby enhancing the processing speed of the module.

2.2 Mobile Target Tracking in an image

Tracking in computer vision can be succinctly defined as the process of identifying and monitoring the position of a detected object in an image sequence over time. This is achieved by assigning a unique identifier to each detected object and subsequently tracking the evolution of its position across successive frames. The tracking of objects can be categorized within two different levels, SOT and MOT.

2.2.1 Single-Object Tracking approaches

SOT focuses on tracking a specific object over time and the process typically involves three main steps:

1. **Initialization:** The tracker is initialized in the first frame by manually selecting the object to track or using an automatic object detection algorithm (e.g. YOLO [37]);
2. **Propagation:** The tracker estimates the position of the object in the subsequent frames by analyzing its appearance and motion information in the previous frames;
3. **Adaptation:** The tracker adjusts to variations in the object's appearance, scale, or pose, as well as changes in environmental conditions, to ensure accurate tracking over time.

On SOT, some works chose to address this problem using classical computer vision like Image-Based Visual Servoing (IBVS) [38–40]. In IBVS approaches, the detection and tracking steps are performed by extracting features from the images, which can be points, lines, corners or more complex structures. The features are then used to compute the error between the current and desired image features. The error is then used as feedback for the control algorithms.

Detection and tracking in IBVS approaches has some noticeable limitations, requiring clear unaltered visual features in the target for detection. This proves impractical for some targets such as people, where the visual features are not so clear. In the works [38–40], the object to follow was a rectangular plane ground robot using the 4 corners as distinct features. IBVS also has difficulties with occlusion since it needs the features to be always identifiable. According to Zheng et al. [39], loss of the target visual features will result in a fatal failure of the tracking task.

To address these concerns and enable less limited applications other authors preferred the Kernelized Correlation Filter (KCF) method [10, 37, 41–43] for vision-based object tracking from UAVs.

The KCF method uses implicit properties of tracked images (circulant matrices) for training and tracking in real-time. Unlike deep learning, which is data intensive, KCF uses these dynamic properties of the scene and movements of image patches to form an efficient representation based on the circulant structure for further processing, using properties such as diagonalizing in the Fourier domain. Correlation-based trackers such as KCF propose one-shot learning and show better performances without GPU acceleration, which makes them very appealing for embedded systems with computational limitations [44].

However the KCF method is very susceptible to partial target occlusions and can only track the target in the image after the selection of the target in the first frame, which is usual done manually [10, 41, 43]. To tackle the problem of recovery after occlusion, Cheng et al. [41] developed an algorithm that analyses the motion between frames to detect movement indicative of the target. This can be susceptible to noise and dynamic environments. Wei [43] tackles this problem by using the Kalman Filter in conjunction with the KCF. This is done by defining a confidence level for the measurements of the KCF. When the confidence level drops, the estimations of the KF are taken into consideration until the target is retrieved. Luo et al. [37] proposes an interesting approach, using the KCF and Kalman Filter approach similar to [43] and deep learning methods (YOLOv3) to initialize the tracker while improving the capabilities of recovery after occlusion. The author uses the Kalman Filter to keep track of the targets movement which is useful in case of occlusion, however after many consecutive failure frames, the UAV stops following the estimations from the Kalman Filter and stays in hover utilizing the YOLOv3 until a new detection is made and restarting the KCF in that case. This, however, is also very susceptible to dynamic scenarios where YOLO can detect more than one pedestrian, in which case it will not be clear which one is the previous target and may result in losing the target.

With technological progresses in GPU boards for UAVs such as the NVIDIA Jetson and improvements in one-stage deep neural detectors such as the newer versions of YOLO, the computational limitations that called for the use of fast methods such as the KCF are slowly being lifted. With that in mind, there can be a transition to more robust deep learning-based approaches for object tracking which may integrate MOT algorithms to compensate for complex and dynamic backgrounds, leveraging multi-target information.

2.2.2 Multi-Object Tracking approaches

MOT is a technique that aims to analyze images to locate and track multiple moving objects, without any prior knowledge about the appearance and the number of target entities. A common approach in many MOT methods is the use of pre-trained object detectors to identify target objects within individual frames. These detections are then linked over time by trackers, by attributing unique identifiers for each object. This method is known as track-by-detection, and several algorithms such as SORT [45], Deep SORT [46], ByteTrack [47], and BoT-SORT [7] have been developed based on this concept. Typically, these algorithms involve the following steps [45]:

1. **Detection:** A pre-trained object detector is used to identify objects of interest in an image.

2. **Estimation model:** The tracker predicts the position of the objects in the current frame based on previous detections.

3. **Data association:** The tracker calculates the similarity between the predictions and the detections, using factors like position, size, and potentially more advanced features like appearance descriptors.

4. **Creation and deletion of identities:** The calculated similarity is used to match identities to current objects being tracked, delete identities when objects are not visible for some frames or create new identities for unmatched objects.

Each of these steps plays a crucial role in ensuring the effectiveness of the tracking process.

DeepSORT is an improvement on the SORT algorithm, and both are commonly used for MOT. SORT [45] manages to achieve great accuracy using a simple effective combination of the Kalman filter and the Hungarian matching algorithm [48], proving to be suitable for real-time applications. DeepSORT [46], improves on the previous algorithm by integrating appearance information by using pre-trained association metrics which represent major improvements in recovery from occlusion.

ByteTrack [47] introduces a novel approach to multi-object tracking by retaining low-score detections in the tracking process. This innovative method allows it to preemptively detect partially occluded objects and yield superior tracking predictions. Unlike traditional methods that set a detection score threshold and discard all information below this level, ByteTrack leverages the similarity between low-score detections and existing tracklets (short tracks representing the path across a limited number of frames) to enhance the algorithm's performance. Initially, ByteTrack matches high-score detections with existing tracklets. However, due to factors such as occlusions, motion blur, or size differences, some tracklets remain unmatched. At this point, ByteTrack considers low-score detections to recover some of these unmatched tracklets. All matched tracklets are retained, while unmatched low-score detections, considered as background noise, are discarded. During the first high-score detection matching, an appearance similarity model can be employed to match tracklets. However, for the second association involving low-score detections, which typically represent occluded objects, reliance on appearance models is not feasible. Therefore, ByteTrack resorts to an Intersection over Union (IoU) approach, which does not depend on detected features. Following the execution of both high-score and low-score associations, unmatched tracklets are stored in a temporary list for a specific number of frames. If they remain unmatched after this period, they are subsequently deleted.

Although ByteTrack is able to achieve better tracking capabilities more robust to occlusions, it has still difficulties handling camera motion. Hence, due to the complex nature of pedestrian tracking and frequent occlusion, the accuracy of SORT, DeepSORT and ByteTrack are still compromised. BoT-SORT [7] was developed with the goal of improving these algorithms, making them more robust to camera movement and occlusions, and is tested directly in pedestrian datasets. BoT-SORT identifies two primary shortcomings in MOT algorithms and integrates them into ByteTrack [47] with the respective solutions:

1. **Camera Motion Compensation**: This is incorporated to prevent inaccurate bounding box positions that can occur due to camera movement. By compensating for the camera motion, the

algorithm can maintain accurate tracking in dynamic scenarios of moving UAVs.

2. **Improved Kalman Filter State Vector**: Instead of merely using the aspect ratio, BoT-SORT improves the Kalman filter state vector to calculate the width and height of the bounding boxes. This leads to a more precise localization of the object, enhancing the overall tracking accuracy.

With these changes, BoT-SORT ranks above previous algorithms including SORT, DeepSORT and ByteTrack in IDF1, MOTA, and HOTA performances as show in figure Fig. 2.5. IDF1, MOTA, and HOTA are metrics used to evaluate the performance of multi-object tracking algorithms. IDF1 [49] measures the consistency of trajectory ID with the ground truth ID. MOTA [50] combines three error sources: false positives, missed targets, and mismatch errors, with 100% indicating perfect performance. HOTA [51] measures the alignment between the predicted and ground truth trajectories as sets of object detections over time, focusing on how well identities are maintained.

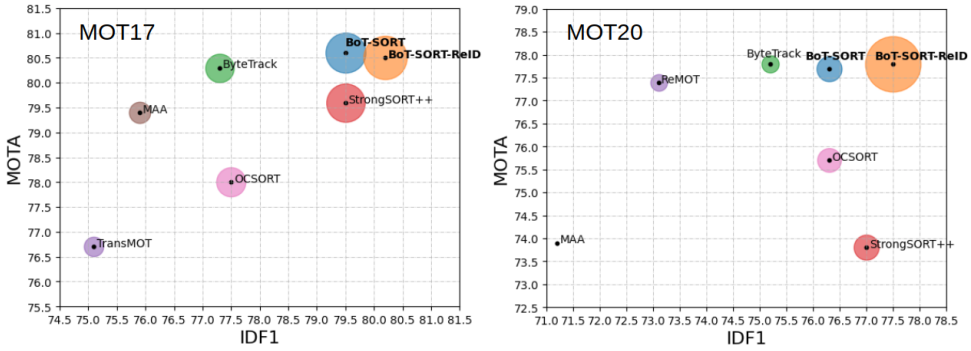


Figure 2.5: IDF1-MOTA-HOTA comparisons of state-of-the-art trackers with BoT-SORT and BoT-SORT-ReID on the MOT17 and MOT20 test set kindly borrowed from [7]. The x-axis is IDF1, the y-axis is MOTA, and the radius of the circle is HOTA.

Nehru et al. [52] conducts a comparison between BoT-SORT and its predecessor ByteTrack. The results indicate that BoT-SORT is capable of improving upon existing strategies while maintaining real-time capabilities. However, due to its simplicity, ByteTrack outperforms BoT-SORT in terms of computational cost.

Regarding UAV-based MOT approaches, Liu and Zhang [5] proposes a detection and tracking strategy for UAVs using YOLOv4 and DeepSORT for vehicles in urban scenes. The combination of YOLO with a track-by-detection method such as DeepSORT shows great accuracy and robustness in multi-object detection and tracking.

Li et al. [53] demonstrates the accuracy of a tracking-by-detection approach using both YOLO (in this case YOLOv7) and the state-of-the-art BoT-SORT algorithm. Results show an effective solution for target occlusion and identity switching in pedestrian target tracking, even under poor illumination conditions and complex scenes, which are some of the most common difficulties in this task. In comparison to similar methods such as YOLOv7-DeepSORT, YOLOv7-StrongSORT and YOLOv7-ByteTrack, YOLOv7-BoTSORT shows the best results between them for non-rigid target tracking of pedestrians.

Yan et al. [54] applies a similar system that utilizes YOLOv8 and BoT-SORT within a Synthetic Aperture Radar imaging framework. The experimental findings suggest that the proposed method exhibits

high precision in both detection and tracking. Furthermore, it is capable of performing tracking operations in real-time.

2.3 Vision-based Mobile Target following

Once the location of the target within the image has been identified using the aforementioned methods, the subsequent step involves leveraging this information to precisely ascertain its position relative to the UAV, enabling effective following.

2.3.1 Mobile Target in-world position estimation

A common strategy for target position estimation involves using the discrepancy between the pixel position of the target and the center of the image frame as an estimate of camera deviation. This method is used to direct the camera towards the target, achieving line-of-sight [10, 37, 41, 42]. Both Liu et al. [10] and Cheng et al. [41] employ this deviation to steer the gimbal towards the target. In contrast, Feng et al. [42] and Luo et al. [37] don't utilize a gimbal system, therefore use this deviation as feedback for lateral and vertical movements to keep the UAV focused on the target.

After the target is in line-of-sight, the next step is to maintain the relative distance for target following. Cheng et al. [41] estimates distance based on the standard pinhole imaging model, which calculates the distance to the target using the height of the UAV, the target's pixel coordinates and the intrinsic parameters of the camera. This is then combined with an Extended Kalman Filter to account for observation noise. On the other hand, Liu et al. [10] and Luo et al. [37] use the proportion of the size of the target in relation to the image as a reference for the following speed.

Alternatively, Feng et al. [42] employs a method of converting the image values to the body frame to estimate the relative position of the target. This process requires depth information to determine the target's height for future iterations, which is not available in their configuration. To acquire this information, the UAV initiates a probing step, capturing two consecutive frames from different perspectives of the target by simply moving sideways. This is carried out under the assumptions that there is no rotation of the camera, no forward/backwards movement and no vertical movement from the UAV, and that the target remains stationary. The author then uses triangulation with these two frames to determine the target's real height. This real height value is subsequently used to estimate the distance to the target by comparing it with the image height. The position of the target is then provided to the flight control software as an input.

2.3.2 Control algorithms for Mobile Target following

Using the estimation mechanisms enumerated previously, Feng et al. [42] uses a simple approach of giving the 3D position calculated directly to the flight controller. Luo et al. [37] employs a 3D proportional control algorithm by using the horizontal and vertical pixel deviations for the lateral and vertical speeds of the UAV. Then, they use the scale change of the target for the forward/backward speed of the UAV. To

achieve smooth movement and prevent rapid changes, the author applies a maximum velocity change threshold, usually referred to as slew rate controller. Liu et al. [10] follows a Proportional Navigation (PN) strategy, aiming at keeping the target in line-of-sight at all times and using the scale change as reference for acceleration changes. The references of desired position and yaw angle are then given to a cascade Proportional Integrative Derivative (PID) controller to control the position and attitude of the UAV. Cheng et al. [41] present a Switchable Tracking Strategy based on the estimated distance to the target, varying between the observing and following modes. In observing mode, the UAV only changes the yaw angle using a PID controller, while in following mode, the UAV will change its horizontal position assuming a 2D tracking problem based on the Lyapunov theory.

2.4 Overall Literature Analysis

As shown in 2.2.1, recent advancements in SOT have addressed some limitations related to occlusions by incorporating new methods of object position monitoring, such as Kalman filters, and leveraging these estimates to enhance existing algorithms. However, they still show some limitations in dynamic scenarios where more than one moving target is in line-of-sight. On top of that, these algorithms, with their single initial detection step, do not fully utilize the capabilities of new deep learning-based detection algorithms such as YOLO. In contrast, MOTs employ a tracking-by-detection approach, which, while more computationally intensive, makes better use of state-of-the-art detection algorithms. Furthermore, MOT methods are more adept at managing complex and dynamic scenarios involving multiple moving entities.

Despite the remarkable utility of MOTs in dynamically monitoring various targets such as vehicles and pedestrians using UAVs, there is a lack of research exploring the use of these methods for target following. Hence, a visible research gap emerges in the application of a combined system that leverages a deep learning method like YOLOv8 with an MOT algorithm like BoT-SORT. Such a combination could enable more intricate operations with UAVs in mobile target tracking and following. The use of an MOT allows the system to monitor not just the target it is following, but also the surrounding objects. This capability can be particularly beneficial for following pedestrians in densely populated environments and provides a more robust way to handle occlusions.

This research introduces a combination of YOLOv8 with the trackers BoT-SORT or ByteTrack to detect and track moving targets. The detections and identities obtained are then integrated into the control system to enable accurate following of a selected target, demonstrating the potential of this combined approach in advancing UAV operations. The multi-target information is used into a comprehensive re-detection scheme that is able to retrieve target tracking and following after partial and full occlusions in dynamic scenarios.

In the realm of mobile target following, this work draws inspiration from Luo et al. [37], Feng et al. [42], employing the deviation between the target's pixel location and the central position of the image frame as feedback for lateral and vertical movements. The objective is maintaining the focus of the UAV on the target, utilizing a PN strategy [10]. The distance to the target is estimated using two different

methods: by comparing the scale of the target in relation to the image, and by using depth sensors.

The UAV forward speed is calculated using a Proportional Integrative (PI) controller using the estimated distance to the target as reference. A Switchable Tracking Strategy [41] is implemented, switching between two methods of distance estimation, since the depth sensors can only provide accurate readings when the UAV is within a close proximity to the target.

Chapter 3

Theoretical Background

In this chapter, some important theoretical background for this thesis will be explored, namely the pinhole model, camera parameters and the Kalman Filter.

3.1 Pinhole Camera Model

The pinhole camera model is an approximation model used to mathematically describe the relation between the three-dimensional coordinates of a point with the two-dimensional projection coordinates in the image frame. This relation takes into consideration that the camera can be modeled as an ideal pinhole camera where the camera aperture is no more than just a point and no lenses are used to focus light. Naturally this approximation doesn't represent the reality in its entirety however, despite its simplicity, it is still an acceptable approximation for most cases.

The pinhole model equivalence can be visualized in Figure 3.1. Considering, in this scene, a reference frame centered around the camera center and an image plane at a distance f in the Z direction, its given that the projection of a point X with coordinates (X,Y,Z) in the image plane is the point x with coordinates (x,y,z) of the image plane. Since the distance from the pinhole to the image plane is fixed, the coordinates for point x can be defined as (x, y, f).

By triangle similarity, it gives that the point X (X,Y,Z) is projected to the point x such as:

$$\begin{cases} x = f\frac{X}{Z} \\ y = f\frac{Y}{Z} \end{cases}, \tag{3.1}$$

effectively mapping the $\mathbb{R}^3$ space to an $\mathbb{R}^2$ space by ignoring the final image coordinate.

3.1.1 Camera intrinsic parameters

Taking into consideration the pinhole camera model, it is possible to derive the a set of parameters used to describe the camera and perform translations between 3D and 2D coordinates. Firstly the homogeneous coordinates for the points mentioned above are taken into consideration where a normalized

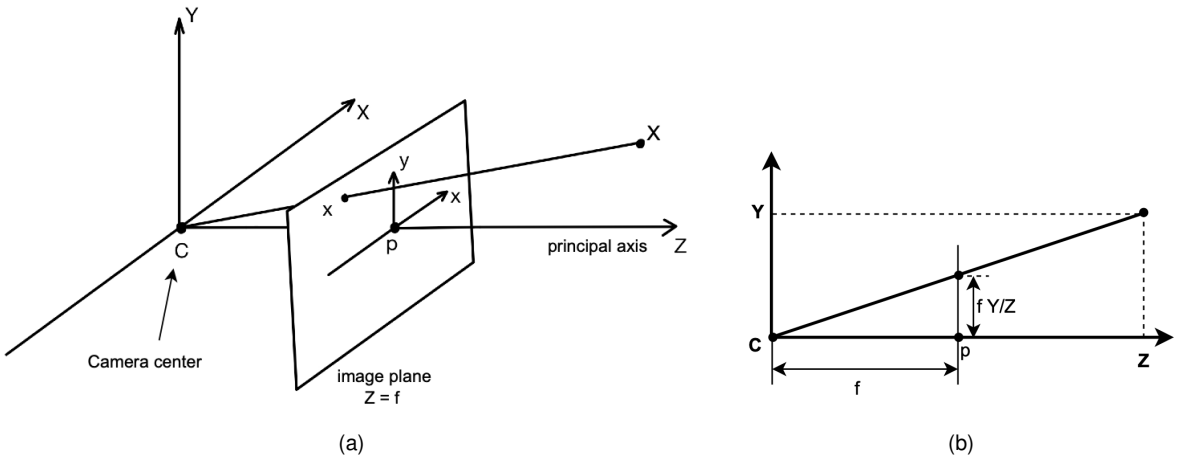


Figure 3.1: Representation of the pinhole model. C is the camera center in the coordinate origin. p is the principal point in the image plane, placed in front of the camera center.

plane is considered at a unit distance ($f = 1$) between the pinhole and the image plane to normalize the coordinates.

With this homogeneous coordinates representation of the system, the relation between the world and the image plane points may be written in terms of matrix multiplication:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}. \tag{3.2}$$

Considering that many cameras have rectangular pixels, the focal distance f is changed for each coordinate into the focal lengths f_x and f_y. In addition, the camera center may not be precisely aligned with the principal point. In order to correct this deviation, the principal point coordinates (ppx, ppy) must be considered. These changes applied to Equation (3.2) give Equation (3.3).

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} f_x X + Zppx \\ f_x Y + Zppy \\ Z \end{pmatrix} = \begin{bmatrix} f_X & 0 & ppx \\ 0 & f_y & ppy \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{3.3}$$

with:

$$K = \begin{bmatrix} f_X & 0 & ppx \\ 0 & f_y & ppy \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

The matrix K shown in Equation (3.4) is called the camera calibration matrix and contains the intrinsic parameters of the camera.

3.2 Kalman Filter

A Kalman Filter [55] is a mathematical model that can be used to estimate the state of a linear dynamic system from a series of noisy measurements. The problem is best described by the author as the ability to measure the value of a random variable $x_1(k)$ given the observations of its noisy counterpart $y(k) = x_1(k) + w(k)$ at all time-steps such as $y(k_0), ..., y(k)$ is known, such as $y(k)$ can also be defined as a random variable and $w(k)$ is a random Gaussian noise. Hence, Kalman [55] defines estimation as the inference of the value of $x_1(t)$ at any time-step k_1 from the observation $y(k_0), ..., y(k)$, for any k. In this sense, the Kalman Filter can perform interpolation if $k_1 < k$, filtering for $k_1 = k$ or predictions for $k_1 > k$.

The problem of target tracking in the image is commonly modeled as linear dynamic system [56] which can be described as:

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \qquad k > 0 \tag{3.5}$$

$$z_k = H x_k + v_k \tag{3.6}$$

$$x(0) = x_0 \tag{3.7}$$

where x_k, u_k, z_k, w_k and v_k represent the state, control, observations and measurements noise, respectively, and A_k, B_k and H represent the system transition, input distribution and observation matrices, respectively.

w_k and v_k are Gaussian, white noises processes that are random with zero mean,

$$E\{w_k\} = 0, \tag{3.8}$$

$$E\{v_k\} = 0 \tag{3.9}$$

have no time-correlation,

$$E\{w_i w_j^T\} = E\{v_i v_j^T\} = 0, \qquad \text{if } i \neq j \tag{3.10}$$

and have covariance matrices Q(k) and R(k), respectively, both positive definite defined by:

$$E\{w_k w_k^T\} = Q_k, \tag{3.11}$$

$$E\{v_k v_k^T\} = R_k. \tag{3.12}$$

Given that all the variables are random and Gaussian, the Kalman Filter takes root in the principles of Bayesian inference which provides a systematic method for updating the probability estimate for a hypothesis as additional evidence is acquired. This probabilistic approach is essential for understanding

how the Kalman Filter integrates noisy measurements to produce optimal state estimates. The goal of the Kalman Filter is to derive the current state based on the previous estimates. The *a priori* state is defined as the estimate at time t $\hat{x}_k^- \in \Re^n$ which is the state based on previous observations and $\hat{x}_k \in \Re^n$ defines the *a posteriori* state estimate given the observation z_k. The *a posteriori* estimate can be given as a linear combination between the *a priori* estimate and a measurement prediction $H\hat{x}_k^-$ as shown in Equation (3.13):

$$\hat{x}_k = \hat{x}_k^- - K(z_k - H\hat{x}_k^-), \tag{3.13}$$

where $(z_k - H\hat{x}_k^-)$ is the residual which represents the difference between the measured and the expected values. The value K is usually referred to as the Kalman Gain and represents the matrix that minimizes the *a posteriori* error covariance $P_k = E([x_k - \hat{x}_k][x_k - \hat{x}_k]^T)$ such as:

$$K_k = P_k^- H^T (H P_k^- H^T + R_k)^{-1}. \tag{3.14}$$

The recursive process of the Kalman Filter follows two main steps: prediction and update. In the prediction step the filter utilizes the proposed system dynamics to estimate the next state of the system based on the current state and the associated error covariance for this prediction considering the uncertainty in the system's dynamics and any process noise.

In the update step, the measured values are used to improve the state estimate. Based on the confidence levels given to the measurement and the prediction (measurement noise and sensor accuracy), the filter combines the information using weighted averages. After the update step, the filter again gives an updated version of the estimated state and respective error covariance.

The iterative steps from the Kalman Filter can be given by:

Prediction Step:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3.15}$$

$$P_k^- = AP_{k-1}A^T + Q_k. \tag{3.16}$$

Update Step:

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R_k)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ P_k &= (I - K_k H) P_k^-. \end{aligned}$$

Chapter 4

Proposed Approach

This chapter describes the full system, starting with the architecture, the implementation of the detection, tracking and following algorithms followed by an overview on the ROS implementation.

In order to properly describe the proposed system, it is important to remember some important definitions:

- **Detection**: The process of identifying and localizing objects of interest within an image. YOLOv8 is used for this step, which provides bounding box detections.
- **Bounding Box**: A rectangular region that encloses an object detected within an image. In YOLOv8, the bounding box contains the information of the size (width, height) of the rectangular region and the center of the bounding box.
- **Tracking**: Following objects of interest over consecutive frames in an image sequence. The BoT-SORT or ByteTrack algorithms are used to associate detected objects across frames, thereby maintaining consistent identities for objects as they move.
- **Target**: The specific object of interest that is being tracked and followed.

4.1 System Configuration

The system configuration is shown in Figure 4.1. The overall system can be split in to four modules:

1. Visual Detection and Tracking Module;
2. Distance Estimation;
3. Following Flight Controller;
4. System Mode Switcher.

The ”Vision Detection and MOT Module” encompasses the Vision Detection and MOT Algorithm where all objects of interest in the field-of-view including the target to follow are detected and tracked

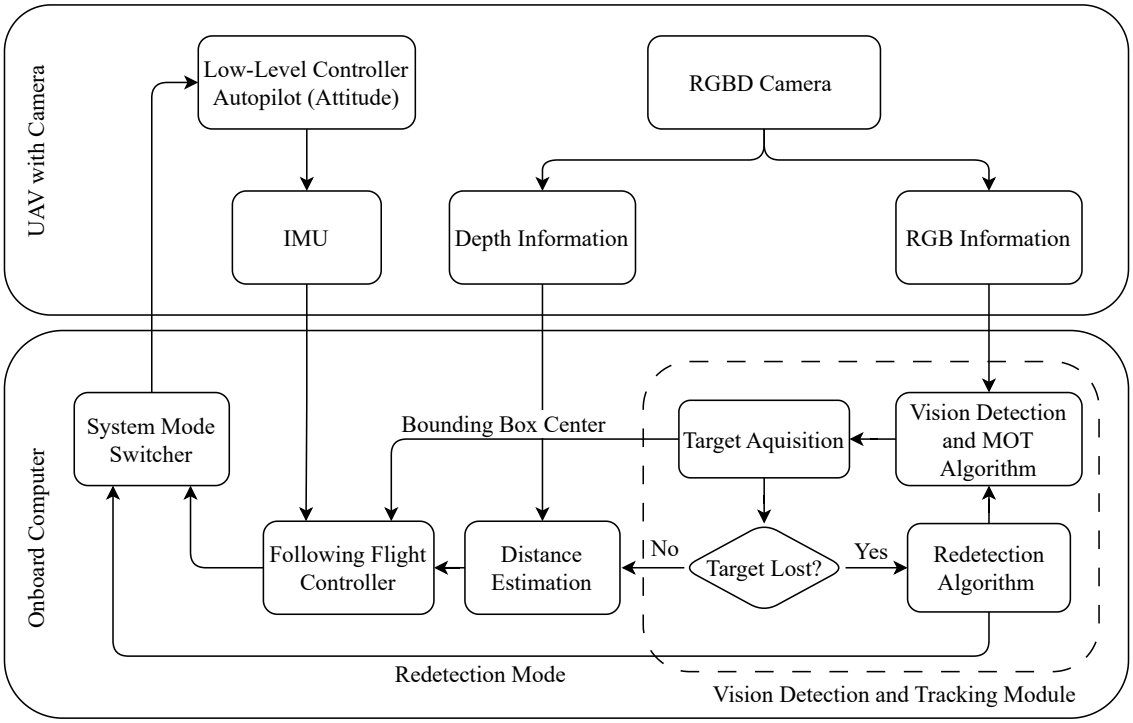


Figure 4.1: System Configuration of the vision-based detection, tracking and following system.

in the image; the Target Acquisition stage where a Kalman Filter is used to estimate the position of the selected target in the image; and the Redetection Algorithm, which is responsible for assessing the mode of the target and redetect it if possible. This module is responsible for using the RGB data from the camera and performing the detection and tracking of all the people in the image, including the target determined to be followed. It is also responsible for clearly differentiating between the target and the bystanders during normal operations or in the event of occlusions. In case of a redetection, the ”Redetection Mode” is sent forward to the System Mode Switcher to determine the control output.

The bounding box information from the ”Vision Detection and MOT Module” is then sent to the ”Distance Estimation” stage to get an estimate of the relative distance from the UAV to the target as control error for the ”Following Flight Controller”. The distance estimation may also be obtained from a depth sensor, such as ones included in a RGB-D camera, when available, due to the natural limitations of depth use from larger distances. From the position of the target relative to the center of the image and the estimated distance, the controller then computes the yaw rate and velocity commands for the UAV, respectively, and sends them to the ”System Mode Switcher”. In turn, it will determine which inputs are sent to the autopilot according to the current mode of the system. The autopilot controls the attitude of the UAV and sends the IMU data as feedback to the controller.

Without loss of generality, the proposed system will undergo testing in complex scenarios featuring multiple pedestrians in motion, where challenges like occlusions and disturbances are likely to arise. Notwithstanding, this system can be applied to many objects of interested by changing the training dataset used for the object detector and re-tunning the control parameters. Variations for distance estimation methods for other object types are also mentioned.

4.2 Visual Detection and Tracking Module

The first module is the Visual Detection and Tracking Module responsible for processing the images from the camera, identifying the target and performing necessary redetections.

4.2.1 Object Detection and Tracking

The first step to achieve target following is to detect the target and all bystanders in the image, and assign each one a unique identifier (ID) for tracking over time. This is accomplished by using a track-by-detection approach where a real-time object detector is used to identify target objects within individual frames. For the detections, the system takes advantage of the state-of-the-art detector YOLOv8, utilizing the smallest and fastest model YOLOv8-nano to get the best detection speeds. The model used is pre-trained with the COCO128 dataset [57] and, for the purpose of this thesis, will only perform the detection of the ”person” class. As mentioned in Section 2.1.2, the YOLOv8 model will analyse the image in real-time and provide a list of detections, highlighting all the objects in the image that are identified with the ”person” class above a threshold confidence level. These detections contain: bounding box surrounding the object, class of the object (”person”, in this case) and confidence score in that result. An example of a detection scenario from this thesis both in simulation and in real-world scenarios is shown in Figure 4.2.

(a) Simulation detection.

(b) Real-world detection.

Figure 4.2: Detection and tracking frames with ID, class (”person”) and confidence score ($\in [0, 1]$).

The confidence score level in this thesis was set to a low value of only 0.25 confidence, in order to increase the robustness to partial occlusions. Drawbacks of low confidence levels include some miss-detections that can include, for example, shadows of people in the image as shown in Figure 4.3. However, this does not influence the capacity of the system to successfully detect, track and follow the selected target, hence it can be considered a good trade-off.

The detections (composed of the bounding boxes and classes) are then linked over time by the tracker, by attributing unique identifiers (IDs) for each object, as also shown in Figure 4.2. Two state-of-the-art trackers, mentioned in the Chapter 2, that are worth to consider are the BoT-SORT MOT and

Figure 4.3: Detection of a shadow as a person derived from low values of confidence score allowed.

its predecessor, ByteTrack. YOLOv8 provides the detections via bounding boxes to the tracker which in turn performs data association with the previous frame to match each detection with a corresponding ID.

4.2.2 Target Acquisition and State estimation

After take-off and initialization of the detection and tracking algorithm, users have the option to select the target to follow from the list of detected people. In this thesis, the algorithm automatically designates the first person detected as the target to follow, storing its respective ID for subsequent frames. In order to take full advantage of the MOT capabilities, the position of other people in the image and respective assigned IDs will also be stored to enhance the redetection capabilities in case of target occlusion.

After detection, a Kalman Filter is used to predict the position of the target in the image even after occlusion [43]. A constant velocity motion model [56] is applied, using the velocities calculated from the movement of the center of the target in the image. The states used are the center coordinates of the bounding box (C_x, C_y), the size of the bounding box (width - w, height - h) and the speed of movement in the image ($\dot{C}_x$, $\dot{C}_y$), calculated from the movement of the center of the bounding box pixel coordinates. The states are defined in Equation (4.1).

$$s(t) = (C_x, C_y, \dot{C}_x, \dot{C}_y, w, h). \tag{4.1}$$

The observations are shown in Equation (4.2).

$$z(t) = (C_x, C_y, w, h). \tag{4.2}$$

The constant velocity model assumes the pixel coordinates of the center of the target (C_x, C_y) move in the image with constant speed ($\dot{C}_x$, $\dot{C}_y$) and direction. Hence, the Kalman Filter can predict the position

of the target during temporary occlusions. The Kalman Filter updates with new observations every time the target is visible. When the mode is set as "Target Missing", the predictions from the Kalman Filter are sent to the controller as the input to continue target following, since it is likely to reappear in the next few frames. If the target does not show after a set amount of frames, the Kalman Filter predictions are used in the redetection process to evaluate potential redetection candidates.

The dynamics of the system can be derived assuming no control input u_k which is the common approach in target tracking scenarios. Hence, the dynamics can be defined as shown in Equation (4.3).

$$x_k = Ax_{k-1} + w_{k-1} \Leftrightarrow \begin{bmatrix} c_{xk} \\ c_{yk} \\ \dot{c_{xk}} \\ \dot{c_{yk}} \\ w_k \\ h_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta T & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{xk-1} \\ c_{yk-1} \\ \dot{c_{xk-1}} \\ \dot{c_{yk-1}} \\ w_{k-1} \\ h_{k-1} \end{bmatrix} + W_{k-1}, \tag{4.3}$$

where ΔT represents the difference between the latest and the current time step and W_{k-1} represents a zero mean Gaussian noise. The observation matrix can be given by Equation 4.4:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{4.4}$$

4.2.3 Redetection algorithm

In vision-based detection and tracking of mobile targets from UAVs, occlusions pose a significant challenge, especially in densely populated areas. Redetection algorithms are crucial for reliable operations, particularly in scenarios involving multiple individuals. In such systems, the availability of multi-target information is a valuable asset to enhance the redetection algorithms and prevent the loss of the target.

Starting from the beginning of the target following process, the algorithm verifies detections using the assigned IDs from the YOLOv8+BoT-SORT or YOLOv8+ByteTrack setups by matching the received detections IDs with the locally stored detection list IDs. If the assigned target ID is absent in the detections, three possible situations may arise:

1. **Target ID change:** Limitation of MOT algorithms that occurs when the target is still visible in the image, however, the MOT algorithm assigned it a different identity than the previous frame. This may happen due to sudden movements of the target or the UAV that cause the tracker to miss-identify the target. Since the algorithm identifies the target via the assigned ID, an identity change would cause a system failure if left unattended. Hence, it is necessary to update the locally defined target ID with the new assigned ID from the MOT module for continuous target following.

This redetection mode allows the system to keep accurate target tracking and following, despite the MOT limitations.

2. **Target missing:** Initiates when the target is no longer detected in the image, potentially due to occlusions or obstructions by other objects, and a counter is initiated. During this step, the system continues to follow an estimate of the position of the target given by the Kalman Filter. The brief window where the Kalman Filter is used allows for a more robust system that will not immediately stop for short occlusions, like when two people cross paths. The Missing stage also adds some robustness to fast target movements that would cause the target to leave the image, by continuing the corrective maneuver even after the target leaves the field-of-view.

3. **Target lost:** Declared after a set of consecutive frames where the target is absent, suggesting a potential loss or concealment. To prevent further deviation from the hidden target, the UAV will stop and hover, looking for potential redetection candidates until the target is redetected. In this stage, a relocation strategy could also be considered in future works to position the UAV in a better view to redetect it.

To perform the redetection process, the algorithm searches all detections for possible redetection candidates. Firstly, if the target ID is not found, it will attempt to check if a ”Target ID Change” is in place. If no match is found, then it will enter the ”Target Missing” mode and later the ”Target Lost” mode.

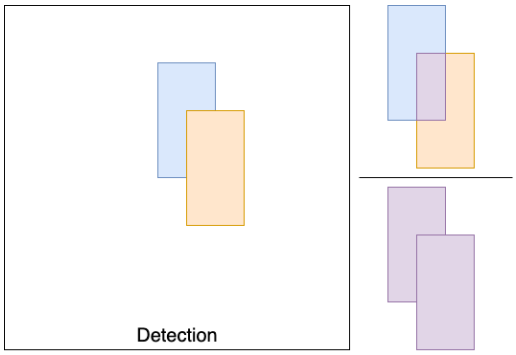
A viable candidate for a successful redetection must fulfill the following conditions:

1. It must represent a new detection not previously tracked, thereby excluding individuals already accounted for as they cannot be the target. This effectively excludes all the bystanders in the area and allows for redetections even in dynamic areas.

2. Candidate detections are evaluated based on a minimum interception threshold with the latest estimated position of the missing target, determined using an IoU approach (Equation (4.5)). The bounding boxes are scaled to allow for a greater recovery range.

3. Among all the candidates that fulfill step 1 and step 2, it must be the one that scored the highest on step 2.

The IoU approach is a common evaluation method to determine similarity between two bounding boxes in an image. It is given by the relation in Equation (4.5) which can be visualized in Figure 4.4.



$$IoU = \frac{Area(B_A \cap B_B)}{B_A \cup B_B} \quad (4.5)$$

Figure 4.4: Representation of the IoU calculation.

If the target is redetected during the ”Target ID Change” or the ”Target Missing” modes, the UAV will immediately resume regular operations and continue following the target. If the redetection happened during the ”Target Lost” state, then the System Mode Switcher will change the input to the controller accordingly, in order to resume target following operations after a complete stop. Further explanation will be given in the System Mode Switcher Section, Section 4.5.

The full candidate evaluation process can be summarized with the pseudo-code in Algorithm 1.

Algorithm 1 Redetection Algorithm

```
function REDETECTION ALGORITHM(detections)
    if target.id in detections then                          ▷ Check if the target is detected
        target_found
    else
        target_not_found
    end if

    for detection in detections do                           ▷ Loop through all the detections
        if detection.id in existing_object_list.id then      ▷ Check if it was a previously detected object
            Update_Object(detection.id, detection.bbox)

        else                  ▷ New detected object, add it to the list. Checks 1st redetection condition
            new_object = Detected_Object(detection.id, detection.bbox)
            existing_object_list.append(new_object)

            if target_not_found then          ▷ Valid candidate for redetection. Check the 2nd condition
                IOU = Calculate_IOU(detection.bbox, target.bbox)
                if IOU > best_IOU then
                    best_IOU = IOU
                    best_detection = detection
                end if
            end if
        end if
    end for

    if target_not_found and best_IOU > threshold then        ▷ Use the best candidate - 3rd condition
        target.id = best_detection.id
        Update_Object(best_detection.id, best_detection.bbox)
    end if
end function
```

4.3 Distance Estimation

Distance estimation is a crucial step to be able to effectively follow the target. This value can be roughly estimated based on the detection bounding box provided by the ”Visual Detection and Tracking Module” or more accurately obtained by a depth sensor installed. Despite the preference for depth information, it is not always available in most configurations and will have a shorter range limit than its RGB counter part.

To estimate the distance (d in Figure 4.5), the system used the relation between the pixel height (h) of the target in the image and a tuned constant value C as shown in Equation (4.6):

$$d = \frac{C}{h}. \tag{4.6}$$

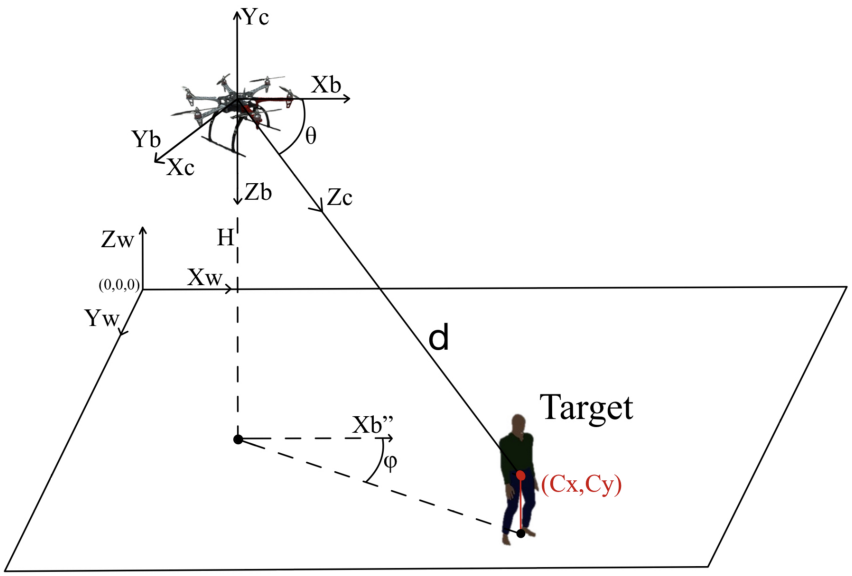


Figure 4.5: Relative distance to the target d and reference frames - UAV body frame F_b, Camera frame F_c and World frame F_w. θ represents the camera pitch angle. H represents the altitude of the UAV. X_b'' is the parallel line to X_b in the plane. ϕ represents the heading of the UAV.

The height of the target in the image is chosen as reference over the area occupied by the bounding box ($w * h$) [43] since it mainly depends on: the real height of the target, the distance to the target and known camera parameters. On the other hand, the width of the target in the image may vary according to the direction of movement, and position of the arms and legs. The constant value can be previously tuned for the average height of a human and adjusted during operations after getting a better estimate from on-board depth sensors. In non-person target tracking and following (such as vehicles or robots), the area of the bounding box can be considered.

The second distance estimation method is the depth-based estimation from the RGB-D camera. As mentioned, depth accuracy varies according to the distance to the target, being the most effective after the UAV reaches close proximity to the target and possibly not available for greater distances. For the Realsense camera used in the experimental setup, the effective range overs around the 10 meter mark. In this sense, the system utilizes the visual-based distance estimation method to approach the target

and the the depth information to fine tune target following.

Taking into consideration the depth accuracy from the RGB-D camera, a distance threshold is defined, from where the system will switch from the visual-based distance estimation method to depth-based estimation if it receives valid data. The depth-based estimation method will take the depth estimate from the center of the bounding box of the target (marked as (Cx, Cy) in Figure 4.5) and calculate the distance to the point using depth. The distance is calculated using the Pinhole Model explained in Section 3.1. The 3D coordinates of the center point of the target are computed using the following relations in Equation (4.7). The principle point coordinates ppx and ppy and the focal lengths f_x and f_y are obtained from the intrinsic parameters of the camera.

$$\begin{aligned} X_C &= \frac{Cx - ppx}{f_x}, \\ Y_C &= \frac{Cy - ppy}{f_y}, \\ Z_C &= depth. \end{aligned} \tag{4.7}$$

where the distance d is computed using Equation (4.8).

$$d = \sqrt{X_C^2 + Y_C^2 + Z_C^2}. \tag{4.8}$$

The distance estimation can be susceptible to observation noise. To achieve smooth control, an exponential low-pass filter in a discrete-time system is used as shown in Equation (4.9). d_{low_pass} is the filtered value, $d(t_{k-1})$ is the previous estimation and $d(t_k)$ is the current estimation. The α value is tuned to prevent higher frequency oscillations in the measurements that could disrupt the controller.

$$d_{low\_pass} = (1 - \alpha) * d(t_{k-1}) + \alpha * d(t_k),\ \alpha \in [0, 1]. \tag{4.9}$$

4.4 Following Flight Controller

The Following Flight Controller is composed of three separate controllers to achieve accurate 3D following of the target: one for the yaw rate, one for the altitude (vertical velocity) and a third one to control the horizontal velocity of the UAV.

4.4.1 Yaw Rate and Vertical Velocity Controllers

This work draws inspiration from [37, 42], employing the deviation between the pixel location of the target and the central position of the image frame as feedback for maintaining the focus of the UAV on the target. The yaw rate and altitude controllers are proportional controllers designed to center the target in the image. Normalized references for the horizontal, and vertical pixel positions used are shown in Equation (4.10).

$$X_N = \frac{C_x}{ppx} - 1, \\ Y_N = \frac{C_y}{ppy} - 1. \tag{4.10}$$

The yaw rate for the controller and vertical velocity are then computed using Equation (4.11) and Equation (4.12) respectively. The values K_ϕ and K_H are the proportional gains, tuned for each controller.

$$\dot{\phi_{ref}} = K_\phi * X_N, \tag{4.11}$$

$$V_z = K_H * Y_N. \tag{4.12}$$

4.4.2 Horizontal Velocity Controller

The algorithm uses an aim-and-approach strategy similar to the PN strategy in [10] to achieve target following, by using the yaw rate and altitude controllers to aim and the horizontal velocity controller to approach the target. The control output for the horizontal velocity is calculated by using a PI controller taking the estimated distance as input. The horizontal velocity controller error is defined according to Equation (4.13).

$$error = d - d_{desired}. \tag{4.13}$$

Where the desired distance $d_{desired}$ is defined by the user according to the following standards:

- Camera resolution, field-of-view and camera angle - from the desired distance the UAV must be able to capture the full size of the target to be followed (lower limit) and needs to have enough resolution to be able to detect the target (higher limit).

- Speed and direction changes - depending on the goal of the mission, a higher desired distance enables a larger captured area which in turn allows more robustness to target speed and direction changes. On the other hand, a lower desired distance allows for greater use of depth information and higher precision in target following, but lower robustness to sudden target movements.

In order to achieve smooth control, the velocity value V goes through a slew rate limiter that prevents sudden and aggressive maneuvers which may cause loss of line-of-sight. The slew rate limiter also sets the initial control output as zero, allowing for controlled initial movement. The limited rate of change SR is defined according to Equation (4.14). The slew rate limiter algorithm is illustrated in Algorithm 2.

$$SR = \frac{V(t_k) - V(t_{k-1})}{t_k - t_{k-1}}. \tag{4.14}$$

In addition to the direct distance between the UAV and the target, the distance in the horizontal plane is also considered as a limiter as to prevent the UAV from overshooting the target. This horizontal distance ($d_{horizontal}$) is calculated according to the current altitude of the UAV, an estimate of the target's

Algorithm 2 Slew Rate Limiter Algorithm

function RATE_LIMITER($V(t_k)$) ▷ Takes the current velocity value as input
  $T \leftarrow t_k - t_{k-1}$
  $SR \leftarrow [V(t_k) - V(t_{k-1})]/T$
  if $|SR| \leq$ max_rate **then**
    $V_{SR} \leftarrow V(t_k)$
  else
    $V_{SR} \leftarrow V(t_{k-1}) \pm$ max_rate $* T$ ▷ sign is positive if $V(t_k) > V(t_{k-1})$ and vice-versa
  end if
  return V_{SR} ▷ Returns the limit velocity
end function

height and assuming the target is in the same plane as the UAV measured altitude as shown in Equation (4.15). This value can also be compromised by the quality of the altitude measurement, which is why it is only used as a limiter and not as the error feedback for the controller.

$$d_{horizontal} = \sqrt{d^2 - (H - \frac{target\_height}{2})^2}. \tag{4.15}$$

Once the horizontal distance goes bellow the set threshold, the limiter will gradually reduce the output from the velocity controller. If the critical horizontal distance value is reached, then the forward velocity will be set to zero.

Figure 4.6 shows the workings of the high level controller. It takes the information from the RGB-D camera to compute the forward velocity, the vertical velocity and the yaw rate, which are then sent to the Flight Control Unit (FCU), the Pixhawk autopilot.

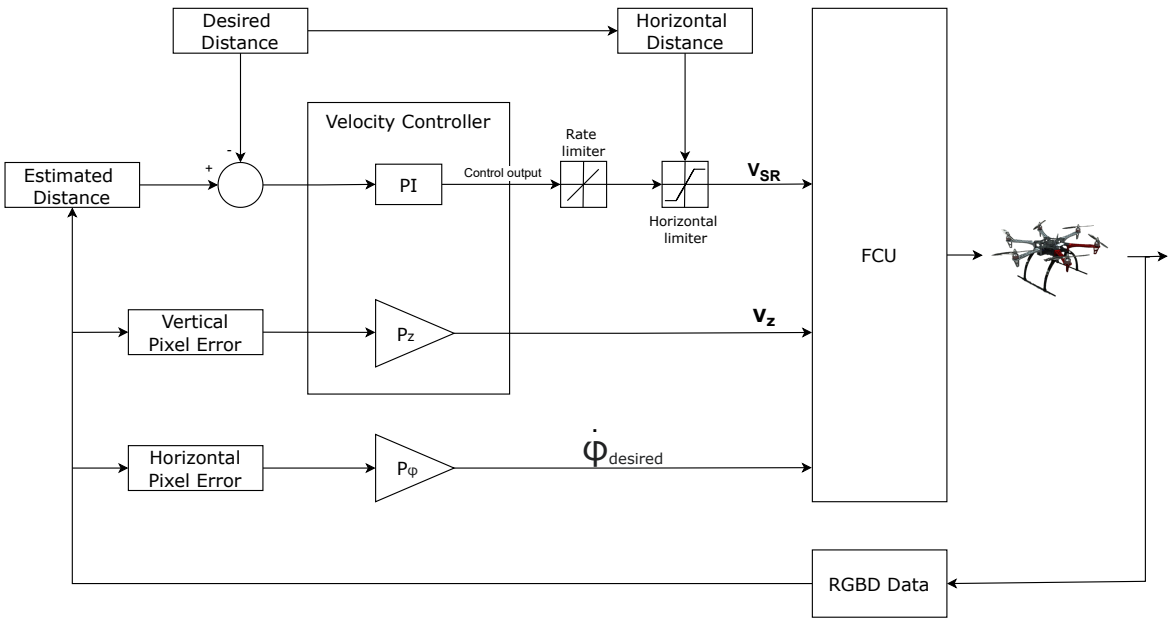


Figure 4.6: Representation of the UAV High-level Controller.

4.5 System Mode Switcher

The System Mode Switcher is the responsible for providing the low-level controller with appropriate control references based on the mode of the system.

The system can operate in the following modes:

- **Search Mode:** no target has been detected and identified yet. The UAV performs a predetermined search pattern until the target is found.
- **Adjusting Mode:** only the yaw rate and vertical velocity controllers are used to center the target in the image.
- **Following Mode:** this may be defined as the mode for standard operations, when the target is identified and followed in 3D. This is also the chosen mode if a Target ID Change occurred.
- **Target Missing Mode:** as previously defined in Section 4.2.3, the target is not visible in the image.
- **Target Lost Mode:** also defined in Section 4.2.3, if the target has been missing for several consecutive frames it is assumed lost/hidden. Adequate procedures are taken to prevent further deviations from a target hidden behind an obstacle.

Figure 4.7 presents a flowchart representing the system modes and their respective interactions.

Following take-off and systems check, the UAV will enter ”Search Mode”. In this mode, the UAV will perform a predetermined search pattern until the target is found. In this thesis, the UAV will perform a climb until the defined safety altitude and slowly rotate until a person is detected. The first person detected will be set as the the target to follow. The system allows the search mode to be redefined according to the specifications of the mission (e.g. making it possible for the user to select the specific target to follow).

After the target is detected and identified, the UAV will enter ”Adjusting Mode”, adjusting the target to the center of the image as best as possible, considering altitude safety limits by controlling the altitude and yaw rate. This allows the system to have a more controlled first approach to the target. After the ”Adjusting Mode” timer is over, the system enters regular operations with the ”Following Mode”, which feeds the output from all the controllers to the autopilot.

Regarding the first recovery mode - ”Target Missing Mode”, the System Mode Switcher will continue to send the commands from all the controllers however, the estimated distance and subsequently the error will be calculated using the predictions from the Kalman Filter.

Finally, in the second recovery mode - ”Target Lost Mode”, the System Mode Switcher will set the UAV to hover. This is done to prevent further deviation from the lost target, here assumed to be hidden behind some obstacle. The system will remain in this mode until a successful redetection is obtained. Unlike the ”Target Missing Mode” that directly shifts to ”Following Mode”, here, once the target is redetected, the System Mode Switcher will initiate the ”Adjusting Mode” for a brief period and reset the rate limiter and the low pass filter previous values. This will ensure the target is properly redetected and followed.

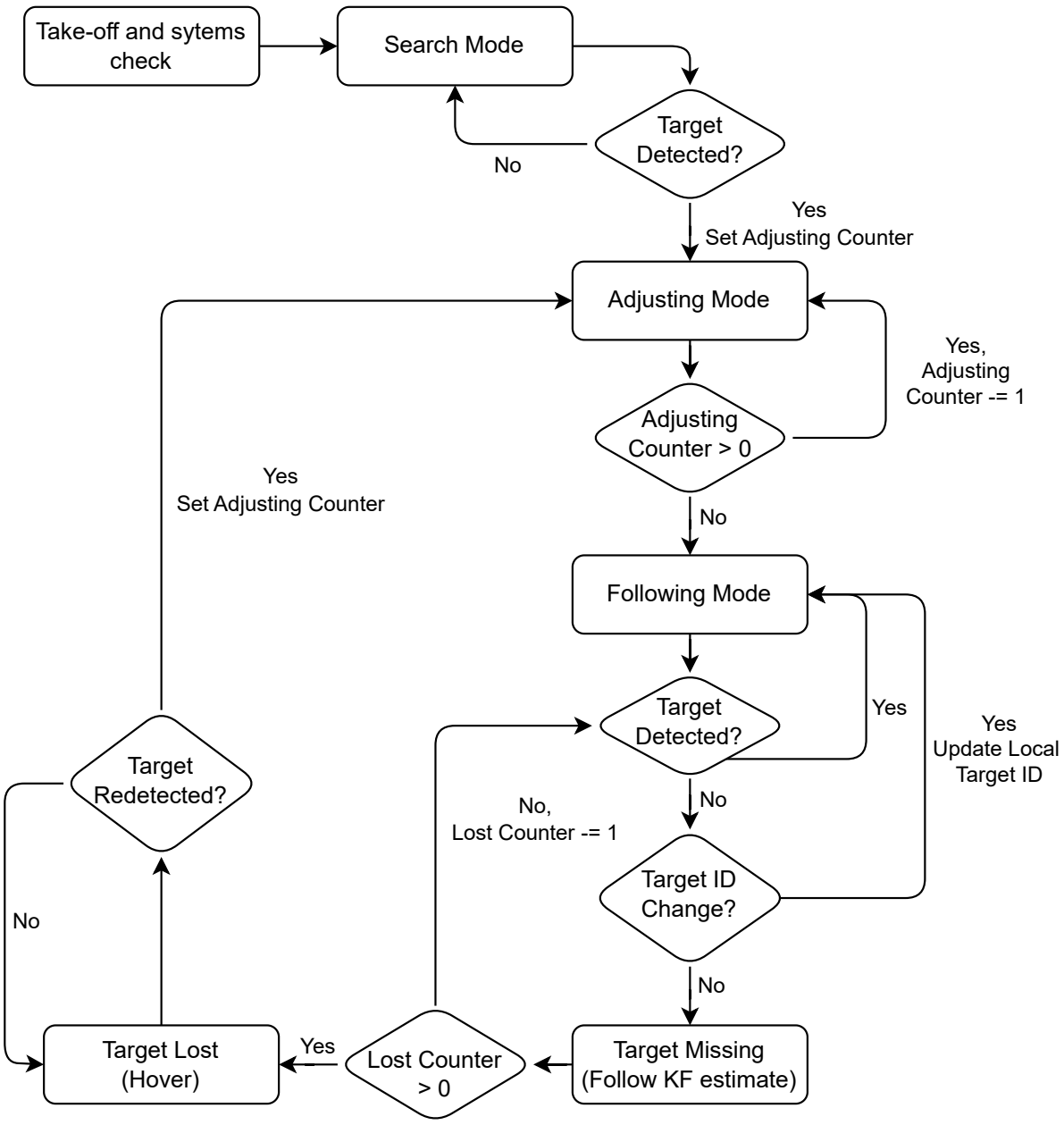


Figure 4.7: Flowchart of the System Mode Switcher.

Chapter 5

Experiments and Results

This chapter presents the experiments conducted to evaluate the vision-based target tracking system, encompassing both simulation and real-world tests. The chapter is divided into two main parts: in the first half of the chapter, the simulation setup and results are shown; in the second half of the chapter, the real-world experimental setup and platform are described along with the results obtained during flight-tests. The performance of two MOT trackers, BoT-SORT and ByteTrack, is compared and evaluated within the system. Additionally, the chapter includes a thorough analysis of the redetection capabilities in various situations, demonstrating the robustness of the system. The UAV used for these experiments was also fully assembled during development of this thesis, a process described in the real-world experimental setup section.

Performance metrics are as follows:

- **UAV and Target Distances**: Total travel distance for the UAV and the target, respectively, in each experiment.
- **Visual Accuracy**: The percentage of frames where the target is correctly identified;
- **Depth Use**: Percentage of time the depth sensor data is utilized for position estimation;
- **Estimation error**: Average error and standard deviations between the estimated distance d and the real distance measured relative to ROS-Gazebo ground-truth, therefore is only used in simulation experiments.
- **Frame Rate**: Rate of frames per second that are processed by the visual detection and tracking module, which in turn gives an accurate representation of the computational speed of the algorithm.
- **Number of ID Changes**: Used to measure target tracking performance and indicates how many times the ID given to the followed target changed. This metric is used to assess the MOT performance in this thesis, and does not consider the developed redetection algorithms that correct for target ID changes, by locally correcting the assigned target ID to the new value.

5.1 Robot Operating System - ROS Implementation

The full system is implemented using the Robot Operating System - ROS framework [58]. ROS is an open-source platform to enable efficient communications between different parts of a system in a robot. These parts are called ”nodes” and are able to perform separate functions. ROS enables the different nodes to communicate information in real-time by deploying a system of ”publishers” and ”subscribers”, which are used inside each node. Information in ROS is published to specified topics. Each publisher populates the topics with information that can be read from any node by subscribing to the topic.

For this thesis, there are 4 main nodes: the camera node, the visual detection and tracking node, the MAVROS node and the target following node. Figure 5.1 shows the full topic tree and the respective topics exchanged between the nodes.

The github repository with the full implementation can be found in the footnotes [1].

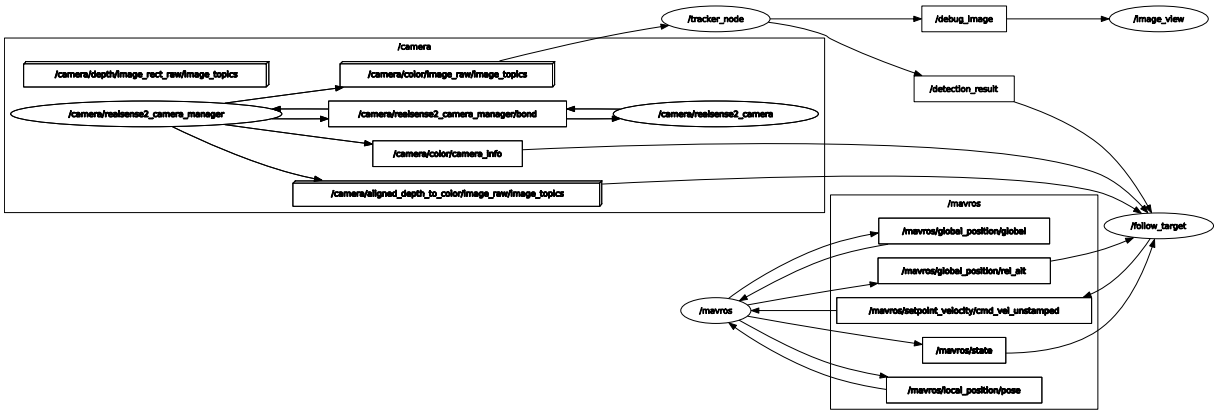


Figure 5.1: Representation of the topic tree for the system. The main topics are shown separated into the /camera node, the /tracker node, the /mavros node and the /follow_target node.

5.1.1 Camera Node

The camera node is responsible for handling the hardware interface with the chosen RGB-D camera and publishing all the necessary information for the Visual Detection and Tracking Module. The full topic list can be found in [59]. The following published topics are used in the system:

- **/camera/color/camera_info**: message of type sensor_msgs/CameraInfo that contains the camera parameters such as the intrinsic parameters of the camera.
- **camera/color/image_raw**: message of type sensor_msgs/Image that contains the RGB image from the camera.
- **camera/depth/image_raw**: message of type sensor_msgs/Image that contains the raw depth data from the camera. This message is then transformed into **camera/align_depth_to_color/image_raw** which contains the depth values aligned to the RGB pixel positions, something necessary in order to match the detections of YOLO to the respective depth values.

[1] https://github.com/diogoferreira08/Target-Following-from-UAV-using-MOT.git

5.1.2 Visual Detection and Tracking Node

The visual detection and tracking node (/tracker node in Figure 5.1) is the node responsible for running both YOLOv8 and the chosen tracker (BoT-SORT or ByteTrack) for each frame [60]. This node subscribes to **camera/color/image_raw topic**, processes the image through YOLOv8 and the tracker and publishes a list of detections to the topic /**detection_result**. This topic contains a list of all the detections in the image, which include the bounding box coordinates for each detected object, its class, the class probability and the assigned ID from the tracker.

5.1.3 MAVROS Node

Referencing the system description of Figure 4.1, there must be proper bi-lateral communication between the onboard computer and the autopilot to send control commands and to receive attitude and positional information from the autopilot. This communication is achieved by using MAVROS, which is a ROS package developed to accurately send and receive MAVLINK messages. Since Pixhawk is using the MAVLINK protocol for communications, and the proposed approach of this thesis is built on top of the ROS Noetic system, then MAVROS is the most efficient way to transfer information.

The MAVROS [61] node is responsible for the interface between the Pixhawk and the Jetson. It publishes all the data from the FCU and receives the commands from the System Mode Switcher.

For the proposed approach we use the following MAVROS topics:

- **mavros/setpoint_velocity/cmd_vel_unstamped**: topic used to send the output of the System Mode Switcher to the Autopilot. It accepts a geometry_msgs.msg/Twist() message that includes linear velocities and angular velocities in the chosen frame of reference. For this system, the linear velocities are used for the vertical and horizontal velocity controllers and the angular velocity in Z is used for the yaw rate controller.

- **mavros/mav_frame**: this message contains the frame of reference used to send velocity commands. In the real-world tests, the BODY_NED frame was used. This frame allows the velocity to be sent in relation to the body of the UAV in a stabilized frame with the x-axis forward and horizontal, the y-axis to the right and the z-axis downwards.

- **mavros/global_position/rel_alt**: altitude measured in relation to the arming altitude.

- **mavros/imu/data**: Inertial Measurement Unit (IMU) data from the Pixhawk. It provides the attitude and orientation of the UAV in quaternions which are then translated to Euler angles.

- **mavros/state**: current mode state of the UAV. Needs to be set to ”GUIDED” in order for the system to accept velocity commands from MAVROS. For regular take-off and manual flight, the state is set to ”LOITER”, ”ALT-HOLD” or ”STABILIZE” modes. The search mode is activated the moment the mode is set to ”GUIDED” and the system will go back into manual flight if the ”GUIDED” mode is deactivated.

- **mavros/local_position/pose**: topic used by the autopilot to process the inner loop attitude control of the system in order to follow the high level commands (yaw rate and velocity) from the developed controllers.

5.1.4 Target Following Node

Lastly, the target following node encompasses the part of the system after receiving the detection results from the vision detection and tracking node until the commands are sent to the MAVROS node. This node subscribes to the following topics:

- /**mavros/state**, since the current state of the UAV is needed to know when to start the system and send commands.

- /**mavros/global_position/rel_alt** used to limit the actuation of the vertical velocity controller within safety heights.

- /**camera/align_depth_to_color/image_raw** used to compute the distance to the target using depth information.

- /**detection_result** contains the detections from YOLOv8+BoT-SORT or YOLOv8+ByteTrack.

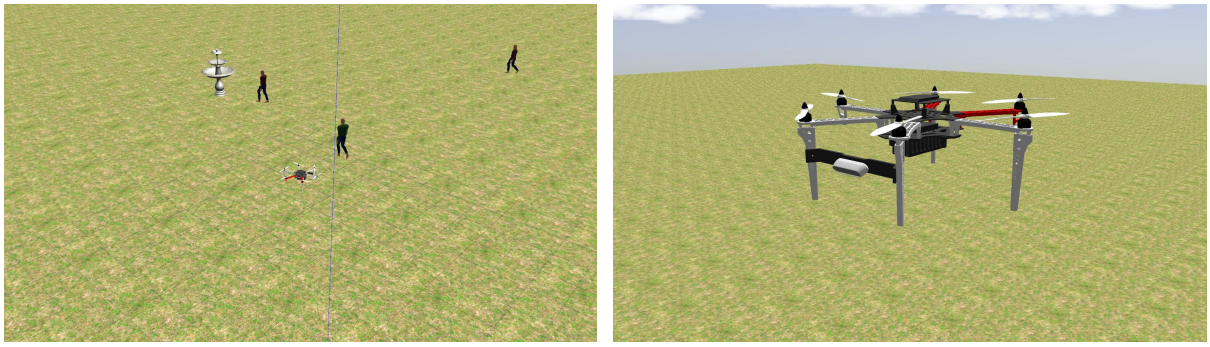
- /**camera/color/camera_info** used to obtain the intrinsic parameters of the camera needed for the system.

For the publishers, the target following node has one mandatory and one optional publish topic. The mandatory topic is the /**mavros/setpoint_velocity/cmd_vel_unstamped** used to publish the velocity and yaw rate message to MAVROS. The optional topic is a custom message named /**rosbag_data** used to store valuable information from the flight tests in order to evaluate the system.

5.2 Simulation environment setup

The system was implemented and tested in simulation by using the MRS UAV System [11], which provides valuable data to prepare for real-world deployment. The experimental setup was a Hexacopter DJI-F550 equipped with a front-facing RGB-D camera Realsense D435 with a 1280x720 resolution installed at a $30°$ angle from the horizontal position, as shown in Figure 5.2(b). The low-level control and IMU acquisition are managed by a Pixhawk FCU. Simulations are conducted within Gazebo-ROS as shown in Figure 5.2(a).

Pedestrians are simulated utilizing the actors from Gazebo-ROS [62], which allow for easy trajectory manipulation using waypoints. In order to perform randomized experiments, a world generation script was developed to generate randomized waypoints based on the following defined conditions for the experiment: maximum and minimum number of pedestrians, average walking speed, available area for the experiments, expected duration of the experiment.



(a) Simulation Environment. (b) Simulation UAV platform.

Figure 5.2: Simulation conditions.

5.2.1 Randomized Experiments

To assess system performance in open environments with multiple pedestrians, five tests with different conditions were conducted in a 900-square-meter area, featuring 3 to 6 pedestrians walking at an average speed of 1 to 1.5 meters per second. Each test spans over 2 minutes, with randomized and not pre-set trajectories. The desired distance ($d_{desired}$) is set to 11 meters, considering the Realsense's depth range in the simulation (10 meters) with an added 1-meter margin to enhance responsiveness to sudden target movements toward the UAV. In order to perform a comparison between MOT methods, the same world conditions (target waypoints) are replicated in the tests conducted using BoT-SORT in Table 5.1 and ByteTrack in Table 5.2. Across all the experiments, the UAV successfully tracked and followed the target, covering a total travel distance of 883.5 meters, while the target traveled for 1264.6 meters. This difference can be attributed to the fact that the UAV will sometimes only need to rotate to match the movement of the target.

Regarding the BoT-SORT results from Table 5.1, it can be noticed that the target is detected in almost all the frames with the lowest detection values of 99% in test number 4 which indicate that the YOLOv8

Table 5.1: Data from the 5 randomized experiments using the BoT-SORT Tracker.

Test No.	Time (s)	Distance UAV (m)	Distance Target (m)	Visual Accuracy (%)	Estimation Error (m)	Depth Use (%)	Frame Rate (Hz)	No. ID Change
1	137.3	91.9	123.8	99.38	0.568 ± 0.425	57.96	13.11	0
2	120.8	68.5	112.3	99.72	0.590 ± 0.570	68.43	13.25	0
3	149.5	97.0	147.2	99.71	0.547 ± 0.467	65.16	13.04	0
4	127.1	93.9	130.6	99.00	0.765 ± 0.550	51.73	9.66	1
5	151.8	94.8	118.4	99.67	0.680 ± 0.693	64.42	11.30	0

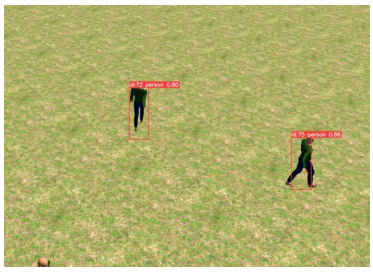
detection method has a high detection accuracy. ”Depth Use” shows that the depth estimation is used over 50% of the time in all the tests, which is expected given the desired distance values of 11m. In order to use depth information during a higher percentage of the tests, a lower desired distance value could be given, however this would implicate a higher risk of the target getting out of line-of-sight if there were sudden speed changes due to the reduced visible area in that configuration. Taking in consideration the dynamic use of both estimation methods for distance estimation, the error values are within a range that allows for accurate target following in dynamic scenarios. The standard deviation for the estimation values is relatively high compared to the average error, mainly due to the differences in accuracy between using the bounding box height or the depth information. Regarding computational speed, the BoT-SORT algorithms averages around 12,07Hz $\pm$ 1,57Hz which is adequate for real-time applications. In addition, the BoT-SORT algorithm proves to be very effective in target following, maintaining the target ID in all but the fourth experiment where the ”Target ID Change” redetection method had to be used once.

Table 5.2: Data from the 5 randomized experiments using the ByteTrack Tracker.

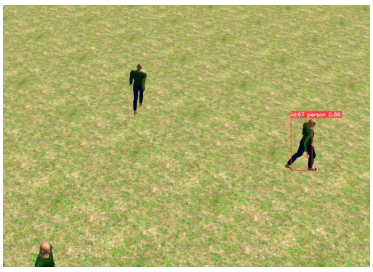
Test No.	Time (s)	Distance UAV (m)	Distance Target (m)	Visual Accuracy (%)	Estimation Error (m)	Depth Use (%)	Frame Rate (Hz)	No. ID Change
1	137.1	90.2	123.7	98.85	0.474 ± 0.370	62.72	18.44	5
2	120.1	67.6	110.7	99.26	0.505 ± 0.505	73.61	18.51	7
3	147.7	94.0	146.8	98.08	0.468 ± 0.390	66.91	17.30	10
4	135.1	93.7	136.9	97.64	0.602 ± 0.468	63.71	16.18	7
5	147.4	91.9	118.7	98.86	0.574 ± 0.552	62.13	17.35	10

For the ByteTrack results in Table 5.2, tests number 1 to 5 were conducted under similar conditions as the respective BoT-SORT experiments (same target movement and relatively the same durations). Regarding the number of ID Changes, it can be seen that there is a substantial performance drop from the BoT-SORT results, which derives from the lack of camera movement compensation enhancements in the ByteTrack algorithm. Despite the higher number of target ID Changes, the redetection algorithm developed in this thesis is able to handle this limitation and continue to follow the target even under these conditions. Also, ByteTrack results show a slight drop in ”Visual Accuracy” which can be attributed to the fact that there are more instances where the target ID is not being assigned by the tracker. This situation is illustrated in Figure 5.3 and typically occurs in the frame prior to an ID Change, due to a loss

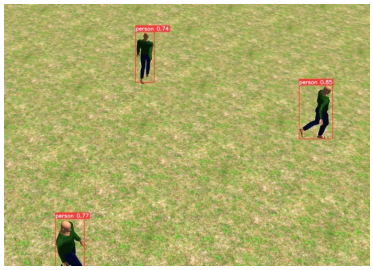
of target tracking by the MOT algorithm. In Figure 5.3(a), the target is followed with ID 72. Figure 5.3(b) shows a miss-detection frame where the target is not detected. In the next frame, Figure 5.3(c), the target is detected again, but the MOT does not match the new bounding box with the previous ID and begins processing a new track, marking the start of an ID change. At this point, the MOT is performing data association and has not yet assigned an ID to the target. Finally, in Figure 5.3(d), the MOT assigns a new ID (80) to the target. The redetection algorithm then identifies this Target ID change using the three-step process mentioned in Chapter 4 and updates the local target ID to the new value (ID 80). Regarding computational load, the simplicity of the ByteTrack algorithm in comparison with BoT-SORT becomes evident with an overall increase in frame rate of 5,49Hz across the 5 tests, with an average of 17,56Hz $\pm$ 0,96Hz which translates to a 31,25% decrease in runtime from the BoT-SORT results.



(a) Target (top left) followed ID 72.



(b) Target (top left) not Detected.



(c) Target (top left) Detected without ID.

(d) Target (top left) ID Change to ID 80.

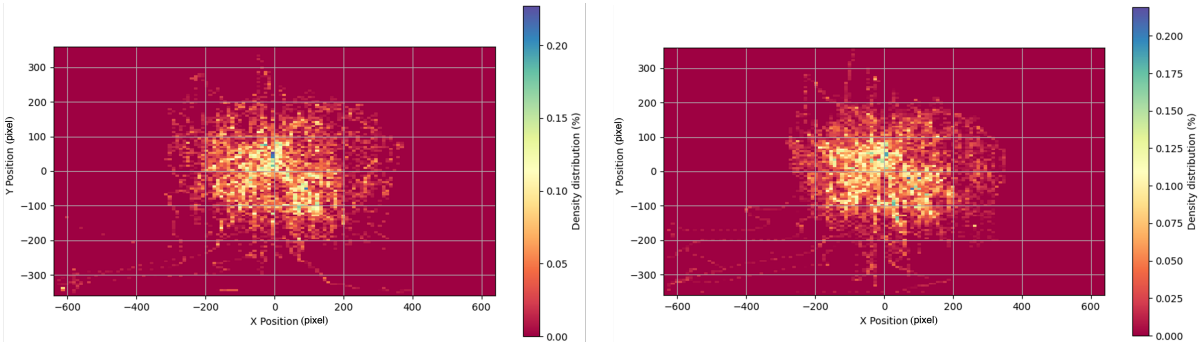
(e) Highlight of the target in Figure 5.3(c).

Figure 5.3: Target ID Change scenario with missed tracking frame during test number 5.

To evaluate the capabilities of the system to maintain line-of-sight and keep the target in the center of the image, the density distribution of the target's bounding box center position in the image was compiled across the five tests for each tracker and shown in Figure 5.4. Results are similar for both trackers with the target remaining predominantly centered. Target deviations along the vertical axis can be attributed to the lack of gimbal stabilization for the camera, coupling forward/backwards movement with a downward/upwards pitch maneuver. It is believed that introducing camera stabilization would greatly improve these results by decoupling both movements.

5.2.2 Long term target following experiment with occlusions

To further evaluate the full system, an experiment was conducted that includes partial and full occlusions up to 10 seconds where the system goes through the various redetection modes such as: Target ID Change, Target Missing and Target Lost. Similarly to the previous experiments, in this case the same scenario was also ran for both trackers in order to compare results which are expressed in Tab. 5.3.



(a) BoT-SORT tracking tests.

(b) ByteTrack tracking tests.

Figure 5.4: Density probability distribution of the center of the target in the image frame during the 5 tests totalling 1264.6 meters of target movement.

Table 5.3: Data from the experiment including target full occlusion.

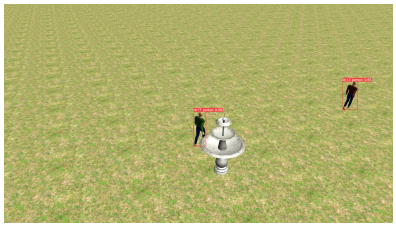
Tracker Used	Time (s)	Distance UAV (m)	Distance Target (m)	Visual Accuracy (%)	Estimation Error (m)	Depth Use (%)	Frame Rate (Hz)	No. ID Change
BoT-SORT	575	366.8	504.2	93.28	0.599 ± 0.502	60.72	9.33	6
ByteTrack	518	348.5	453.0	91.31	0.587 ± 0.640	60.80	15.07	46

Overall, results align closely with those of obstacle-free experiments, with the only notable difference being a decrease in ”Visual Accuracy.” This decrease can be attributed to instances where the target experiences full occlusions during the course of the experiment. The differences between the YOLOv8+BoT-SORT setup and the YOLOv8+ByteTrack are even more pronounced in the long-term experiments with the BoT-SORT tracker performing much better in terms of keeping the IDs of the targets while the ByteTrack tracker has the better frame rate. Considering the several instances of partial and complete occlusions and the number of ID changes, especially in the ByteTrack experiments, it can be said that the redetection methods performed well during both experiments, maintaining target following throughout the full experiments. Despite the redetection algorithm being able to handle target ID changes, the robustness of the BoT-SORT algorithm makes it more suitable for target following applications where dynamic scenarios can provoke unforeseen circumstances, which a more robust tracker will be able to handle. A demonstration video showcasing the entire experiment using YOLOv8+BoT-SORT is available in the footnotes[2]. The video includes useful time annotations, highlighting key mode transitions and emphasizing significant moments throughout the experiment.

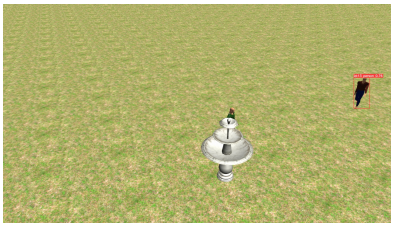
An example of an occlusion scenario following the target (green shirt) can be seen in Figure 5.5, which depicts the sequence of 6 frames: 5.5(a) before the target hides, 5.5(b) during the ”Target Lost” phase, 5.5(c) upon target reappearance, and 5.5(d), 5.5(e) and 5.5(f), subsequent to target redetection. As shown, the system correctly follows the target on the left even when a bystander is present in the image. Once the target hides behind the obstacle in Figure 5.5(b), the system enters ”Target Missing” and then ”Target Lost” stages without ever switching to the pre-existing bystander since it doesn't satisfy the first redetection condition - ”It must represent a new detection not previously tracked, thereby excluding

[2]Simulation YOLOv8+BoT-SORT long tracking experiment: `https://youtu.be/YrquRNc5tKM`

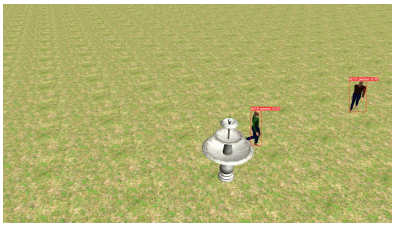
individuals already accounted for as they cannot be the target". Once the target exits the obstacle in Figure 5.5(c) and reappears in the detections, it fulfills all redetection conditions: it is a new detection that is closest to the last seen position of the target. Hence, in Figure 5.5(c) and Figure 5.5(d), the system redetects the target (green shirt) and goes into adjusting mode and then in Figure 5.5(e) and Figure 5.5(f) into following mode once more.



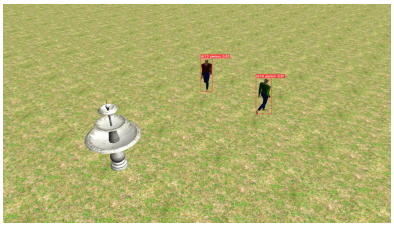
(a) Following the target (left). Bystander in the distance (right).



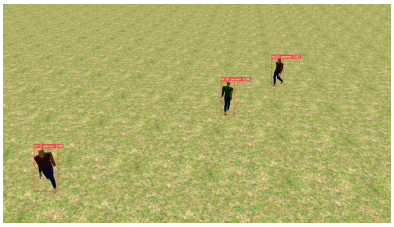
(b) Target (left) hidden behind obstacle.



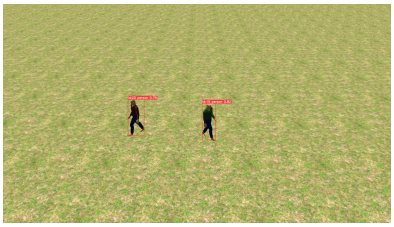
(c) Target reappears in the detections.



(d) Target (right) redetected - adjusting mode.



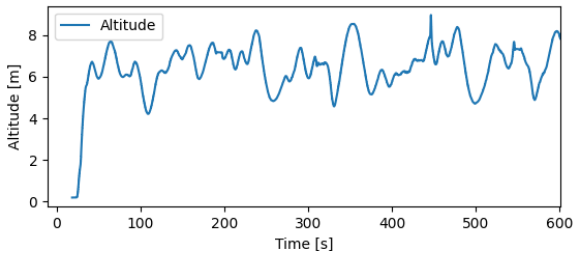
(e) Target (middle) followed.



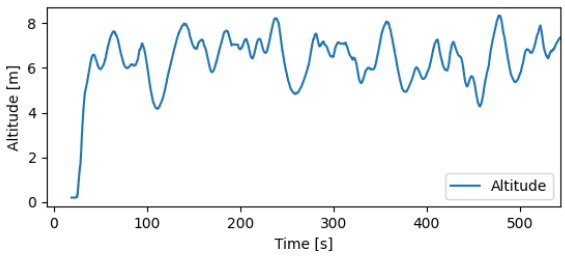
(f) Target (right) followed.

Figure 5.5: Redetection algorithm experiment - target successfully redetected after full occlusion behind an obstacle.

The altitude over time for both experiments is presented in Figure 5.6, showing similar results with altitude levels varying between 4 and 8 meters.



(a) Altitude in the BoT-SORT Experiment.

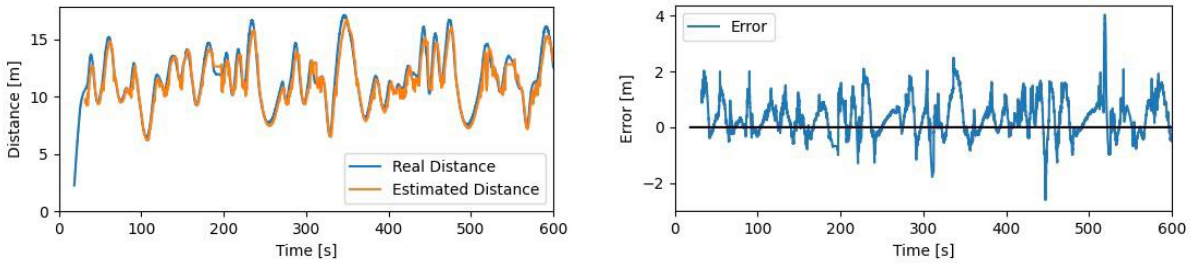


(b) Altitude in the ByteTrack Experiment.

Figure 5.6: Altitude over time during the experiments measured to the take-off level.

The estimated distance compared to the real distance and the respective error are shown for the BoT-SORT experiments in Figure 5.7 and for the ByteTrack experiment in Figure 5.8. It is evident that while the system maintains accuracy, certain frames exhibit noticeable peak errors. These peaks correspond to momentary fluctuations in the bounding box of the target when relying on visual information. This phenomenon is exemplified in Figure 5.9 for the peak near 520 seconds in the BoT-SORT experiment, where the bounding box undergoes temporary enlargement across a few frames, mainly due to the limitations of the YOLO detection process. The proposed system mitigates these limitations in two ways: first, by using depth information to eliminate dependency on the bounding box once the target is

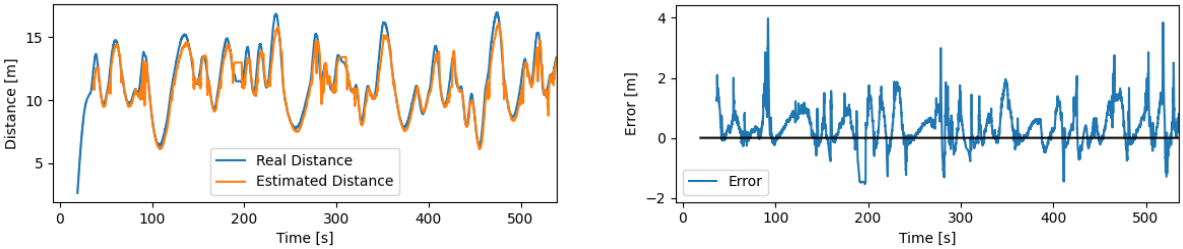
within the depth sensor's range; and second, by applying a low-pass filter to the distance values, which prevents high-frequency oscillations in the estimation from affecting the controller.



(a) Estimated vs Real Distances from the Target.

(b) Error between Real and Estimated Distances.

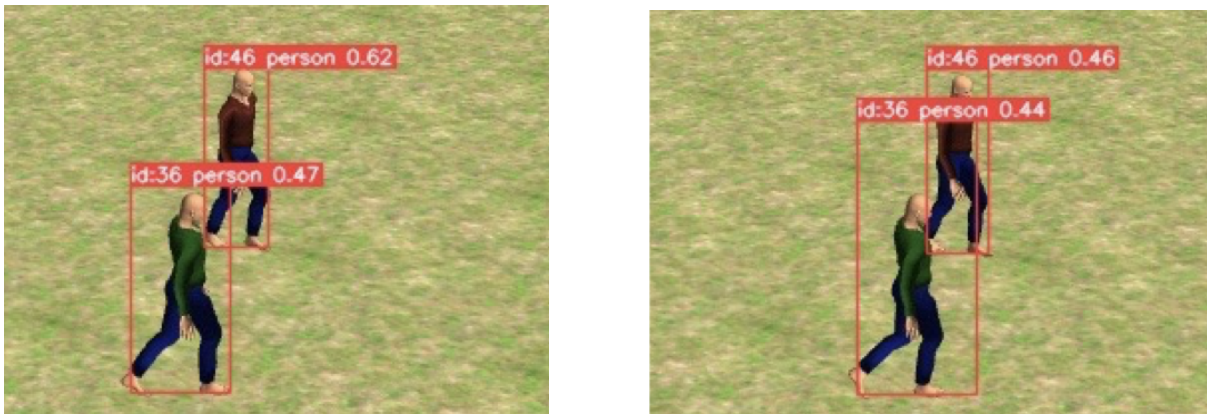
Figure 5.7: Estimated distance and respective error over time for the BoT-SORT experiment.



(a) Estimated vs Real Distances from the Target.

(b) Error between Real and Estimated Distances.

Figure 5.8: Estimated distance and respective error over time for the ByteTrack experiment.



(a) Frame preceding bounding box enlargement.

(b) Frame exemplifying bounding box enlargement due to close proximity between two people.

Figure 5.9: Bounding box enlargement during target tracking.

5.3 Real-world Experiments

After having the proof-of-concept finalized in simulation, and the full system evaluated, it is possible to safely transition the system into the real-world. This section covers the experimental setup and the results obtained. The first batch of real-world experiments were conducted in the outdoor facilities of Instituto Superior Técnico (IST) in Lisbon[3], specifically on the football court marked red in Figure 5.10 with 40mx20m dimensions. In these experiments, the algorithm was tested in dynamic environments where multiple instances of redetection scenarios were studied.

Figure 5.10: Football court at IST Lisbon where the real-world tests were conducted.

The second batch of the real-world experimentation aimed at testing the full system in a longer target following and open-field experiments, complete with partial and full occlusions. For this purpose, the experiments where conducted at the Portuguese Navy Operational Experimentation Center (CEOM) in Tróia[4] shown in Figure 5.11, an experimental and investigation section of the Portuguese navy aimed at developing innovating unmanned technologies. In this venue, it was possible to test the system in longer tracking and following scenarios and obtain valuable data to evaluate target following performance as well as the ability to keep the target in line-of-sight. In total, five tests were conducted: two in the runway (marked 1 in Figure 5.11) using BoT-SORT and ByteTrack and two more with both trackers in the Heliport marked as 2. The final test was also conducted in the runway, in strong wind conditions above 10 m/s to test the system in limit conditions. Video footage from the experiments can be seen in the footnotes[5].

5.3.1 Experimental setup and Platform

The UAV platform, shown in Figure 5.12 was assembled from scratch based on the DJI-F550 Hexacopter frame. For the on-board computer, a NVIDIA Jetson Xavier NX Developer Kit was used, which is capable of high-power performance on a portable light-weight system. The FCU of choice was the Pixhawk 2.1. Cube Orange, which is connected to the NVIDIA Jetson via the telemetry port TELEM2 to transmit control commands. Due to some know issues with the telemetry connection between the

[3]Google maps IST field location: `https://maps.app.goo.gl/RhEaq1UJcRoMFrTT9`
[4]CEOM Google maps location: `https://maps.app.goo.gl/5WTGRWUyhjNFL7KMA`
[5]Real-world experiments video: `https://youtu.be/eQuAWoovpI8`

Figure 5.11: Portuguese Navy Operational Experimentation Center (CEOM) at Tróia. Runway marked with number 1 and Heliport marked with number 2.

NVIDIA Jetson and the Pixhawk 2.1. Cube Orange [63, 64], instead of using the TX, RX ports, a USB to TTL Serial, HW-597 converter was introduced to allow communications. The MAVROS node is launched through this connection using the /dev/ttyUSB0 port with a baud-rate of 921600. In addition, the FCU is connected to a telemetry radio module via the TELEM1 telemetry port. This module operates under the MAVLINK protocol with a frequency of 433MHz and up to 500mw. It is responsible for handling all communications between the FCU and the Ground Station during flight operations. For the GPS, a $here^{+}$ HEX GPS was used on the flight platform with an $here^{+}$ RTK GPS support in the Ground Station. A scheme of the several connections and parts of the system is shown in Figure 5.13. For the camera sensor view and secure landing, a carbon fiber landing gear was installed that allows for a clear unobstructed view from the Realsense D435 RGB-D camera, installed at a 30 degree angle from the horizontal position as in the simulation, shown in Figure 5.12(b). The realsense camera resolution in the real-world was set as 640x480 at 30FPS, for both RGB and depth. The depth images are aligned in real-time with the RGB images to allow accurate matching between the YOLOv8 detections and depth values.

The desired distance $d_{desired}$ was set to 8 meters considering the measured accuracy of the realsense in the real-world as shown in Figure 5.14. The accuracy was measured based on a set of real-world experiments, by comparing the realsense measured depth distances with the real distance measurements between the camera and a blank wall. It can be seen that the accuracy decreases linearly with the distance from the camera with the 20% error line chosen as the threshold for depth use, falling around the 8-meter mark.

5.3.2 Redetection Results

To properly test the redetection capabilities of the proposed system, several real-world scenarios where considered where the target would go through all the redetection modes: Target ID Change,

(a) UAV platform during flight tests.

(b) Camera installation angle.

Figure 5.12: UAV platform developed for real-world tests.

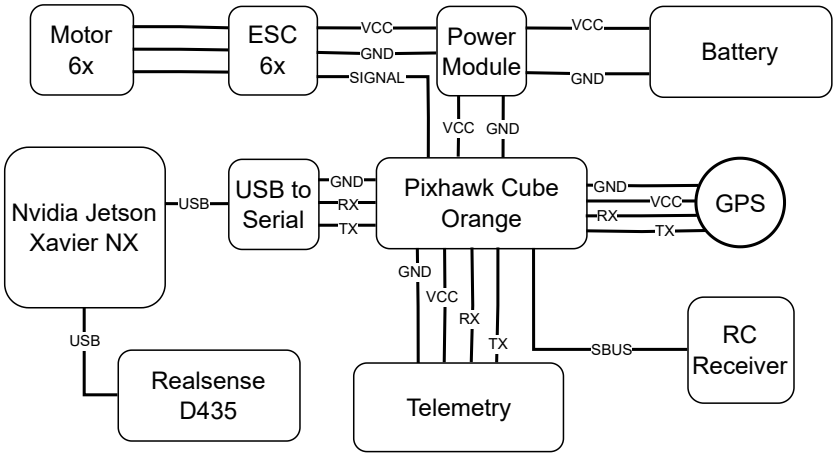


Figure 5.13: Developed UAV platform System Scheme.

Target Missing and Target Lost modes.

First, the effectiveness of the Target ID Change redetection mode was evaluated. As mentioned in Chapter 4, an ID change is a phenomenon in target tracking where the target is assigned a different identity by the MOT, in consecutive frames. This is more frequent in dynamic environments and can be exaggerated due to sudden camera movements. Hence, the Target ID Change redetection mode is vital step to make the system able to track the target in dynamic environments by overwriting the locally defined target ID with the newly assigned ID from the MOT algorithm, keeping accurate target tracking and following, despite MOT limitations. Figure 5.15 shows an example of several consecutive Target ID Change redetections where the target is successfully redetected without interference in target following. These results demonstrate that the system is robust to ID Changes, addressing one of the biggest limitations of MOT methods in these applications.

Second, the Target Missing mode was assessed through scenarios with temporary occlusions, since this mode serves as an intermediary phase between the initial detection failure of the target and its categorization as lost. During this phase, an estimation of the position of the target is pursued to navigate

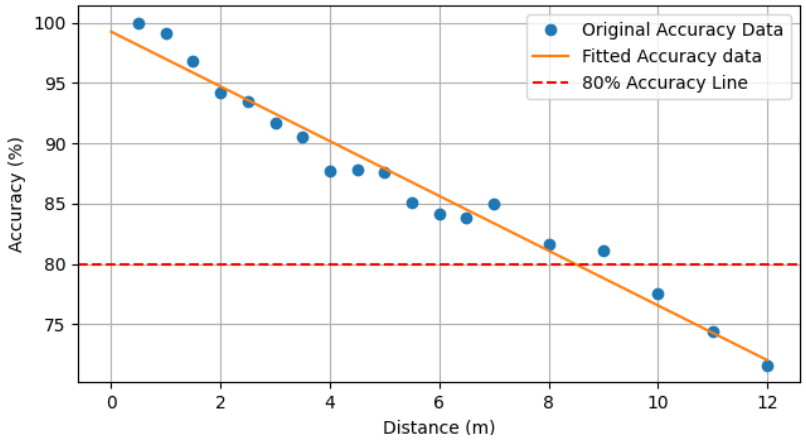
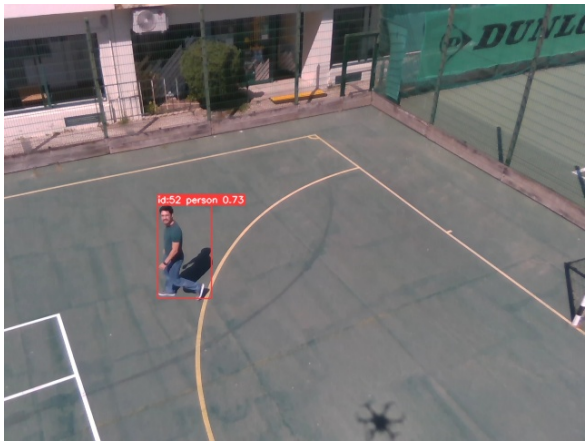


Figure 5.14: Realsense Depth Accuracy over distance.

(a) Target followed with ID 51.



(b) Target ID Change - overwrite local target ID from 51 to 52.

(c) Target ID Change - overwrite local target ID from 52 to 53.

(d) Target ID Change - overwrite local target ID from 53 to 54.

Figure 5.15: Target ID change redetection experiment: system recognizes and adapts to ID changes from the MOT module by updating the locally defined target ID.

temporary occlusions without completely stopping the UAV. Figure 5.16 shows an instance where the target was obscured as it crossed behind a bystander. Results demonstrate a continuous target following behaviour even during the occlusion period by utilizing the Kalman Filter predictions during the Target Missing mode.

(a) Following the target (left). Bystander in the right.

(b) Target crosses behind the bystander. Initiate Target Missing mode.

(c) Target Missing - following Kalman Filter estimate.

(d) Target (right) redetected and followed.

(e) Target (right) followed.

(f) Target (right) followed.

Figure 5.16: Target Missing redetection experiment: the target is redetected after temporary occlusion when crossing behind a bystander.

To test more complex redetection scenarios with full occlusions, three progressively more challenging scenarios where considered. Firstly, the target gets behind a non-person static obstacle in the terrain. In this scenario, the capabilities of redetection are tested on a basic level without any interference from bystanders. Figure 5.17 shows the sequential steps during the first experiment. The system tracks the target in the Following Mode for Figure 5.17(a). During Target Missing Mode (from Figure 5.17(b) until Figure 5.17(c)) the UAV attempts to follow the missing target by using the Kalman Filter estimate. After some consecutive frames without redetecting the target, the system will switch to the Target Lost mode where all movement is stopped (Figure 5.17(c)). Once the target is redetected (Figure 5.17(d)) the System Mode Switcher will activate the Adjusting Mode to center the target in the image (Figure 5.17(e)) which is then followed by the Following Mode (Figure 5.17(f)).

In the second scenario, the target walks alongside another person and then hides behind that bystander for a brief amount of time. Here the redetection algorithm is challenged in its ability to differentiate between a bystander and the target in a redetection scenario.

In the third experiment, the ability to handle dynamic scenes with multiple people is tested by having the target and two more bystanders walking randomly in the same area. The target then hides behind one of the bystanders in the center while the other keeps moving around in the image (Figure 5.19(c)). It can be seen that the system ignores the bystanders and correctly waits for an accurate redetection

(a) Target Following Mode ID 7. (b) Target hidden behind bystander. (c) Target Lost. Stop movement.

(d) Target reappears. (e) Target redetected ID 16. Adjusting Mode. (f) Target redetected ID 16. Following Mode.

Figure 5.17: Target Lost Experiment 1 - Obstacle occlusion redetection scenario. Target successful redetection after full occlusion behind a non-identifiable object.

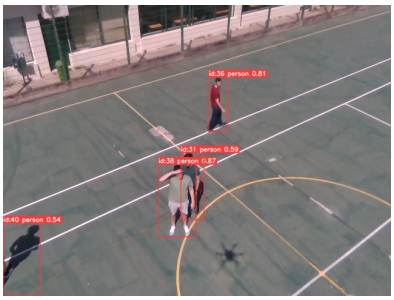
(a) Following the target (left). Bystander in the right. (b) Target hidden behind bystander. Target Missing Mode. (c) Target Lost mode.

(d) Target reappears in the detections. (e) Target Adjusting Mode. (f) Target Following Mode.

Figure 5.18: Target Lost Experiment 2 - Two person redetection scenario. Target successful redetection after full occlusion behind bystander.

candidate while the target is hidden. Once the target is redetected in the image (Figure 5.19(d)), it is correctly identified and followed (Figure 5.19(e) and Figure 5.19(f)).

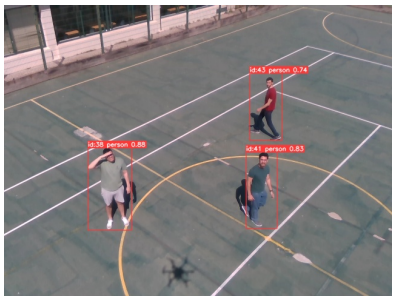
(a) Following target green ID-31. Bystanders: grey ID-38 and red ID-36.



(b) Following the target (green ID-31).

(c) Target hides behind bystander ID-38 (grey). Bystander ID-36 (red) moving.

(d) Target (green ID-41) reappears in the detections.



(e) Target Adjusting Mode.

(f) Target Following Mode.

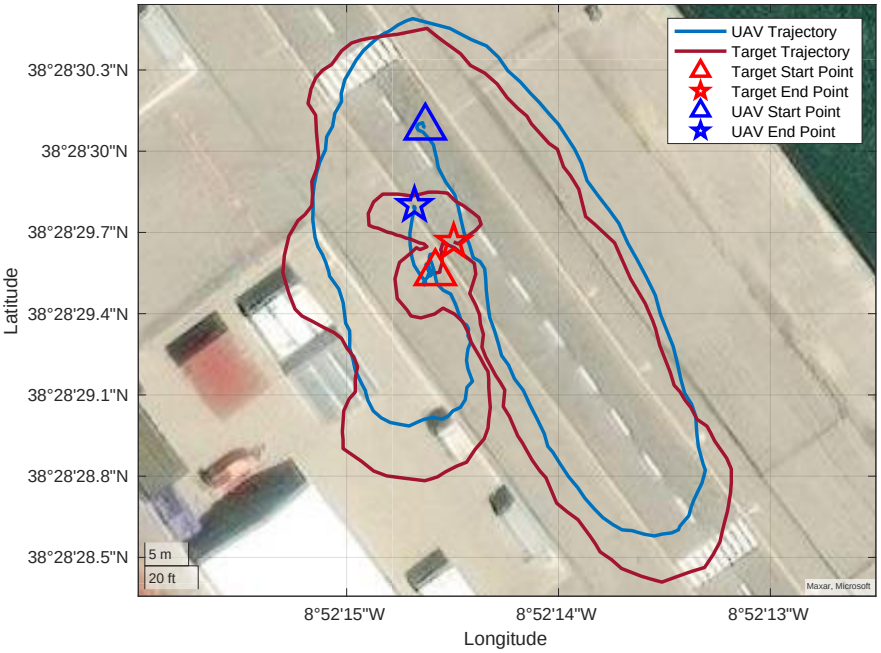
Figure 5.19: Target Lost Experiment 3 - Dynamic scenario with multiple people. Target successful redetection after full occlusion behind bystander while another person walks around in the image.

5.3.3 Target tracking and following

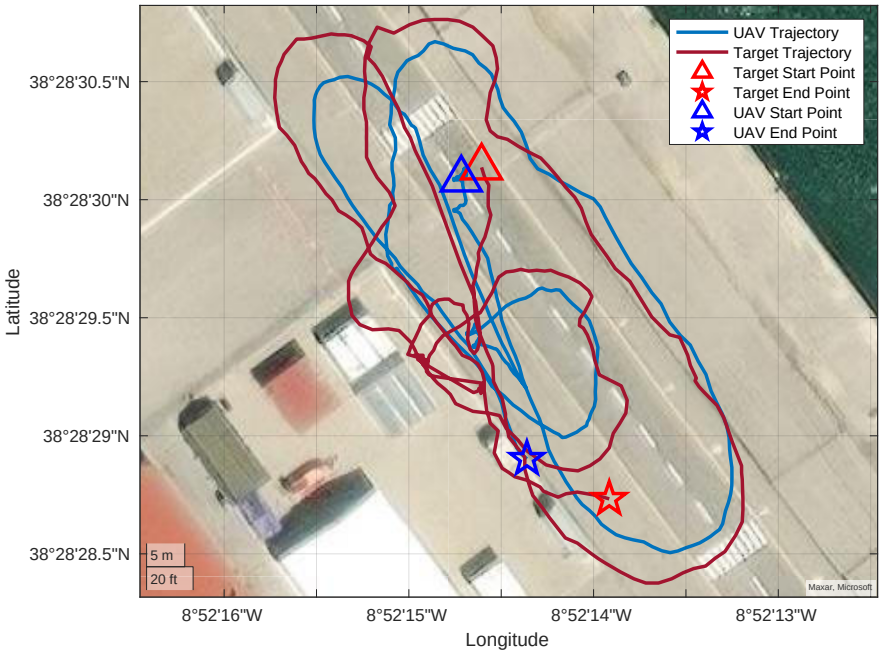
In this subsection, the target tracking and following capabilities were evaluated for both the BoT-SORT and the ByteTrack trackers during five long term tracking and following experiments in CEOM (Figure 5.11). During the experiments, the target also underwent through several different occlusion scenarios such as hiding behind other people, thereby also testing the redetection capabilities of the system.

The trajectories of the experiments conducted in the runway are illustrated in Figure 5.20 and the trajectories for the tests in the heliport are shown in Figure 5.21. Results for the experiments are shown in Table 5.4. For the real-world experiments the same overall metrics of performance evaluation are used as in the simulation experiments, with the exception of the estimation accuracy which is not available in the real-world due to the lack of accurate ground truth positioning caused by the accumulation of GPS deviations in the UAV and GPS deviations in the target's equipment used to map the trajectory.

Regarding UAV and Target distances, the same pattern as the simulation results can be seen with the UAV traveling slightly less distance than the target, totaling 1233m for the UAV distance and 1370m for the target due to the rotation capabilities of the UAV. Depth Use values are considerably lower than their simulation counterpart, which can be attributed to the reduced performance of the realsense D435 camera in the real-world for distances greater than 8 meters. Considering the reduced field-of-view of



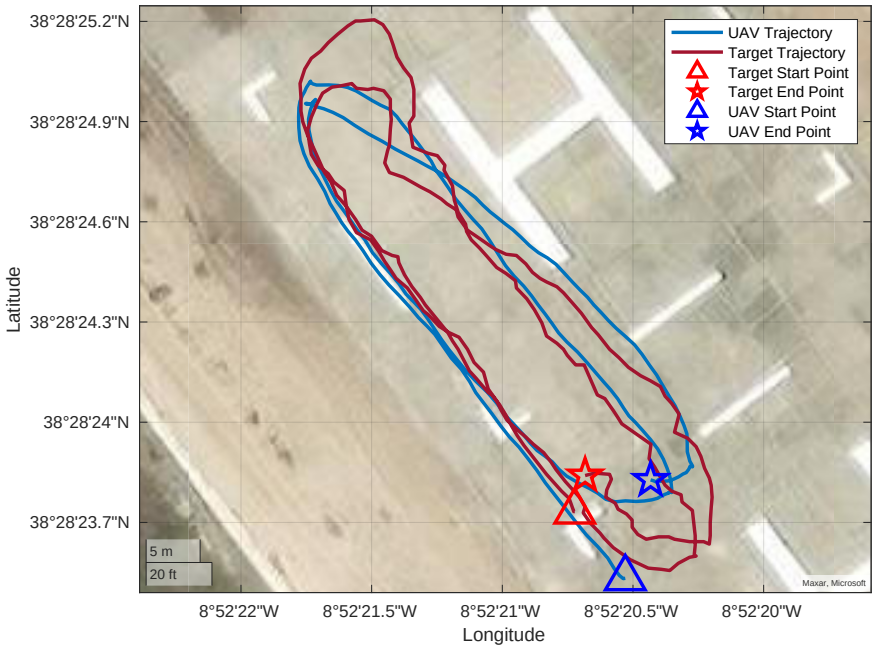
(a) BoT-SORT Experiment.



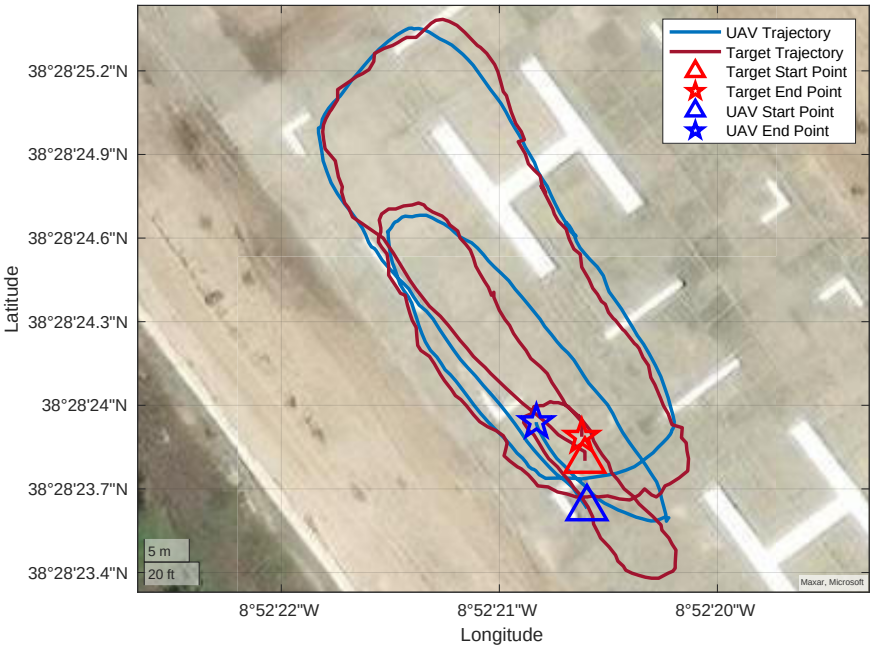
(b) ByteTrack Experiment.

Figure 5.20: UAV and target trajectories during the runway experiments. The target walks randomly at average walking speed with some moments of higher speed variations (e.g. running). Moments of full occlusion are performed with the target hidden behind bystanders.

the camera in the real-world and the reduced resolution from 1280x720 to 640x480, it is unviable to keep the drone at smaller distances from the target which hinders the Depth Use performance. However, this method still proves very effective in preventing the UAV from overshooting the position of the target



(a) BoT-SORT Experiment.



(b) ByteTrack Experiment.

Figure 5.21: UAV and target trajectories during the heliport experiments. The target performs clockwise laps at an average walking speed without speed variations and without major occlusions.

in the event of partial occlusions where the bounding box information is not reliable. Considering this safeguard feature, it can be considered that the use of depth information is a vital cornerstone for the reliability and robustness of the system, even if it is only used between 9.60% and 15.63% of the time.

Regarding Target ID Changes, it is still very noticeable the limitations in the ByteTrack tracker compared to the BoT-SORT tracker, that has substantial improvements. In addition, in the real-world sce-

Table 5.4: Data from the real-world experiments conducted in CEOM.

Experiment Scene	Tracker Used	Time (s)	Distance UAV (m)	Distance Target (m)	Visual Accuracy (%)	Depth Use (%)	Frame Rate (Hz)	No. ID Change
Runway	BoT-SORT	274	308	340	92.31	15.63	6.96	13
Runway	ByteTrack	376	439	510	90.27	10.35	8.05	77
Heliport	BoT-SORT	252	228	250	99.69	10.83	7.62	3
Heliport	ByteTrack	320	258	270	88.12	9.60	8.92	118

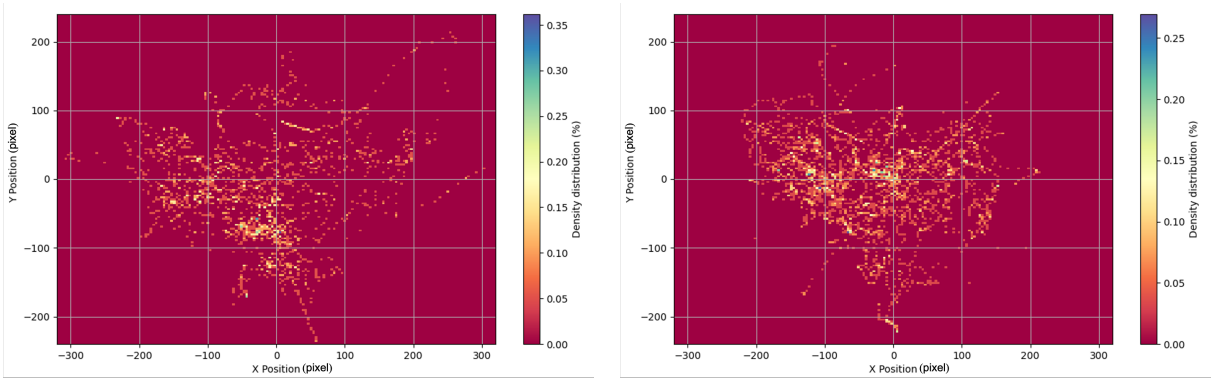
nario despite the ByteTrack experiments still achieving higher frame rate, the difference is much less pronounced which makes a better case for the trade-off between effectiveness and computational load that the BoT-SORT tracker offers. Overall, in terms of computational capabilities, the system still obtains frame rates capable of sustaining real-world performance, although it is noticeable a decrease in efficiency from the simulation experiments, mainly due to the lower level of processing power available on-board. Visual Accuracy is within expected levels, comparing with simulations results and considering the partial and full occlusions present. The decrease in visual accuracy for the ByteTrack tracker can be justified by the increased amount of ID Changes that are usually associated with some frames without detections or missed tracks.

To further evaluate the capabilities of the system at maintaining line-of-sight, the heatmaps representing the position of the center of the bounding box of the target for the four experiments are shown in Figure 5.22, with the average and standard deviations shown in Tab. 5.5. Firstly, a distinction is observed between the runway and heliport experiments, where the latter shows overall better results in maintaining line-of-sight. This difference can be attributed to the nature of the experiments conducted: while in the heliport the target followed a circular trajectory with little speed variations and direction changes, in the runway, the trajectory was more erratic with speed and direction changes (walking, jogging and even running as shown in Figure 5.23). With this in consideration, it can be seen that there are no noticeable differences between both trackers. In the heliport experiments there is a visible shift in the density distribution to the right, caused by the clockwise direction of the movement in those experiments while in the runway the distribution is more evenly spread. Hence, it can be determined that the horizontal deviations are mainly caused by the sideways movement of the target in relation to the UAV when turning.

Table 5.5: Average and standard deviation of the target pixel positions during the experiments.

Experiment	X error (pixel)	Y error (pixel)	Total error (pixel)
Runway w/ BoT-SORT	76.47 ± 96.75	61.38 ± 74.63	68.92 ± 56.58
Runway w/ ByteTrack	76.86 ± 88.27	47.11 ± 62.62	61.98 ± 50.26
Heliport w/ BoT-SORT	63.21 ± 68.80	35.06 ± 47.54	49.14 ± 48.30
Heliport w/ ByteTrack	61.18 ± 63.54	42.63 ± 56.33	51.91 ± 45.02

Investigating this correlation further, as depicted in Figure 5.24, a slight delay is observed between the yaw rate reference provided to the FCU by the yaw rate controller developed in this thesis (shown in blue) and the actual actuation and consequent yaw rate produced by the UAV (shown in orange). This



(a) Runway w/ BoT-SORT tracking tests.

(b) Runway w/ ByteTrack tracking tests.

(c) Heliport w/ BoT-SORT tracking tests.

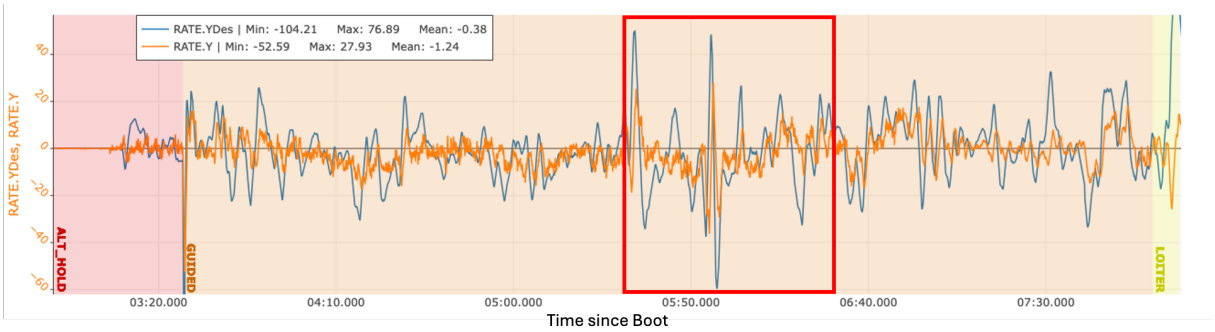
(d) Heliport w/ ByteTrack tracking tests.

Figure 5.22: Heatmap with the probability distribution of the center of the target in the image frame for the four experiments.

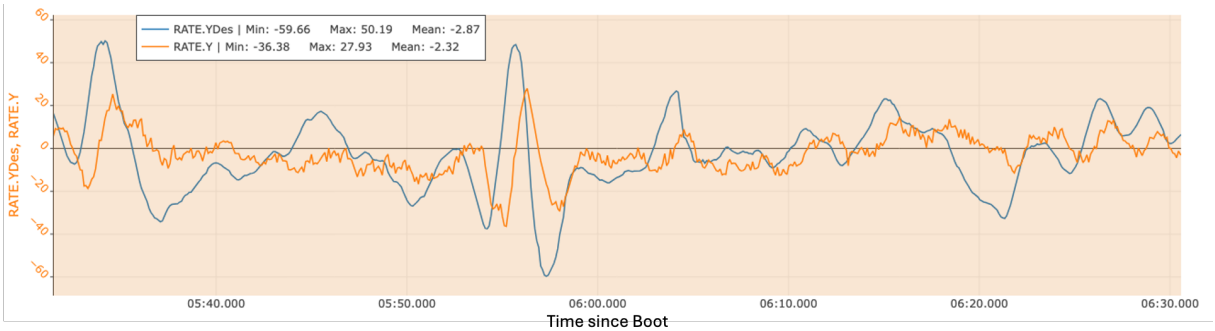
Figure 5.23: Running test during the ByteTrack experiment in the runway where the target quickly accelerates away from the UAV. UAV on the top left corner and target in the bottom right.

delay signifies the limitations of the FCU autopilot, which can be mitigated by fine-tuning the autopilot's parameters and filters. In this work, a standard approach was adopted where only the essential parameters were adjusted to ensure stable and controlled flight. These parameters include the Stabilize Roll/Pitch and Rate Roll/Pitch parameters that control the roll and pitch response, and the Stabilize Yaw

and Rate Yaw parameters that control the yaw response. However, it's important to note that this is a baseline tuning and there's room for further optimization. A comprehensive tuning process, involving all parameters and filters, could potentially yield better performance. This process, however, can be quite lengthy and requires a significant amount of intuition and expertise. It would involve adjusting filters such as the Fast Fourier Transform Based Harmonic Notch Filter and the accelerometer filters for less noisy measurements which require extensive in-flight tuning. Given the scope of this thesis, a detailed tuning process was not pursued. However, future work could explore these additional tuning possibilities to further optimize the performance of the UAV.



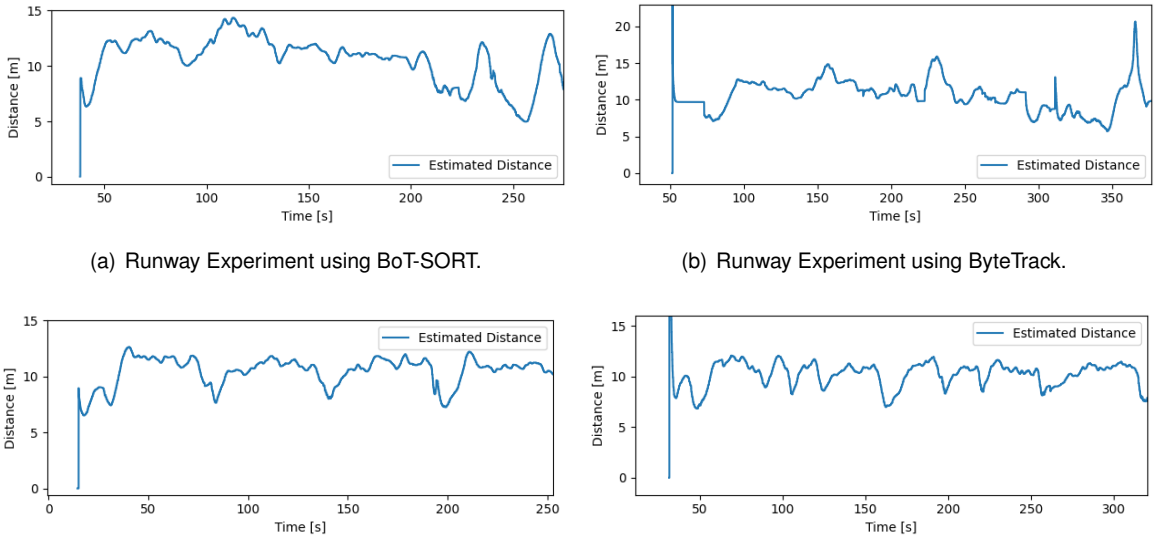
(a) Plot of the full experiment.



(b) Zoomed in section of the yaw rate plot.

Figure 5.24: Desired yaw rate (blue) and respective yaw rate actuated by the UAV (orange) over the course of the full BoT-SORT experiment in the runway during the Guided flight mode and including take-off and landing marked by the Alt-Hold and Loiter flight modes.

Distance estimation values from the four experiments are shown in Figure 5.25, with the respective average estimated distance to the target and standard deviations shown in Tab. 5.6. It can be seen that the target averages around 10 meters rather than the pretended 8 meters set as the desired distance, which explain the lower Depth Use compared to the simulation experiments. The main reason for this deviation might be the difference in the behavior of the target between the simulation and the real-world experiments. While in the simulation, the actors would often walk towards the UAV, in the real-world, the target often walks or even runs away from the UAV. This behavior makes it so the UAV is often catching up to the target giving higher average distances.



(a) Runway Experiment using BoT-SORT.

(b) Runway Experiment using ByteTrack.

(c) Heliport Experiment using BoT-SORT.

(d) Heliport Experiment using ByteTrack.

Figure 5.25: Estimated distance from the target over time during the CEOM experiments.

Table 5.6: Average and standard deviations of the estimated distance from the target over time during the CEOM experiments.

	Runway w/BoT-SORT	Runway w/ByteTrack	Heliport w/BoT-SORT	Heliport w/ ByteTrack
Average	10.28	10.77	10.32	9.95
Standard Deviation	2.153	1.952	1.278	1.630

5.3.4 Limit Testing

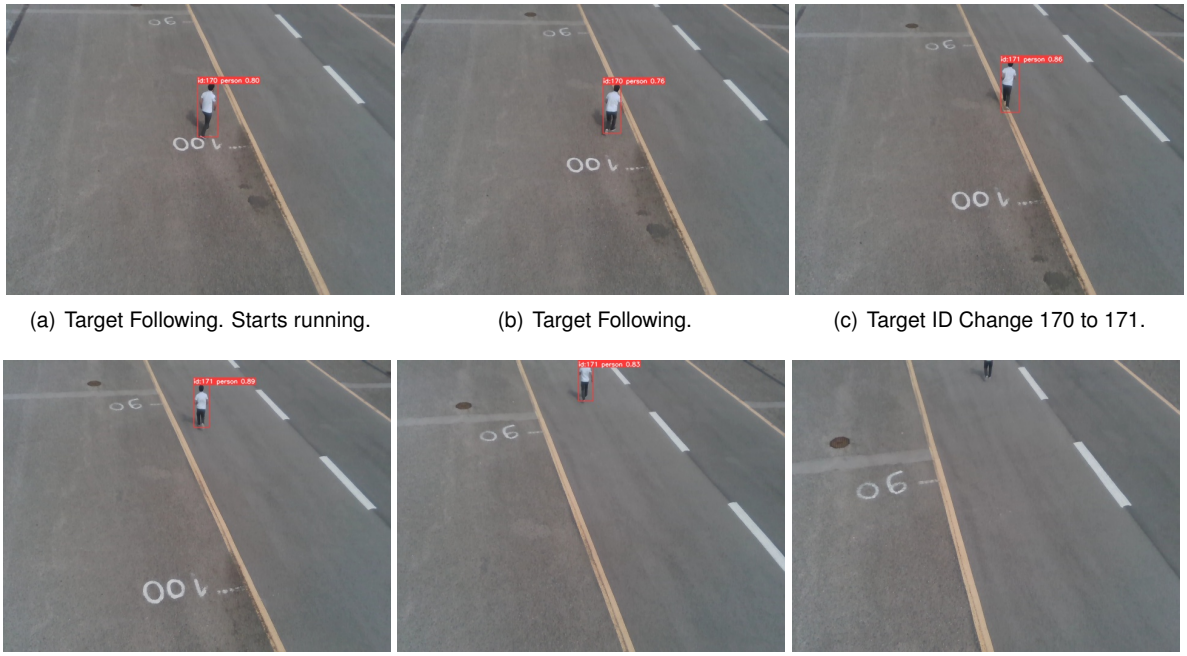
Limit testing aims to evaluate the performance and robustness of the proposed system under extreme or challenging conditions. The requirements set at the beginning of the experiments provide a framework for these tests:

- Follow a moving target at a regular walking speed (1 to 1.5 m/s) with small accelerations.
- Maintain robustness in conditions where the target is surrounded by bystanders and can experience occlusions, such as hiding behind bystanders (as discussed in Section 5.3.2).
- Operate within reasonable wind conditions (less than 7 m/s).

The first condition was tested during the two runway experiments, where the system followed a moving target at the specified speed. Target redetection in dynamic environments was thoroughly examined in Section 5.3.2. The wind condition was later tested in a third experiment on the CEOM runway, where the system operated under heavy winds up to 10 m/s. These tests collectively assess the system's ability to perform reliably in various challenging scenarios.

Speed change test

One of the initial parameters set by the user for the developed system is the desired distance to the target as mentioned in Section 4.4. For the developed experiments, this parameter was set considering an average walking speed between $(1 \sim 1.5m/s)$, which poses a natural difficulty in target following if the target suddenly starts running as shown in Figure 5.23. The major limitation in this scenario is the potential for the target to get out of frame. However, the developed system aims at providing a robust solution to many potential situations including speed changes. Hence, in order to handle it, the system takes advantage of the Target Missing mode in which it continues to follow an estimate of the target position for a brief period before stopping all movement and assuming the Target Lost mode. In Figure 5.26, the sequence of captured images from the movement of Figure 5.23 is exhibited from the perspective of the UAV until the target enters the Target Missing mode. Figure 5.27 illustrates the target recovery and redetection process, where the UAV continues to move forward chasing an estimate of the target in the Target Missing mode, successfully redetects the target and recovers optimal positioning by centering it in the image.



(a) Target Following. Starts running.

(b) Target Following.

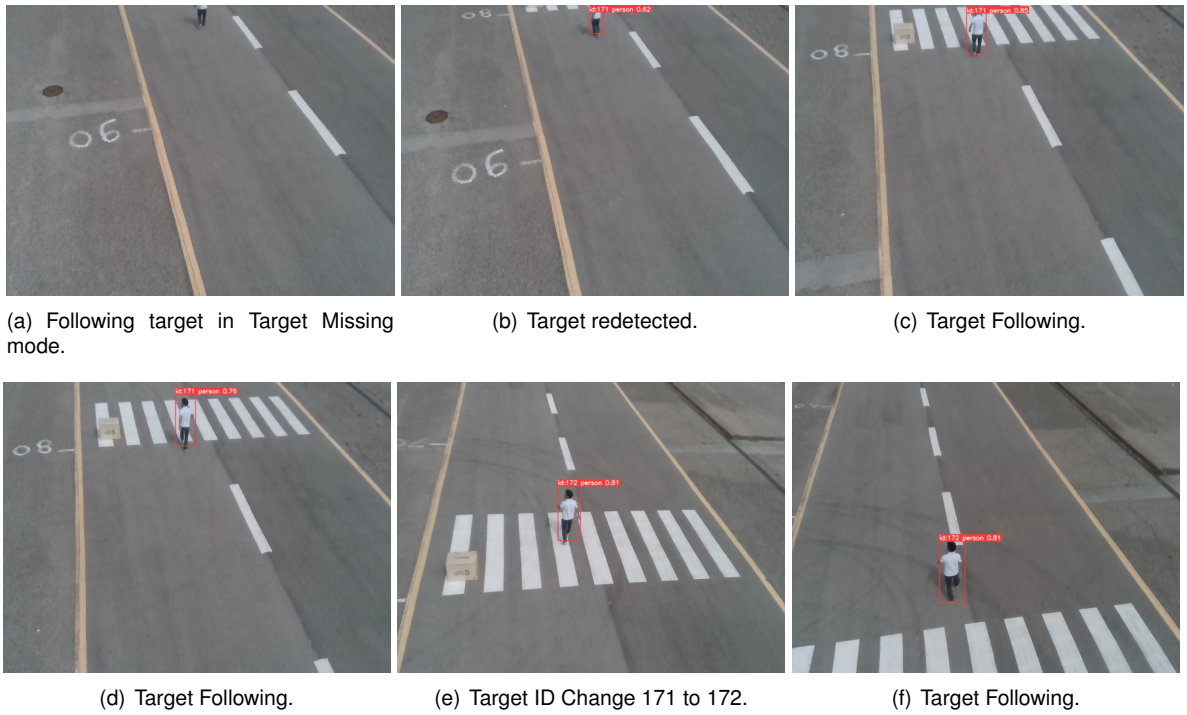
(c) Target ID Change 170 to 171.

(d) Target Following.

(e) Target Following. Partially occluded.

(f) Target not detected. Initiate Target Missing mode.

Figure 5.26: Limit testing - target runs away from the UAV.



(a) Following target in Target Missing mode.

(b) Target redetected.

(c) Target Following.

(d) Target Following.

(e) Target ID Change 171 to 172.

(f) Target Following.

Figure 5.27: Limit testing - redetection of the target using the Target Missing mode.

Heavy wind test

It is always preferential to avoid heavy winds when dealing with UAV operations due to the inherent perturbations associated with them. These perturbations can induce unforeseen problems with the system that can be addressed in order to increase the range of applications. Hence, this final test performed in the runway in heavy wind conditions fulfilled this purpose of determining the consequences of intense external perturbations. By analysing the images taken from the on-board camera and the roll values, the major difficulties that arise from these abnormal conditions can be summed up by an increase in the roll angle of the UAV in an attempt to achieve a stabilized flight. The roll angle from the first BoT-SORT experiment in the runway averages around 1.9416° while the heavy wind experiments averages around 5.443° . Given the lack of roll stabilization from the camera (that could, for example, be provided by a gimbal system), the external perturbations have a direct effect on the bounding boxes that surround the target as seen in Figure 5.28, which is also mentioned by De Smedt et al. [65]. Taking into consideration that the mainly used distance estimation method depends on the size of the bounding box, when subject to heavy wind conditions, the roll angle of camera will directly influence the height of the bounding box and consequently the estimated distance to the target when the depth information is not available. Therefore, if the application of the system would: require operating in heavy wind conditions, need precise target distance estimation precision, and there is limited or unavailable depth information; a roll correction algorithm [65] might be worth to consider in future works, which would provide better distance estimation values for some extra processing time without the need to incorporate extra camera stabilization hardware.

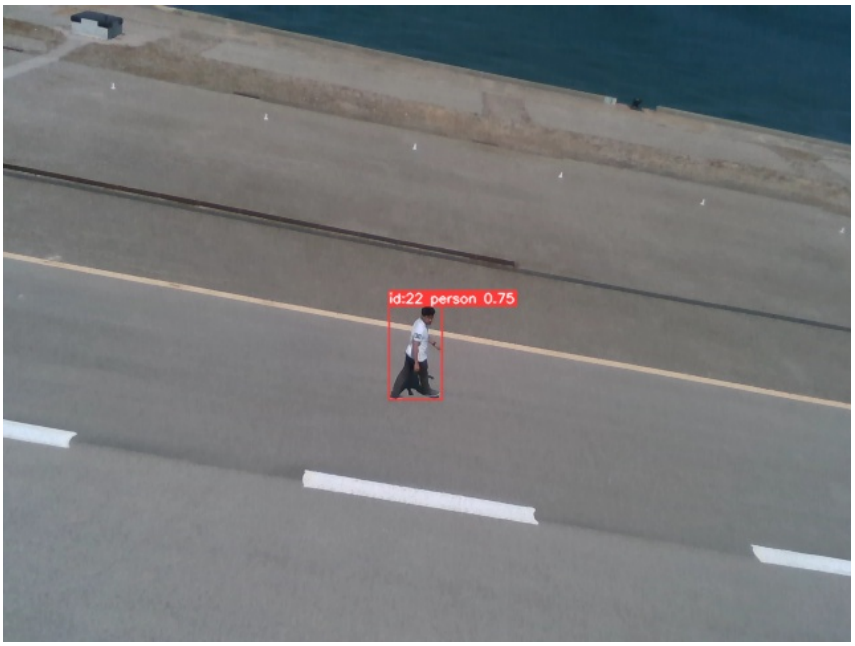


Figure 5.28: Example of the bounding box rotation due to higher roll angle caused by wind compensation.

Chapter 6

Conclusions

The final chapter wraps up the work developed in this thesis. It starts by reviewing the research problem and the objectives of the thesis and how they were initially addressed. Then, the key findings are summed up as well as some conclusions about the obtained results and the implications they pose for this research field. Limitations of the system are reviewed alongside some future improvement suggestions and other lines of work that can arise from applying this newly developed system in different scenarios.

6.1 Achievements

This thesis investigates the possibility of introducing the current state-of-the-art MOT algorithms in vision-based target tracking and following from an UAV. The major challenges related to this topic are the computational limitations imposed by the added complexity of tracking multi-target information and the limitations imposed by these methods regarding ID changes in moving cameras, caused by the difficulty in maintaining IDs during fast camera movements. The developed system takes on these challenges by using the most recent object detection methods such as YOLOv8 and fast embedded systems such as the NVIDIA Jetson Xavier NX to enable real-time processing speeds. Regarding ID changes, this work proposes a redetection algorithm capable of going above and beyond by both solving the ID change problems and enhancing the redetection capabilities even in dynamic scenarios with full occlusions. Furthermore, this thesis proposes a distance estimation method composed of two stages: utilizing visual bounding box information from afar and leveraging depth information once it becomes available. The developed controller follows an approach similar to the PN method by centering the target in the image using both the yaw rate and the vertical velocity controllers and then following the target by moving forward or backwards. The system underwent extensive testing, first in a simulation environment and later in two real-world scenarios. The real-world UAV platform was also fully developed for the purpose of this thesis, which included: assembling the frame and onboard components, soldering all the connections, designing the system layout and setting up the correct connections to allow accurate communications between all systems.

Performance evaluation took into consideration two state-of-the-art MOT algorithms (BoT-SORT and ByteTrack) in order to have comparative data and investigate how much the recent camera motion model compensations from BoT-SORT would improve tracking results. As expected, the BoT-SORT algorithm outperforms ByteTrack in terms of tracking precision, however, it comes at the cost of having a lower frame rate. If one were to take the ID Changes results at face value, it would be unthinkable to even consider using ByteTrack for a target following application which might justify the lack of research into the use of MOT algorithms in target following scenarios. However, the developed redetection system of this thesis, namely the Target ID Change redetection method, makes it possible to overlook these limitations since they are corrected as soon as they appear. In addition, BoT-SORT camera correction models show great improvements in keeping the ID of the target constant over time, which makes it more suitable for these applications. Real-world results continue in the same trend, although the differences between both trackers become less pronounced in terms of computational efficiency. On the other hand, BoT-SORT remains almost unwavering in terms of ID changes compared to ByteTrack which makes BoT-SORT a preferable choice when weighing both metrics.

By overcoming the ID change challenge and implementing the three-step redetection process, the system is able to outperform existing methods by leveraging the multi-target information to be able to redetect the target even in dynamic scenarios where the target remains hidden while there are other bystanders in the image. The use of MOT removes the dependency on other redetection methods such as motion measurements between frames [41] or the use of YOLO without a tracker [37] which are not able to handle multiple pedestrians moving in the frame. Results show that the system is capable of handling complex and dynamic scenarios where partial or full occlusions are common, from simple scenarios such as hiding behind an object, to a more complex scene where it hides behind a bystander while others walk in the frame.

Moreover, the 3D flight controller effectively follows the target while keeping it centered in the image showcasing robustness even during sudden direction changes. The distance estimation method relying on both visual data and depth information performs a vital role in the system by first allowing target following as long as the target is detected by using the bounding box information that is always available. Secondly, the depth information on top of providing better estimation values when the UAV is close to the target, also prevents the UAV from overshooting the target, a situation that could arise from partial occlusions where the bounding box appears smaller than the full size of the target. The combination of both estimation methods makes for a robust and versatile system designed to work in a wide range of applications.

6.2 Limitations & Future Work

While this research has made significant contributions in advancing the field of vision-based target tracking and following by UAVs, it is important to acknowledge some of the limitations. Starting from the assumptions, although the proposed system is capable of tracking and following any target that can be identified using YOLO, this work was developed with the goal of tracking and following walking

pedestrians. With this in mind, in order to fully generalize the system for any target, new assumptions have to be made regarding the expected target size, speed changes and direction changes. Hence, one of the limitations includes the maximum acceleration of the target and direction changes which may cause loss of line-of-sight if there is a sudden and fast movements. One such case was tested in the ”limit testing” subsection 5.3.4 where the target suddenly accelerated leaving the field of view of the UAV. In the tested scenario, the system was able to successfully redetect the target, however, if the same movement were to be done in such a way that the target would run towards and under the UAV, it would prove difficult to perform a similar redetection. In cases where such scenarios are expected, it would be advisable to set a higher value for the desired distance to increase the field-of-view of the camera. This would however fall onto the second limitation of the system - the depth sensor's range. This limitation is imposed by the available hardware which can only accurately retrieve depth information when close to the target. Regarding the ability to maintain accurate target centering, limitations arise due to the lack of camera stabilization since the pitch and roll of the UAV are coupled with the pitch and roll of the camera. This can influence the position of the target in the vertical axis of the image when moving forward or backwards and in the horizontal axis when rolling (which may increase in heavy wind conditions as shown in the limit testing subsection 5.3.4).

Future works may consider the integration of a gimbal which would decouple the movement of the UAV from the movement of the camera and obtain better results. In addition, further tunning of the autopilot parameters and filters could be done to compensate for the delay between the yaw rate reference and the controller output. Finally, a non-linear controller and a path planner could also be considered to both improve target following and provide an alternative view point for the UAV in cases where the target is concealed. In terms of applications, it would pose an interesting challenge to apply the developed system to other target following scenarios such as vehicles, ships or bicycles to experiment with different target sizes and speeds. Also, the ability to leverage multi-target information opens up new possible avenues of research such as conditional target switching based on the detected behavior of the targets or even multi-target following by keeping more than one target in line-of-sight.

In summary, this thesis has successfully validated the use of state-of-the-art MOT algorithms in the context of vision-based target following by UAVs, addressing key challenges such as computational limitations and ID changes in dynamic environments. The system takes advantage of multi-target information to improve upon existing redetection algorithms increasing the overall reliability in complex scenarios. In overcoming the presented obstacles, this thesis lays a solid foundation for future research and enables new practical applications in the field of mobile target following from UAVs by leveraging multi-target information.

Bibliography

[1] J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han. Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake. *Journal of Field Robotics*, 33(3):290–321, 2016. doi: https://doi.org/10.1002/rob.21615. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21615.

[2] A. Prabhakaran and R. Sharma. Autonomous intelligent uav system for criminal pursuit -a proof of concept. *The Indian Police Journal*, 68:1–20, 06 2021.

[3] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer. *Autonomous Drone Cinematographer: Using Artistic Principles to Create Smooth, Safe, Occlusion-Free Trajectories for Aerial Filming*, pages 119–129. 01 2020. ISBN 978-3-030-33949-4. doi: 10.1007/978-3-030-33950-0_11.

[4] Y. Jie, L. Leonidas, F. Mumtaz, and M. Ali. Ship detection and tracking in inland waterways using improved yolov3 and deep sort. *Symmetry*, 13(2), 2021. ISSN 2073-8994. doi: 10.3390/sym13020308. URL https://www.mdpi.com/2073-8994/13/2/308.

[5] X. Liu and Z. Zhang. A vision-based target detection, tracking, and positioning algorithm for unmanned aerial vehicle. *Wireless Communications and Mobile Computing*, 2021:5565589, 2021. ISSN 1530-8669. doi: 10.1155/2021/5565589. URL https://doi.org/10.1155/2021/5565589.

[6] S. Liu, X. Li, H. Lu, and Y. He. Multi-object tracking meets moving uav. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8866–8875, 2022. doi: 10.1109/CVPR52688.2022.00867.

[7] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking, 2022.

[8] P. Nousi, I. Mademlis, I. Karakostas, A. Tefas, and I. Pitas. Embedded uav real-time visual object detection and tracking. In *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 708–713, 2019. doi: 10.1109/RCAR47638.2019.9043931.

[9] M. Hussain. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, 11(7), 2023. ISSN 2075-1702. doi: 10.3390/machines11070677. URL https://www.mdpi.com/2075-1702/11/7/677.

[10] X. Liu, Y. Yang, C. Ma, J. Li, and S. Zhang. Real-time visual tracking of moving targets using a low-cost unmanned aerial vehicle with a 3-axis stabilized gimbal system. *Applied Sciences*, 10: 5064, 07 2020. doi: 10.3390/app10155064.

[11] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska. The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 102:26, 2021. ISSN 1573-0409. doi: 10.1007/s10846-021-01383-5.

[12] D. Ferreira and M. Basiri. Leveraging multi-object tracking in vision-based target following for unmanned aerial vehicles. In *2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 88–93, 2024. doi: 10.1109/ICARSC61747.2024.10535936.

[13] J. Kaur and W. Singh. Tools, techniques, datasets and application areas for object detection in an image: a review. *Multimedia Tools and Applications*, 81:1–55, 04 2022. doi: 10.1007/s11042-022-13153-y.

[14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177.

[15] T. Lindeberg. Scale Invariant Feature Transform. *Scholarpedia*, 7(5):10491, 2012. doi: 10.4249/scholarpedia.10491. revision #153939.

[16] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001. doi: 10.1109/CVPR.2001.990517.

[17] W. Pitts and W. S. McCulloch. How we know universals the perception of auditory and visual forms. *The bulletin of mathematical biophysics*, 9(3):127–147, Sept. 1947. ISSN 1522-9602. doi: 10.1007/BF02478291. URL `https://doi.org/10.1007/BF02478291`. doi:10.1007/BF02478291.

[18] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[20] A. Lohia, K. D. Kadam, R. R. Joshi, A. M. Bongale, K. Dhananjay, and R. Raghvendra. Bibliometric analysis of one-stage and two-stage object bibliometric analysis of one-stage and two-stage object detection detection, 2021. URL `https://digitalcommons.unl.edu/libphilprac`.

[21] P. Soviany and R. T. Ionescu. Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. In *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 209–214, 2018. doi: 10.1109/SYNASC.2018.00041.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

[23] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd : Deconvolutional single shot detector, 2017.

[24] X. Cheng and J. Yu. Retinanet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021. doi: 10.1109/TIM.2020.3040485.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91.

[26] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, 13(1), 2021. ISSN 2072-4292. doi: 10.3390/rs13010089. URL https://www.mdpi.com/2072-4292/13/1/89.

[27] G. Jocher. YOLOv5 by Ultralytics, May 2020. URL https://github.com/ultralytics/yolov5.

[28] B. Aydin and S. Singha. Drone detection using yolov5. *Eng*, 4(1):416–433, 2023. ISSN 2673-4117. doi: 10.3390/eng4010025. URL https://www.mdpi.com/2673-4117/4/1/25.

[29] N. Al-Qubaydhi, A. Alenezi, T. Alanazi, A. Senyor, N. Alanezi, B. Alotaibi, M. Alotaibi, A. Razaque, A. A. Abdelhamid, and A. Alotaibi. Detection of unauthorized unmanned aerial vehicles using yolov5 and transfer learning. *Electronics*, 11(17), 2022. ISSN 2079-9292. doi: 10.3390/electronics11172669. URL https://www.mdpi.com/2079-9292/11/17/2669.

[30] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu. A forest fire detection system based on ensemble learning. *Forests*, 12(2), 2021. ISSN 1999-4907. doi: 10.3390/f12020217. URL https://www.mdpi.com/1999-4907/12/2/217.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL http://arxiv.org/abs/1912.01703.

[32] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[33] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng. Panet: Few-shot image semantic segmentation with prototype alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[34] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO, Jan. 2023. URL `https://github.com/ultralytics/ultralytics`.

[35] R. King. Brief summary of yolov8 model structure - issue 189. `https://github.com/ultralytics/ultralytics/issues/189`, 2023.

[36] J. Solawetz. "what is YOLOv8? the ultimate guide. [2024]". `https://blog.roboflow.com/whats-new-in-yolov8/`, Jan. 2023. Accessed: 2024-3-28.

[37] D. Luo, P. Shao, H. Xu, and L. Wang. Autonomous following algorithm for uav based on multi-scale kcf and kf. In *2023 4th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 430–436, 2023. doi: 10.1109/AINIT59027.2023.10212671.

[38] R. Mebarki, V. Lippiello, and B. Siciliano. Nonlinear visual control of unmanned aerial vehicles in gps-denied environments. *IEEE Transactions on Robotics*, 31(4):1004–1017, 2015. doi: 10.1109/TRO.2015.2451371.

[39] D. Zheng, H. Wang, J. Wang, X. Zhang, and W. Chen. Toward visibility guaranteed visual servoing control of quadrotor uavs. *IEEE/ASME Transactions on Mechatronics*, 24(3):1087–1095, 2019. doi: 10.1109/TMECH.2019.2906430.

[40] H. Bouzerzour, M. Guiatni, M. Hamerlain, and M. Boudali. Robust uncooperative ground target surveillance using vision-based sliding mode control of quadrotor uav. pages 1–8, 10 2022. doi: 10.1109/IECON49645.2022.9968329.

[41] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu. An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1732–1738, 2017. doi: 10.1109/IROS.2017.8205986.

[42] Y. Feng, D. Wang, and K. Yang. Research on target tracking algorithm of micro-uav based on monocular vision. *Journal of Robotics*, 2023:6657120, 2023. ISSN 1687-9600. doi: 10.1155/2023/6657120. URL `https://doi.org/10.1155/2023/6657120`.

[43] H. Wei. A uav target prediction and tracking method based on kcf and kalman filter hybrid algorithm. In *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pages 711–718, 2022. doi: 10.1109/ICCECE54139.2022.9712830.

[44] S. Yadav and S. Payandeh. Critical overview of visual tracking with kernel correlation filter, 12 2021. ISSN 22277080.

[45] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. doi: 10.1109/ICIP.2016.7533003.

[46] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017. doi: 10.1109/ICIP.2017.8296962.

[47] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, 2021. URL https://api.semanticscholar.org/CorpusID:238744032.

[48] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[49] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. 9 2016. URL http://arxiv.org/abs/1609.01775.

[50] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:246309, 2008. ISSN 1687-5281. doi: 10.1155/2008/246309. URL https://doi.org/10.1155/2008/246309.

[51] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixe, and B. Leibe. Hota: A higher order metric for evaluating multi-object tracking. 9 2020. doi: 10.1007/s11263-020-01375-2. URL http://arxiv.org/abs/2009.07736http://dx.doi.org/10.1007/s11263-020-01375-2.

[52] M. Nehru, J. Attzs, T. Mahendrakar, M. J. Meni, R. T. White, and I. Silver. Comparison of tracking-by-detection algorithms for real-time satellite component tracking. 2023. URL https://api.semanticscholar.org/CorpusID:268892020.

[53] T. Li, Z. Li, Y. Mu, and J. Su. Pedestrian multi-object tracking based on YOLOv7 and BoT-SORT. In L. Shen and G. Zhong, editors, *Third International Conference on Computer Vision and Pattern Analysis (ICCPA 2023)*, volume 12754, page 127541I. International Society for Optics and Photonics, SPIE, 2023. doi: 10.1117/12.2684256. URL https://doi.org/10.1117/12.2684256.

[54] S. Yan, Y. Fu, W. Zhang, W. Yang, R. Yu, and F. Zhang. Multi-target instance segmentation and tracking using yolov8 and bot-sort for video sar. In *2023 5th International Conference on Electronic Engineering and Informatics (EEI)*, pages 506–510, 2023. doi: 10.1109/EEI59236.2023.10212903.

[55] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL https://doi.org/10.1115/1.3662552.

[56] N. L. Baisa. Derivation of a constant velocity motion model for visual tracking, 2020.

[57] T. Roboflow. Coco 128 dataset. https://universe.roboflow.com/team-roboflow/coco-128, sep 2021. URL https://universe.roboflow.com/team-roboflow/coco-128. visited on 2024-05-28.

[58] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. volume 3, 01 2009.

[59] Realsense ros wiki, Feb 2024. URL https://wiki.ros.org/realsense_camera.

[60] K. Mori. ultralytics_ros, May 2023. URL https://github.com/Alpaca-zip/ultralytics_ros.

[61] Mavros ros wiki, Feb 2024. URL https://wiki.ros.org/mavros.

[62] G. ROS. Gazebo-ros actor wiki, Feb 2024. URL https://gazebosim.org/docs/harmonic/actors.

[63] Hyonlim. Jetson uart connection to telem1 or telem2, Aug 2021. URL https://discuss.cubepilot.org/t/jetson-uart-connection-to-telem1-or-telem2/6258/15.

[64] P. Dario. Cube orange communication error with jetson nano, Jul 2021. URL https://discuss.cubepilot.org/t/cube-orange-communication-error-with-jetson-nano/6867/5.

[65] F. De Smedt, D. Hulens, and T. Goedemé. On-board real-time tracking of pedestrians on a uav. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, 2015. doi: 10.1109/CVPRW.2015.7301359.