TÉCNICO
LISBOA

Detection and Tracking of Non-Cooperative UAVs: a deep learning moving-object tracking approach

João Paulo Dinis Lopes

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Afzal Suleman
Prof. Mário Alexandre Teles de Figueiredo

Examination Committee

Chairperson: Prof. Fernando José Parracho Lau
Supervisor: Prof. Afzal Suleman
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

December 2022

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 thank Professor Afzal Suleman for the amazing opportunity to live in Canada for half a year and for all the support and guidance throughout this past few months.

I would also like to thank Professor Mário Figueiredo for all his advice on the implementation of the algorithms and for his critiques to my work.

A special mention to Daniel Justino and Stephen Warwick for the useful and constructive discussions and I am grateful for all their help throughout this project.

I am also grateful to all my friends who came with me to Canada, especially João F., Inês and Carolina for all the time we spent together and the fun we had during this adventure.

Finally, a word of gratitude to my family including my dog, Milu. I wouldn't be here without you.

Lastly, I would like to thank my girlfriend, Sofia, for her visit to Canada, for all the moral support and for making my life more meaningful and happier.

Abstract

The recent proliferation of small Unmanned Aircraft Systems (sUAS) has brought with it serious safety and security concerns for both the civilian and military regulatory organizations. The detection and tracking of these aerial devices is necessary to implement Counter sUAS (C-sUAS) measures to mitigate the threats posed by these non-cooperative airborne systems.

Conventional visual object detection methods look at individual frames independently, ignoring temporal context (memory) information that successive frames might possess. For object tracking, there are two types of memory that should be considered to improve performance: memory of past observations and memory of past detections. Memory of past observations allows the algorithm to focus on the differences between the current and past observations, effectively making moving objects stand out against the background. Memory of past detections allows the algorithm to more easily identify true objects and their class, by having in mind the objects that previously occupied the same position.

This thesis presents a novel approach on C-sUAS operations, based on both types of memory, using consecutive frame subtraction and a new version of the ByteTrack tracker. The final system merges the state-of-the-art detector You-Only-Look-Once (YOLOv7) with the state-of-the-art tracker ByteTrack, with a new assignment metric, achieving an F1-score taking the track identification into account (referred to as IDF1) of 0.960, and a Multiple Object Tracking Accuracy (MOTA) of 0.991, both on a test subset created from the Anti-UAV dataset. All the computations were performed on a single Graphics Processing Unit (GPU) at a rate of 29.4 frames per second.

Keywords: Unmanned aircraft system, infrared imaging, object detection, YOLO, tracking

Resumo

A recente proliferação de pequenos veículos aéreos não tripulados trouxe consigo sérias preocupações a nível de proteção e segurança para as organizações reguladoras civis e militares. A deteção e rastreamento destes dispositivos é fundamental para o desenvolvimento de medidas capazes de enfrentar e mitigar as ameaças apresentadas por estes sistemas aéreos não cooperativos.

Os métodos convencionais de deteção visual de objetos analisam imagens individualmente e de forma independente, ignorando informação contextual que imagens sucessivas possam conter. Para o rastreamento de objetos, existem dois tipos de memória que devem ser considerados para um melhor desempenho: memória de observações passadas e memória de deteções passadas. A memória de observações passadas permite ao algoritmo concentrar-se nas diferenças entre as observações atuais e anteriores, destacando os objetos em movimento. A memória de deteções passadas permite ao algoritmo identificar mais facilmente os objetos e suas classes, tendo em conta os objetos que anteriormente ocupavam a mesma posição.

Esta tese apresenta uma nova abordagem, baseada em ambos os tipos de memória, usando subtração de imagens consecutivas e uma nova versão do algoritmo ByteTrack. O sistema final combina o detector *You-Only-Look-Once* (YOLOv7) com uma versão melhorada do algoritmo ByteTrack, para obter um *F1-score* tendo em conta os identificadores dos resultados, IDF1 de 0,960 e uma *Multiple Object Tracking Accuracy* (MOTA) de 0,991 num subconjunto de teste criado a partir do *Anti-UAV dataset*. Todos os cálculos foram realizados numa única *Graphics Processing Unit* (GPU) a uma taxa de 29,4 imagens por segundo.

Palavras-Chave: Veículo aéreo não tripulado, radiação infravermelha, detector de objectos, YOLO, rastreamento

Contents

List of Tables

List of Figures

List of Acronyms

AP	Average Precision.
CfAR	Center for Aerospace Research.
CNN	Convolutional Neural Network.
CSP	Cross Stage Partial.
Deep SORT	Simple Online and Realtime Tracking with a Deep association metric.
E-ELAN	Extended Efficient Layer Aggregation Network.
ECC	Enhanced Correlation Coefficient.
FAA	Federal Aviation Administration.
FN	False Negative.
FP	False Positive.
FPS	Frames Per Second.
GPU	Graphics Processing Unit.
ID	Identification.
IDF1	F1-score taking the track ID into account.
IDFN	ID False Negative.
IDFP	ID False Positive.
IDS	ID Switch.
IDTP	ID True Positive.
IoU	Intersection over Union.
IR	Infrared.
LiDAR	Light detection and ranging.
LSS-UAS	Low, Slow and Small Unmanned Aircraft Systems.
LWIR	Long Wave InfraRed.

mAP Mean Average Precision.

MOT Multi-Object Tracking.

MOTA Multiple Object Tracking Accuracy.

NATO North Atlantic Treaty Organization.

NMS Non-Maximum Suppression.

RADAR Radio Detection and Ranging.

RGB Red, Green and Blue.

ROI Region of Interest.

SORT Simple Online and Realtime Tracking.

SWIR Short Wave InfraRed.

UAS Unmanned Aircraft Systems.

UAV Unmanned Aerial Vehicle.

YOLO You Only Look Once.

YOLOR You Only Learn One Representation.

Chapter 1

Introduction

In this chapter, the context of this thesis is described, with an overview of the current capabilities of Unmanned Aerial Vehicles (UAVs) and events indicating their potential. Afterwards, a brief summary of the related work and state of the art is presented, serving as a basis for this project. This chapter ends with an overview of the content of this thesis, as well as its outline.

1.1 Context and Motivation

With the advancement of technology, it is becoming ever so easier to acquire small aircraft piloted by remote control or on-board computers, often referred to as drones. As the capabilities of these small aircraft increase, so does their ability to operate more effectively for both beneficial and harmful purposes. Due to their small size, affordability and versatility, UAVs are becoming increasingly popular and their applications are widening. These applications include precision agriculture, remote inspections, search and rescue, environmental monitoring, among others [1].

Despite all these industrial applications, the main market for small UAVs is still the recreational one. As of the end of May 2021, 865 505 drones had been registered in the United States with the Federal Aviation Administration (FAA), 62% of these consisting of recreational drone registrations [2]. Hobbyists, empowered by the relatively low prices of these flying devices, as well as recent technological advances in computer control, computer vision and high resolution cameras, have been rapidly increasing the popularity of small, personal drones, used in aerial photography, racing, etc.

This increase in popularity has raised some concerns regarding safety and privacy. Small personal drones can be controlled remotely, can be easily transported and can be modified to carry harmful payloads, allowing them to target both individuals and infrastructure. For instance, between the 19th and the 21th of December 2018, Gatwick airport, in London, had to close due to repetetive drone sightings, affecting more than 140,000 passengers and cancelling more than 1,000 flights [3]. Earlier the same year, on the 4th of August 2018, two drones detonated explosives at a military event in Caracas, during a speech by the President of Venezuela, Nicolás Maduro, in an apparent assassination attempt [4]. Even more recently, in 2022, small commercial drones have seen extensive use in the Russia-Ukraine war, with reported uses including: collecting footage of possible war crimes, inspecting buildings that have been hit, helping restore power infrastructure that has been damaged and even fitting explosive payloads on suicide drones to disable enemy armored vehicles [5] [6].

According to the North Atlantic Treaty Organization (NATO), Low, Slow and Small Unmanned Aircraft Systems (LSS-UAS) with surveillance sensors or explosive ordnance as payload create new challenges for today's war fighters [7]. Aware of this threat, from the 2nd until the 12th of November 2021, NATO

Communications and Information Agency (NCI Agency) organized a counter-drone exercise at the Lieutenant General Best Barracks in Vredepeel, Netherlands, in a live testing event used to ensure that commercial systems from different NATO nations could work together to counter the threats posed by these small drones [8].

NATO's industrial advisory group (NIAG) categorizes LSS-UAS according to their characteristics as presented in table 1.1.

Table 1.1: NATO classification of LSS-UAS [7].

Categories	Weight[kg]	Payload [kg]	Coverage [km]	Endurance [h]	Altitude [m]
Nano	< 0.5	< 0.1	< 1.5	< 0.5	< 100
Micro	< 2	< 1	< 10	< 1.5	< 1500
Mini Light	< 10	< 5	< 25	< 3	< 3000
Mini Heavy	< 25	< 12	< 50	< 5	< 4000
Small	< 150	< 50	< 150	< 12	< 6000

The work developed in this thesis focuses on the detection and tracking of Unmanned Aircraft Systems (UAS) in the Micro, Mini Light and Mini heavy categories, as these are the consumer grade UAVs with the highest potential to conduct nefarious missions, with possible damage to both individuals and infrastructure.

One of the reasons why small UAS are so effective, is the lack of ability to accurately detect and restrain one of these devices. These drones can take off and land from practically anywhere, can fly in very complex scenarios, (such as urban environments) and their relatively small size makes them difficult to locate. Their size and agility also makes intercepting them a very difficult task. The current solutions for the problem of detection and tracking of small UAS involve the use of one, or several of the following approaches:

- **Eletro-Optical/Infrared (IR) sensors** - This solution deploys the use of visible light and/or IR cameras in the frequency range (300 GHz - 790 THz). These cameras gather data easily interpretable by humans and that can be used for detection and identification. Although visible light cameras have a limited scope of operation, infrared cameras can operate in cloudy weather and in day or nightime conditions. The disadvantage of this approach resides on the requirement for a line of sight and a limited field of view of these sensors, which are also quite complex with computationally expensive processing algorithms.

- **Acoustic sensors** - This solution deploys an array of microphones, as well as an acoustic signature library to detect and identify UAVs. These microphones are cheap, lightweight and consume very little power when compared to the alternatives. The downsides of this approach are the vulnerability to decoys and the difficulty to use in noisy environments like airports or in airborne applications due to strong acoustic interference.

- **Radio-frequency sensors** - This solution exploits the communication link most UAVs use for remote control through the use of sensors which listen for signals in these frequencies. Radio-frequency sensors are often low-complexity and, therefore, easy to implement. They can operate in all weather and day/night conditions, are very effective against off-the-shelf UAVs and can even be used to locate the pilot and take control of the drone, under some conditions. The disadvantages of this approach are that knowledge regarding UAV communication specifications (e.g. frequency bands, modulations, etc.) is required, which could leave the system blind to modified radio frequencies that exceed receiver capabilities. This solution is also extremely vulnerable to completely autonomous systems which do not require communication.

- **RADAR** - Radio Detection and Ranging (RADAR) has been the traditional way to detect airborne threats for several decades. Radars offer high coverage, good accuracy and high reliability, as they can operate in all weather and day/night conditions and can even offer information regarding the velocity of the target. The downsides of this approach are: the limited performance for low altitudes and speeds, the line of sight requirement and the high cost and power consumption. The relatively low cross section most UAVs tend to present, can also significantly hinder detection and identification, as birds or other objects can be easily confused for drones.
- **LiDAR** - Similar to RADAR, Light detection and ranging (LiDAR) also emits pulses of radiation and waits for a response. LiDAR uses the same principles as RADAR, but instead of radio waves, short and precise laser light impulses are emitted. Even though more compact and more suited for small distances, LiDAR shares most of the advantages and disadvantages of RADAR, as it is also heavy, expensive and requires a lot of power. The point cloud it provides is also too sparse for precise identifications.

Currently, the advances in visual object tracking systems are making the use of eletro-optical and infrared sensors an increasingly valuable approach, with these systems being able to achieve an almost human degree of visual perception.

1.2 Related Work

Over the years, several visual object tracking frameworks have been proposed, empowered by the recent advances in computer vision. These tracking frameworks can be divided into three main categories: correlation filter based trackers, siamese network based trackers, and detection based trackers [9].

Correlation filter based trackers can detect objects very fast in the frequency domain and are able to learn how targets change in real time. In [10] a background-aware correlation filter based on handcrafted features that can efficiently model how both the foreground and background of the object varies over time is proposed. The authors from [11] use a shallow network to learn the representation of aircraft in the embedding space, and integrate the feature embeddings into an efficient convolution operator framework for aircraft tracking using infrared-imagery. In [12] a computationally efficient Red, Green and Blue (RGB) and infrared UAV detection and tracking system is proposed. This system uses the output from YOLOv4 to initialize a discriminative correlation filter based object tracker allowing for a higher tracking speed.

Siamese trackers learn a matching function that is used to search for a previously detected instance of an object within a new and larger contextual region. SiamMOT [13] introduces a region-based Siamese Multi-Object Tracking network, which includes a motion model that estimates the object's movement between two frames such that detected instances are associated. SiamTPN [14] proposes a Siamese Transformer Pyramid Network, which exploits the inherent feature pyramid of a lightweight network (ShuffleNetV2) and reinforce it with a Transformer to construct a robust target-specific appearance model.

Detection based models are the most common in the literature, mainly due to their simplicity, as they just adapt other networks used primarily in object detection to the task of object tracking. In [15] common object detectors, such as Faster RCNN [16], SSD512 [17], YOLOv3 [18] and DETR [19] are combined with trackers: SORT [20], Deep SORT [21] and Tracktor [22] to establish a comprehensive performance benchmark for comparing these algorithms in the context of UAV visual detection and tracking using deep neural networks.

In the context of applying detection based algorithms to infrared systems: in [23] the Single Shot Detector (SSD) [17] is used to detect persons, bicycles and cars in infrared-videos and in [24] YOLOv3 [18] is trained to detect persons (both civilians and first-responders) and vehicles (both civilian-cars and response-vehicles) in search and rescue scenarios using RGB and infrared images. Similarly, [25] uses YOLOv4 [26] on RGB and thermal images for human detection from an aerial perspective. Finally, [27] uses the Faster-RCNN [16] as a detector and Multi-Domain Network (MDNet) [28] as a tracker for visible and thermal drone monitoring and [29] uses YOLOv2 and a multi-object Kalman filter tracker, similar to SORT [20], for real-time drone detection and tracking with visible, thermal and acoustic sensors.

Lastly, there are approaches that take advantage of the moving nature of some objects to facilitate their tracking. In [30], a vision-based detection of non-cooperative UAVs is proposed. This system uses frame differencing followed by a spatial filter to detect candidate points for intruders in the images. Afterwards, candidate points are tracked over time to only keep candidate points which are successfully detected in several consecutive frames and the results are refined using a Kalman filter. In [31] a moving object detector using thermal imagery is proposed to detect humans and vehicles. The first stage of this system consists on using a probabilistic model (i.e. Gaussian Mixture Model) to detect moving objects in both visual and infrared frames and segmenting these into foreground and background pixels. The two resulting binary masks are then combined by fusing the foreground regions and are thereafter post-processed to finally obtain Regions of interest (ROIs) in the visual frame. The second (and final) stage consists on taking the images from the ROIs, resizing them and feeding them into a pre-trained Convolutional Neural Network (CNN) which is used for feature extraction and classification of the moving objects.

1.3 Project Overview

The work developed in this thesis carries on the work by Daniel Justino [32] and Mariana Santos [33] at the Center for Aerospace Research (CfAR), who developed a system for detection and tracking of non-cooperative UAS based on LiDAR and camera data. This thesis focuses only on the detection and tracking of these UAVs using the information from the camera, in this case, an infrared camera.

The proposed method combines the YOLOv7 detector with a new version of the Bytetrack tracker to achieve reliable tracking at a rate of nearly 30 frames per second. Other trackers like SORT and a modified version of Deep SORT are also tested, as well as the YOLOR detector. The final system takes a continuous stream of infrared frames, starts by aligning and subtracting consecutive images (to highlight moving objects) and concatenates this result with the current frame. This information in then scanned by the YOLOv7 detector to find possible targets in the image. The results from the detector are then aligned to compensate for camera motion and given to the ByteTrack tracker to improve the results, rejecting false positives and linking consecutive detections belonging to the same object into a track with a unique Identification (ID).

Note that the first alignment is simply used to transform the previous image to the current frame of reference, to perform the subtraction, whereas the second alignment is used to provide the tracker with consistent bounding box positions (i.e. in the frame of reference of the first image on the respective video).

1.4 Thesis Outline

Apart from this initial introductory chapter, this thesis is organized as follows: Chapter 2 describes the state-of-the-art methods used throughout the literature and that will also be applied to this work. This chapter also presents the evaluation metrics for the tasks of object detection and tracking which will later be used to compare the several algorithms. Chapter 3, Methodology, describes the work developed in this dissertation, explaining in detail all the steps taken to put together the proposed system. Chapter 4, Results, presents the outcome of the experiments conducted to compare the different approaches. Finally, Chapter 5, Conclusions and Future Work, briefly goes over this entire project, summing up its key findings. This chapter also describes the contribution and future approaches which could be explored to further improve the results.

Chapter 2

Theoretical Background

This chapter starts by describing the technical concepts used in this thesis, as well as some of the algorithms used to tackle the main computer vision challenges. The evaluation metrics used to compare the several models are also explained and defined, first for the task of object detection, followed by the task of multi-object tracking.

2.1 Visible and Infrared Light Cameras

A camera is an optical instrument which captures and records visual images of scenes. Digital cameras achieve this through an array of sensors which collect visible light, and circuitry which converts this information into electric signals. As the name suggests, visible light cameras are sensitive to visible light, with wavelengths in the 400-700 nm range, the same spectrum perceived by the human eye. These cameras are designed to replicate human vision capturing light in the red, green and blue wavelengths (RGB) for accurate color representation.

Just like the human eye, visible light cameras require light. Their performance is also greatly deteriorated by atmospheric conditions such as fog, haze, smoke, and even heat waves. In the context of surveillance, visible light cameras often need to be paired with illumination or thermal infrared cameras in order to work at night, in low illumination scenes, or in other environments where visible cameras are insufficient [34].

A thermal camera or thermal imager is a device that creates an image in much the same way as a visible light camera, with the difference residing on the wavelengths of the radiation used, in this case, infrared radiation (IR). Infrared cameras are usually sensitive to wavelengths from about 1 µm (1000 nm), corresponding to Short Wave InfraRed (SWIR) to about 14 µm, corresponding to Long Wave InfraRed (LWIR). In Figure 2.1 these wavelengths are presented in the context of the entire electromagnetic spectrum.

LWIR imaging has the advantage of being able to perceive objects by their own light and discriminate between them based on temperature. This is due to the Planck curve for blackbody radiation which peaks in the LWIR (at about 9 µm) for objects at room-temperature (300 K). In addition, LWIR imaging works well for outdoor use due to the relatively low output power of the Sun at these wavelengths [36].

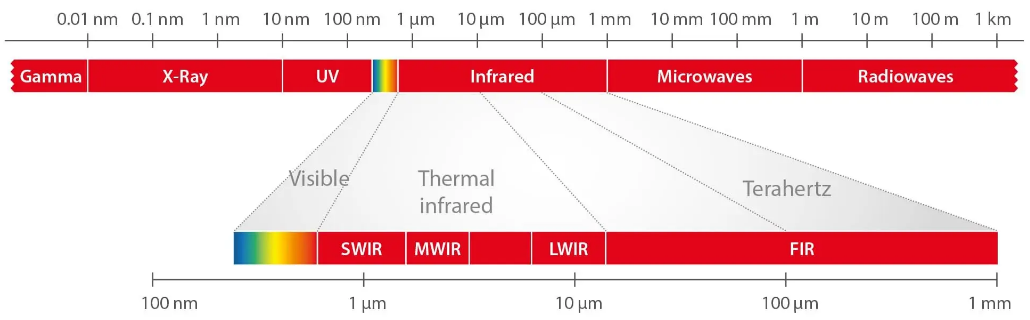


Figure 2.1: Electromagnetic spectrum [35].

2.2 Image Alignment

Image alignment is the technique of warping one image (sometimes two) so that the features in both images line up perfectly. In a typical image alignment problem there are two images of a scene related by a motion model (or transformation). Different image alignment algorithms aim to estimate the parameters of these motion models using different tricks and assumptions. Once these parameters are known, warping one image so that it aligns with the other is straight forward [37].

Usually algorithms try to estimate one of the following motion models (in increasing order of complexity):

- Translation: this model assumes one image is a shifted version of the second image. Only requires the estimation of two parameters, x and y.
- Euclidean: this model assumes one image is a rotated and shifted version of the second image. Requires the estimation of three parameters, x, y and angle of rotation.
- Affine: this model assumes the first image has been transformed using a combination of rotation, translation, scale, and shear. This motion model requires 6 parameters: two for translation, one for scale, one for rotation and two for shear. When a square undergoes an Affine transformation, parallel lines remain parallel, but lines meeting at right angles can no longer remain orthogonal.
- Homography: so far all transformations only account for 2D effects. The Homography transformation is a three dimensional mapping between two planar projections of an image and is represented by a 3x3 transformation matrix.

A visual representation of these motion models is presented in Figure 2.2.

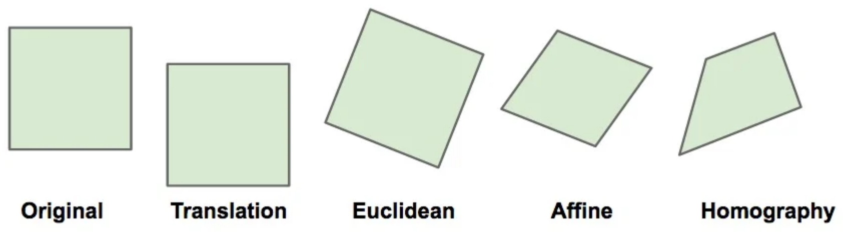


Figure 2.2: Motion models [37].

2.2.1 ECC Algorithm

One algorithm that aims at estimating the motion model between two given images, is the Enhanced Correlation Coefficient (ECC) maximization [38]. The ECC algorithm is a gradient-based iterative image alignment algorithm which achieves high accuracy in parameter estimation (i.e. subpixel accuracy) with relatively low computational complexity. This algorithm is also invariant to photometric distortions in contrast and brightness since it considers the correlation coefficient (zero-mean normalized cross correlation) as an objective function.

Starting with a pair of image profiles: $I_r(\mathbf{x})$ and $I_w(\mathbf{y})$, where the first is the reference (or template) image and the second is the warped image; and using $\mathbf{x} = [x_1, x_2]^t$, $\mathbf{y} = [y_1, y_2]^t$ to denote coordinates. Considering a set of coordinates $T = \{\mathbf{x}_k, k = 1, \dots, K\}$ in the reference image, the alignment problem consists on finding the corresponding coordinate set in the warped image.

The focus is not on arbitrary correspondences though, but in those that are structured and can be modeled with a well-defined vector mapping $\mathbf{y} = \phi(\mathbf{x}; \mathbf{p})$, where $\mathbf{p} = [p_1, \cdots, p_N]^t$ is a vector of unknown parameters.

The alignment problem is therefore reduced to the problem of estimating the parameters $\mathbf{p}$ such that:

$$I_r(\mathbf{x}) = I_w\left(\phi(\mathbf{x}; \mathbf{p})\right), \quad \forall\, \mathbf{x} \in T. \tag{2.1}$$

Under the warping transformation, $\phi(\mathbf{x}; \mathbf{p})$, the coordinates T are mapped into the coordinates $\mathbf{y}_k(\mathbf{p}) = \phi(\mathbf{x}_k; \mathbf{p})$. Defining the reference vector, $\mathbf{i}_r$, and the corresponding warped vector, $\mathbf{i}_w(\mathbf{p})$, as:

$$\begin{gathered} \mathbf{i}_r = [I_r(\mathbf{x}_1)\ I_r(\mathbf{x}_2) \cdots I_r(\mathbf{x}_K)]^t, \\ \mathbf{i}_w(\mathbf{p}) = [I_w(\mathbf{y}_1(\mathbf{p}))\ I_w(\mathbf{y}_2(\mathbf{p})) \cdots I_w(\mathbf{y}_K(\mathbf{p}))]^t, \end{gathered} \tag{2.2}$$

and denoting with $\bar{\mathbf{i}}_r$ and $\bar{\mathbf{i}}_w(\mathbf{p})$ their respective zero-mean versions (obtained by subtracting from each vector its corresponding arithmetic mean), the algorithm uses the following criterion to quantify the performance of the warping transformation:

$$E_{\mathrm{ECC}}(\mathbf{p}) = \left\| \frac{\bar{\mathbf{i}}_r}{\|\bar{\mathbf{i}}_r\|} - \frac{\bar{\mathbf{i}}_w(\mathbf{p})}{\|\bar{\mathbf{i}}_w(\mathbf{p})\|} \right\|^2, \tag{2.3}$$

where $\|.\|$ denotes the usual euclidean norm.

Minimizing $E_{ECC(p)}$ is equivalent to maximizing the following enhanced correlation coefficient:

$$\rho(\mathbf{p}) = \frac{\bar{\mathbf{i}}_r^t \bar{\mathbf{i}}_w(\mathbf{p})}{\|\bar{\mathbf{i}}_r\|\, \|\bar{\mathbf{i}}_w(\mathbf{p})\|} = \hat{\mathbf{i}}_r^t \frac{\bar{\mathbf{i}}_w(\mathbf{p})}{\|\bar{\mathbf{i}}_w(\mathbf{p})\|}, \tag{2.4}$$

where, for simplicity, $\hat{\mathbf{i}}_r = \bar{\mathbf{i}}_r / \|\bar{\mathbf{i}}_r\|$ denotes the normalized version of the zero-mean reference vector, which is constant.

The maximization of $\rho(\mathbf{p})$ is now performed using gradient-based approaches, replacing the original optimization problem with a sequence of secondary optimizations. This algorithm can be used to estimate any of the above mentioned motion models, with the transformation being a hyper-parameter of the ECC function used to trade-off speed for complexity.

Note that it is common to apply a scaling factor to both the images, reducing their size, before running this algorithm. Such practice allows for faster runtimes and improves the probability of convergence, however, it also leads to a loss of information, resulting in less accurate predictions.

2.3 Visual Object Detection

To fully comprehend an image, precisely estimating the concepts and locations of the objects contained in it, is one of the core computer vision problems. This task is referred to as object detection and its objective is to surround an object with a bounding box and classify which type of object it is.

Visual detection algorithms can be loosely divided in three main steps: informative region selection, feature extraction, and classification [39]. Informative region selection consists on trying to find the approximate location of every object in the image, for instance, by using exhaustive search strategies like sliding windows. Feature extraction consists on transforming the raw data (i.e. pixel values) into derived values (features), containing the relevant information in a more abstract, complex and rich format, facilitating the subsequent step. Classification consists on assigning a label to an object given its features, effectively identifying it as part of a particular category (or class).

State-of-the-art object detectors can be divided into two main categories: one-stage detectors and two stage-detectors. Two-stage detectors follow the above mentioned three steps. These models start by generating region proposals where the models draw candidates of objects in the images (independent from the category), and only then do they extract the features from these candidates and classify them. These methods prioritize detection accuracy, and examples include Faster R-CNN [16], Mask R-CNN [40] and Cascade R-CNN [41]. One-stage detectors skip the region proposal step, and instead run the whole image through a neural network and output a fixed number of predictions. These methods prioritize inference speed, and some examples are: YOLO [42] , SSD [17] and RetinaNet [43].

2.3.1 YOLO

You Only Look Once (YOLO) [42], and its variants, are the current state-of-the-art in real-time object detection. YOLO uses a deep CNN to detect any sort of object present in an image. It was first introduced in 2016 and has gained a lot of popularity over the years due to its speed and accuracy. The input to this network is a tensor with dimensions $w \times h \times d$, where w and h are the width and height of the input image and d in the number of channels, usually three (red, green and blue).

YOLO starts by dividing the input image into an $S \times S$ grid, with each cell in this grid being responsible for detecting the objects centered in it. Each cell outputs B bounding box detections, with every detection consisting on the probability that there is an object, followed the description of that object (i.e. center position (x, y) and dimensions (w, h)). Each grid cell also predicts C values corresponding to the conditional class probabilities $P_r(Class_i|Object)$. The output of this network is therefore a tensor of size $S \times S \times F$, where S is a hyper parameter (originally $S = 7$) and F is the number of output filters:

$$F = B \times 5 + C. \tag{2.5}$$

Since the YOLO architecture always outputs $S \times S \times B$ bounding boxes, two steps are taken after running the neural network to hopefully reject all the bounding boxes which do not correspond to an object in the image.

The first step aims at filtering out unfit predictions. This step consists on discarding all bounding boxes with a probability that there is an object (also known as confidence score) smaller that a fixed threshold, for instance: 0.5.

The second step, Non-Maximum Suppression (NMS), is taken to reject duplicate detections of the same object. This stage consists on going through every remaining detection box in order of maximum confidence, and excluding every other detection with an IoU greater that a fixed threshold (IoU_{thresh}).

A schema of the YOLO algorithm is presented in Figure 2.3. In this figure it is possible to visualize the grid applied to the input image, followed by the bounding box and class probabilities predicted by the

CNN, as well as the final detections after the two post-processing algorithms.

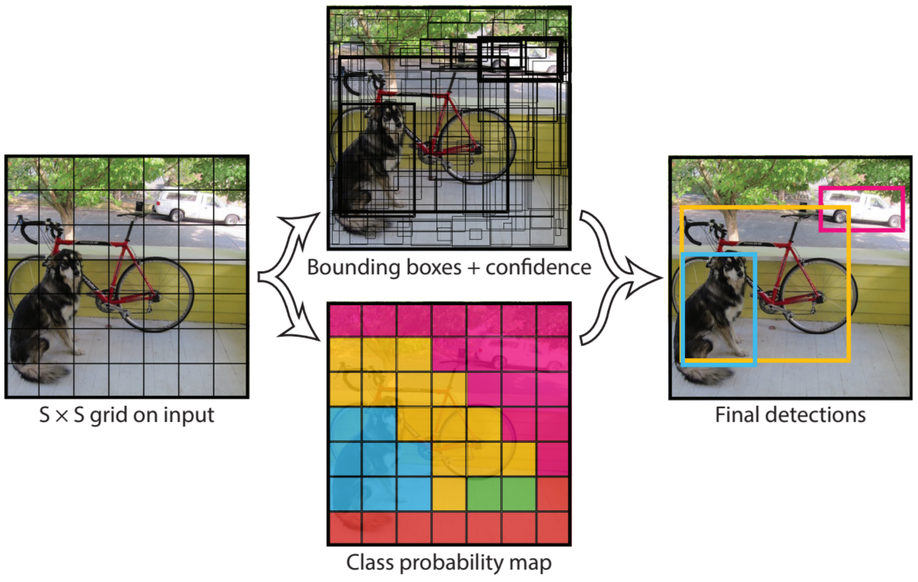


Figure 2.3: Different steps of the YOLO model [42].

Over the years several iterations of the YOLO algorithm have been proposed, adding better training practices, as well as, incremental improvements to the CNN, improving the overall performance of this object detector.

2.3.2 YOLO Iterations

The original YOLO implementation only predicted one set of class probabilities per grid cell, regardless of the number of bounding boxes, B. YOLOv2 [44] changed this allowing each box to have its own set of class probabilities, effectively allowing one grid cell to detect instances of objects belonging to different classes. In practise, this meant changing the number of filters in the last layer to:

$$F = B(C + 5). \tag{2.6}$$

Along with this, each bounding box B would now be specialized in the detection of objects of a specific size and aspect ratio. This concept is called anchor boxes and the expected shape of objects was determined trough k-means clustering of all the bounding boxes in the training dataset. This and other changes like batch normalization resulted in a more accurate detector which was still fast.

Several other variations of the original YOLO model have been published since its introduction in 2016. Recently, two YOLO models have stood out for their relatively high accuracies and low inference times. These models are the YOLOR [45] and YOLOv7 [46].

2.3.3 YOLOR

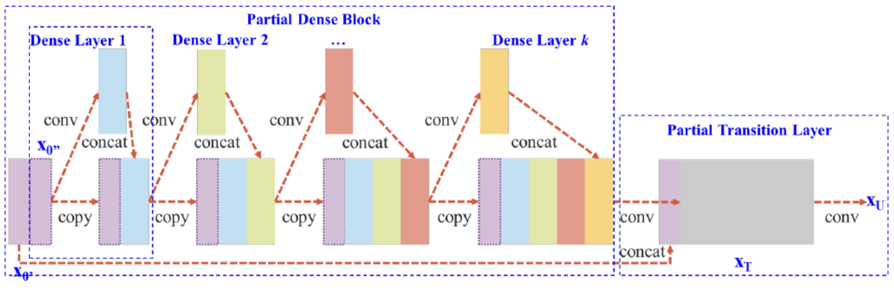
YOLOR [45] was introduced in 2021 and proposes a network that can generate a unified representation to simultaneously serve various tasks. To achieve this, the authors propose to encode together explicit knowledge (knowledge that directly corresponds to observations) and implicit knowledge (knowledge incorporated in the model and that has nothing to do with observation). In YOLOR explicit knowledge is obtained from the shallow layers of the neural networks and Implicit knowledge is modeled as vectors, neural networks and matrix factorization with parameters learned during training. Explicit and implicit knowledge are then merged using addition, multiplication or concatenation.

An overview of the different models proposed by the YOLOR authors is presented in Table 2.1.

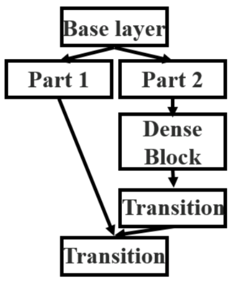
Table 2.1: Results for the different YOLOR models presented by the authors [45], obtained using the COCO dataset [47].

Model	Test Size	APtest	AP50test	AP75test	batch1 throughput	batch32 inference
YOLOR-CSP	640	52.80%	71.20%	57.60%	106 FPS	3.2 ms
YOLOR-CSP-X	640	54.80%	73.10%	59.70%	87 FPS	5.5 ms
YOLOR-P6	1280	55.70%	73.30%	61.00%	76 FPS	8.3 ms
YOLOR-W6	1280	56.90%	74.40%	62.20%	66 FPS	10.7 ms
YOLOR-E6	1280	57.60%	75.20%	63.00%	45 FPS	17.1 ms
YOLOR-D6	1280	58.20%	75.80%	63.80%	34 FPS	21.8 ms

In this table, models are presented in increasing order of complexity, leading to higher accuracies but slower inference speeds. Cross Stage Partial (CSP) models refer to the CSPNet [48] backbone used by the network to enhance its learning capability. The CSPNet architecture divides the layers of the CNN into several stages. The model then separates the feature map of the base layer of a stage into two parts, one part goes through a sequence of dense convolutional layers followed by a transition layer; the other part is concatenated with the output of this transition layer to pass to the next stage through another transition layer. Transition layers are layers that change feature-map sizes via convolution and pooling. In Figure 2.4 two similar diagrams of a stage of the CSPNet backbone are presented.



(a) Diagram complete with all the convolution (conv), concatenation (concat) and copy operations.



(b) Simplified diagram.

Figure 2.4: Diagrams of a stage of the CSPNet backbone [48].

2.3.4 YOLOv7

YOLOv7 was introduced in 2022 and is now the state-of-the-art in real-time object detection [49]. The main features introduced or improved in YOLOv7 are: model re-parameterization, model scaling and an Extended Efficient Layer Aggregation Network (E-ELAN).

Model re-parameterization refers to the practice of averaging a set of model weights to create a model that is more robust to the general patterns it is trying to learn. The authors of YOLOv7 use gradient flow propagation paths to see which modules in the network should use re-parameterization strategies and which should not.

Model scaling is a way to scale up or down an already existing model to make it fit different applications and computing devices. It applies different scaling factors to several hyper-parameters such as resolution (size of input image), depth (number of layers), width (number of channels), or the number of stages, in order to achieve a good trade-off in the amount of network parameters, inference speed, and accuracy. The YOLOv7 authors propose that when performing model scaling on concatenation based models, only the depth of a computational block should be scaled, with a corresponding width scaling in the transition layer to allow the concatenation of the extra layers.

Finally, the authors of YOLOv7 propose an Extended Efficient Layer Aggregation Network, which they named E-ELAN, as an alternative to the CSP architecture used in YOLOR. E-ELAN doesn't affect the original gradient path, but uses group convolution to increase the cardinality (the size of the set of transformations) of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This can enhance the features learned by different feature maps, improving the use of parameters and calculations.

An overview of the different models proposed by the YOLOv7 authors is presented in Table 2.2.

Table 2.2: Results for the different YOLOv7 models presented by the authors [46], obtained using the COCO dataset [47].

Model	Test Size	APtest	AP50test	AP75test	batch 1 FPS	batch 32 average time
YOLOv7	640	51.40%	69.70%	55.90%	161 FPS	2.8 ms
YOLOv7-X	640	53.10%	71.20%	57.80%	114 FPS	4.3 ms
YOLOv7-W6	1280	54.90%	72.60%	60.10%	84 FPS	7.6 ms
YOLOv7-E6	1280	56.00%	73.50%	61.20%	56 FPS	12.3 ms
YOLOv7-D6	1280	56.60%	74.00%	61.80%	44 FPS	15.0 ms
YOLOv7-E6E	1280	56.80%	74.40%	62.10%	36 FPS	18.7 ms

Comparing YOLOv7 to YOLOR, from the tables 2.1 and 2.2 it is possible to conclude that YOLOR appears to have higher average precision whereas YOLOv7 focuses on achieving faster inference speed.

2.4 Multi-object Tracking

Multi-Object Tracking (MOT) is the problem of automatically identifying multiple objects in a video and representing them as a set of trajectories with high accuracy [50]. Some MOT systems make use of pre-trained object detectors to locate target objects within individual frames, followed by trackers which connect these sets of detections across time, forming tracks with unique track IDs. This approach is referred to as tracking-by-detection, and multiple algorithms have been presented over the years, like the SORT [20], Deep SORT [21] and ByteTrack [51]. Typically these algorithms share part of the following steps:

(i) detection - where a pre-trained object detector tries to find objects of interest in an image;

(ii) motion prediction - where the tracker tries to predict the position of the objects in the current frame based on previous detections;

(iii) affinity stage - where the tracker computes the similarity between the predictions and the detections, using the position, size, and possibly more advanced features like appearance descriptors;

(iv) association stage - where the similarity is used to assign the new detections to the respective objects being tracked.

More recent works have shifted the focus to a tracking-by-attention paradigm, motivated by the recent advances in deep learning, empowered by the transformer architecture (with the use of attention mechanisms), initially in natural language processing, and now in object detection. Some examples of transformer based trackers include: MOTR [52], TrackFormer [53] and TRAT [54]. Even though these architectures are promising and can achieve state-of-the-art tracking results, they still fall behind the above mentioned tracking-by-detection systems in terms of inference speed.

2.4.1 SORT

SORT [20] is a tracking-by-detection algorithm that tries to achieve reliable tracking with minimal computational complexity. It works by describing every track (and, therefore, every object) with a unique track ID and a Kalman filter with state:

$$x = [u \quad v \quad s \quad r \quad \dot{u} \quad \dot{v} \quad \dot{s}]^T, \tag{2.7}$$

where u and v represent the bounding box center coordinates (in pixels), s the bounding box area, r the aspect ratio, and the three remaining variables represent the respective derivatives. Note that the aspect ratio is considered to be constant.

The tracker is updated for every frame in the video, and the IoU metric is used to compute the similarity between all the detections and all the targets being tracked. The assignment step is then solved optimally via the Hungarian algorithm, using an IoU threshold (IoU_{min}) to reject unfit assignments. When a detection is paired with a target, the detected bounding box is used to update the track state, with the velocities being solved optimally by the Kalman filter framework. If no detection is associated to the target, its state is simply predicted without correction using the linear velocity model.

New tracks are created every time a detection isn't assigned to an existing track, but they stay in a probationary period until enough detections are assigned to that object to activate it. Tracks are initialized using the geometry of the detected bounding box and setting the velocity to zero (the covariance of the velocity component is also set to large values, reflecting its uncertainty). Tracks are terminated if they are not detected for T_{Lost} frames.

SORT doesn't have any learnable parameters and only requires, as input for every iteration, a list with the bounding boxes detected for that particular frame. It can therefore work together with any object detector with minimal modifications.

2.4.2 Deep SORT

Simple Online and Realtime Tracking with a Deep association metric (Deep SORT) [21] is an extension to SORT that integrates appearance information to improve the performance of the baseline algorithm. Due to this extension, the system is able to track objects through longer periods of occlusions and can more easily differentiate objects even if they come within extreme proximity of each other.

This algorithm replaces the original association metric with a more informed similarity that can be broken down into two parts: one for motion similarity, $d^{(1)}$, and one for appearance similarity, $d^{(2)}$. To build the association problem, both metrics are then combined using a weighted sum:

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j), \tag{2.8}$$

where λ is the hyper-parameter used to tune the weight given to each of the two types of information.

To measure motion similarity, the squared Mahalanobis distance between the newly arrived measurements and the predicted Kalman Filter states is used, according to:

$$d^{(1)}(i,j) = (\mathbf{d}_j - \mathbf{y}_i)^T \mathbf{S}_i^{-1} (\mathbf{d}_j - \mathbf{y}_i), \tag{2.9}$$

where $\mathbf{S}_i$ represents the covariance matrix of the i-th track distribution; $\mathbf{d}_j$ represents the j-th box detection and $\mathbf{y}_i$ the i-th track in memory. The Mahalanobis distance is used to take state estimation uncertainty into account, measuring how many standard deviations the detection is away from the mean track location.

In order to quantify visual resemblance, a convolutional neural network, trained to discriminate pedestrians on a large-scale person re-identification dataset, is first used to generate appearance descriptors. The appearance similarity is then given by the smallest cosine distance between the descriptors of the i-th track and the j-th detection:

$$d^{(2)}(i,j) = min\{1 - \mathbf{r}_j^T \mathbf{r}_k^{(i)} | \mathbf{r}_k^{(i)} \in \mathcal{R}_i\}, \tag{2.10}$$

where $\mathbf{r_j}$ is the appearance descriptor corresponding to the j-th detection, with $||\mathbf{r_j}|| = 1$, and $\mathcal{R}_i$ is a gallery with the last $L_k = 100$ associated appearance descriptors for each track i.

Note that Deep SORT uses a Kalman filter state slightly different to its predecessor. This time the aspect ratio is not considered constant and tracks are described by:

$$\mathbf{x} = [u \quad v \quad \gamma \quad h \quad \dot{x} \quad \dot{y} \quad \dot{\gamma} \quad \dot{h}]^T, \tag{2.11}$$

where u and v represent the bounding box center coordinates, γ the aspect ratio, h the bounding box height, and the four remaining variables represent the respective derivatives, all in pixel coordinates.

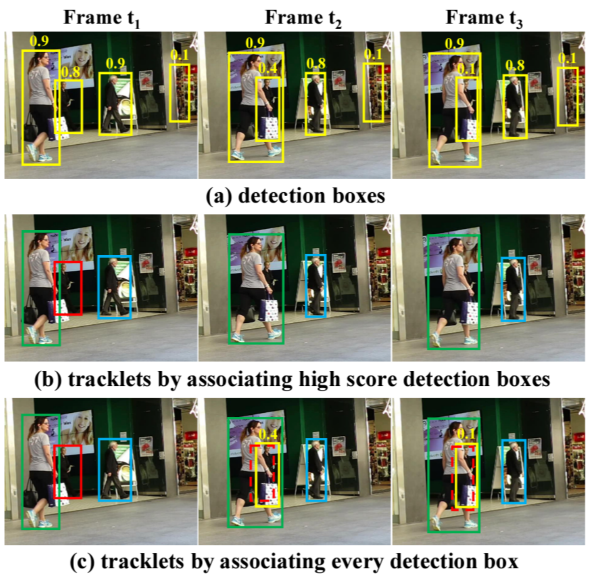
2.4.3 ByteTrack

ByteTrack [51] is another multi-object tracker heavily inspired by SORT. Like SORT, it uses IoU to assign the new detections to the tracks in memory and uses Kalman filters to predict the positions of the objects in the current frame, given the past detections. For the state of the Kalman filter, it uses the same as Deep SORT, given by (2.11).

Where ByteTrack differs from the previous methods, is how it keeps every single detected box, unlike most trackers which only keep the high score detections (with a confidence greater than a threshold). The ByteTrack algorithm starts by separating the detections into high score and low score ones using a threshold. High score detection boxes are associated first with the corresponding tracks, with unmatched detections creating new tracks and unmatched tracks proceeding to the next step. In the next step, low score detection boxes are associated to the unmatched tracks, recovering the lost objects. In this step unmatched detections get discarded and unmatched tracks are deactivated but remain in memory for future assignments and keep being updated by the Kalman filter.

In Figure 2.5 it is possible to visualize the effect of associating every single detection box. Here, ByteTrack is able to keep assigning bounding boxes to the pedestrian represented by a red track, whereas most other trackers would simply lose this pedestrian as low score bounding boxes would be discarded.

As of April 2022, using MOTA (described in Section 2.5.2) as the performance metric, and the MOT17 dataset as a benchmark, ByteTrack is the state-of-the-art in multi-object tracking, achieving the first place in this dataset, as evidenced by Table 2.3.



Frame t_1
Frame t_2
Frame t_3
0.9
0.8
0.9
0.1
0.9
0.4
0.8
0.1
0.9
0.1
0.8
0.1
(a) detection boxes
(b) tracklets by associating high score detection boxes
0.4
0.1
(c) tracklets by associating every detection box

Figure 2.5: Effect of associating every detection box [51].

Table 2.3: Leaderboard of the MOT17 challenge (ranked by MOTA), as of April 2022 [55].

Rank	Model	MOTA	IDF1	Year
1	ByteTrack	80.3	77.3	2021
2	StrongSORT	79.6	79.5	2022
3	OC-SORT	78.0	77.5	2022
4	STGT	76.7	75.1	2021

2.5 Evaluation Metrics

In order to evaluate and compare the proposed work to the existing literature, it is first necessary to have a clear understanding of the metrics used to quantify the performance of both detectors and trackers.

The most commonly used metrics, presented bellow, usually vary between zero and one, where zero is the worst possible result and one is the best.

2.5.1 Object detection

Addressing the object detection metrics first, a few intermediary values have to be computed to get to the mean average precision, the most widely criterion to benchmark different detection models.

2.5.1.1 Intersection over union

The intersection over union is used to measure the similarity between two bounding boxes. Considering any two bounding boxes, B_1 and B_2, the IoU is given by:

$$IoU = \frac{Area(B_1 \cap B_2)}{Area(B_1 \cup B_2)}. \tag{2.12}$$

This metric is most commonly used to measure the similarity between predicted and ground truth bounding boxes, in particular, to classify whether two bounding boxes belong to the same object or not.

As given by (2.12), two bounding boxes will have an IoU of zero if they do not overlap, and an IoU of one if they overlap perfectly (*i.e.* if both have the same position and dimensions). In Figure 2.6 it is possible to visualize the intersection and union between two bounding boxes.

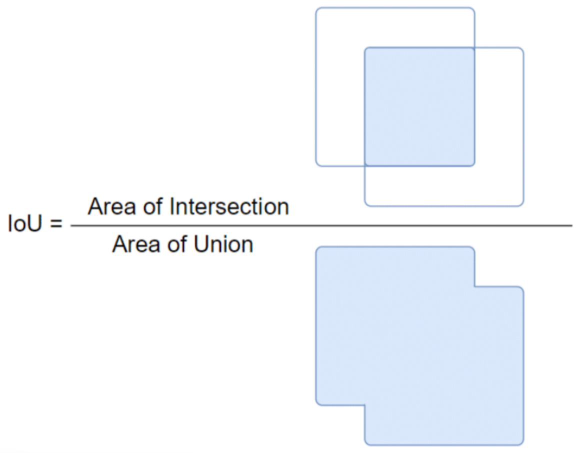


Figure 2.6: Visual representation of the IoU [33].

2.5.1.2 Precision

The Precision, P, is one of the metrics used to evaluate object detectors. It measures the ratio of detections which actually belong to an object in the ground truth. It is defined as the number of correct detections over the total number of detections predicted by the model:

$$P = \frac{TP}{TP + FP}, \tag{2.13}$$

where TP is the number of true positives and FP is the number of false positives. A model with a perfect precision is a model that only outputs correct detections.

Note that a detection is considered to be a TP if it has an IoU greater than a threshold with an object present in the ground truth, otherwise it will be considered a FP. This threshold is usually set to to 0.5.

2.5.1.3 Recall

The Recall, R, is another metric used to evaluate object detectors, complementary to the precision. It measures the ratio of objects in the ground truth correctly detected by the model. It is defined as the number of correct detections over the total number of objects present in an image:

$$R = \frac{TP}{TP + FN}, \tag{2.14}$$

where TP is the number of true positives and FN is the number of false negatives. A model with a perfect recall is capable of finding every object in an image.

2.5.1.4 Precision-Recall curve

Both precision and recall try to measure the performance of object detectors using complementary approaches, with the tuning of the confidence score threshold of the detector being used to either maximize precision or recall.

Lowering the confidence threshold will result in a lot of detections, probably including the objects in the ground truth, resulting in a high recall. On the other hand, increasing this threshold will result in very few predictions but with a high success rate, leading to a high precision.

This Precision-Recall trade-off can be plotted into a curve, by changing the confidence score threshold of the detector and getting multiple precision and recall points. One example of a typical Precision-Recall curve is illustrated in Figure 2.7.

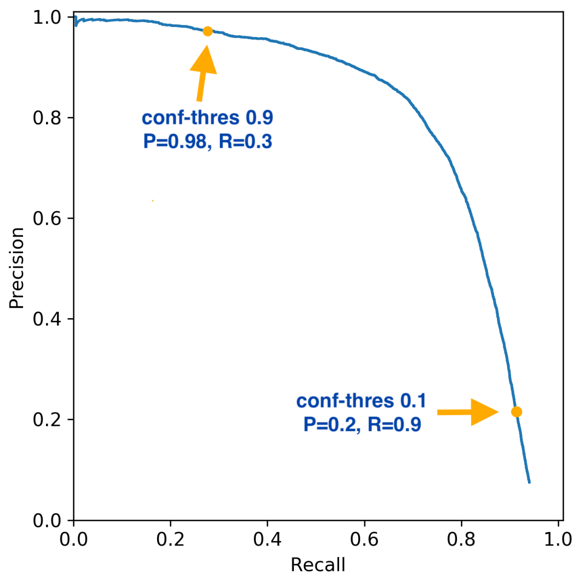


Figure 2.7: Precision-Recall curve example [56].

2.5.1.5 Average Precision

The Average Precision (AP) is defined as the area under the Precision-Recall curve:

$$AP = \sum_{n} (R_n - R_{n-1}) \times P_n, \tag{2.15}$$

where R_n is the recall value of point n, and P is the maximum precision for any recall value larger than R_n, i.e., $P_n = max\{P(R) : R \geq R_n\}$.

2.5.1.6 Mean Average Precision

The Mean Average Precision (mAP) is the average across all classes of the AP value per class,

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{2.16}$$

where N represents the total number of classes and AP_i the average precision of the i^{th} class.

Since this metric depends on the IoU threshold used to distinguish true positives from false positives, it is common for the mAP to be followed by the IoU threshold used. For instance, mAP@0.5 means the threshold used to compute the mAP was set to 0.5. Another common variation is mAP@[.5:.95], which is the average mAP using IoU thresholds from 0.5 to 0.95, with a step of 0.05.

2.5.2 Multi-Object Tracking (MOT)

As per "The CLEAR MOT Metrics" [57], tracker evaluation should, for each time frame, respect the following methodology:

(i) establish the best possible correspondence between hypotheses (the tracker output) h_j and objects in the ground truth O_i,

(ii) for each found correspondence, compute the error in the object's position estimation,

(iii) accumulate all correspondence errors:

   (a) False Negatives (FNs) or Misses,

   (b) False Positives (FPs),

   (c) Identification Switches (IDS) or Mismatches - occurrences where the tracking hypothesis for an object changes compared to previous frames. This could happen, for example, when an object track is reinitialized with a different track ID, after it was previously lost, or when two or more objects are swapped as they pass close to each other (a visual representation of this occurrence is presented in Figure 2.8).

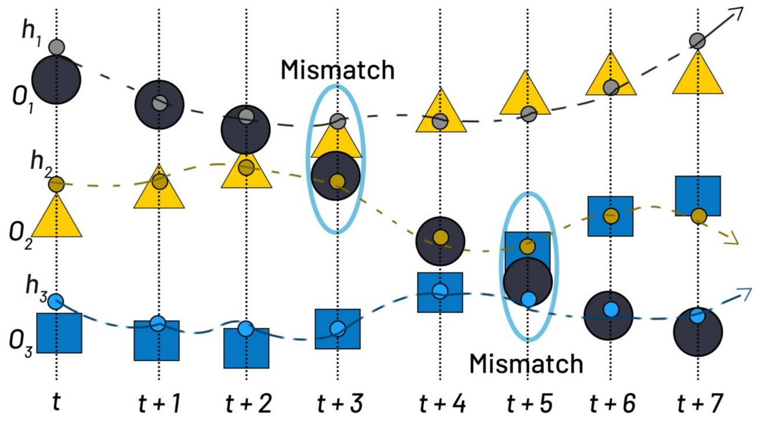


Figure 2.8: Visual representation of two mismatches (or ID switches) [58].

With these accumulated errors (after going trough every frame), two metrics are used to quantify the performance of multi-object trackers: IDF1 and MOTA.

2.5.2.1 IDF1 score

The F1-score taking the track ID into account (IDF1) balances identification precision and recall through their harmonic mean [59]. In practice, this means taking the ratio of correctly identified detections over the average number of ground-truth and computed detections:

$$IDF1 = \frac{2IDTP}{2IDTP + IDFP + IDFN}, \tag{2.17}$$

where IDTP refers to the true positives taking the ID into account (i.e. the bounding boxes and the IDs of the tracks and ground truth must match). The same goes for the ID false positives (IDFP) and ID false negatives (IDFN).

2.5.2.2 Multiple Object Tracking Accuracy

The Multiple Object Tracking Accuracy (MOTA) [60] uses a different expression to evaluate the performance of trackers, this time taking the number of mismatches directly into account. The MOTA score is given by:

$$MOTA = 1 - \frac{FN + FP + IDS}{GT}, \tag{2.18}$$

where the denominator: GT, refers to the total number of objects in the ground truth.

Note that technically it is possible for this metric to take negative values.

Chapter 3

Methodology

The present chapter begins with a description of the system previously developed at CfAR for the detection and tracking of non-cooperative UAS, which will serve as the baseline for the proposed contributions. Subsequently, an overview of the proposed system is presented, followed by a detailed description of its components and the steps taken for their integration. Lastly, the limitations of the developed system are addressed.

3.1 Previous Work

The work presented in this thesis carries on the counter-UAV project developed at CfAR in the past two years, with main contributions from Daniel Justino [32] and Mariana Santos [33]. The previously developed system uses a LiDAR as well as a camera to detect, track and estimate the position of flying UAVs in the world coordinate frame.

3.1.1 Description

A diagram of the previous algorithm is illustrated in Figure 3.1. The system starts with LiDAR data: a point cloud which is first filtered and then clustered to detect possible flying objects nearby. This information is then used to create Regions of Interest (ROIs) on the images taken from the camera using 3D to 2D projection. YOLO then searches these regions, along with the ones taken from the tracker, for possible UAVs, outputting bounding boxes in the camera frame of reference. The tracker receives these detections, updates the tracks stored in memory and predicts the location of the UAVs in the following frame, thus creating ROIs to be used in the next iteration.

To convert these tracks back to the world coordinate frame, two methods are used, depending on the available information. First, if YOLO detects a UAV with corresponding LiDAR information (i.e. if there is a LiDAR 2D projection inside the bounding box returned by the detector), the point cloud is directly used to project the position back to the three dimensional space. Second, if there are no correspondent LiDAR points for the detected UAV, the position is estimated using only the results from the camera. To achieve this, a model of the apparent physical size of every tracked object is built on the first detection and updated every time LiDAR information is available. With this model of the target, the size of the bounding box provided by the detector and the geometric camera parameters, the 2D position in the camera frame of reference can be projected back to the three dimensional space, thereby allowing continuous reliable tracking of objects even in the absence of constant LiDAR data. Since position estimation based only on bounding box size is relatively noisy, Kalman filters are used to smooth out the results.

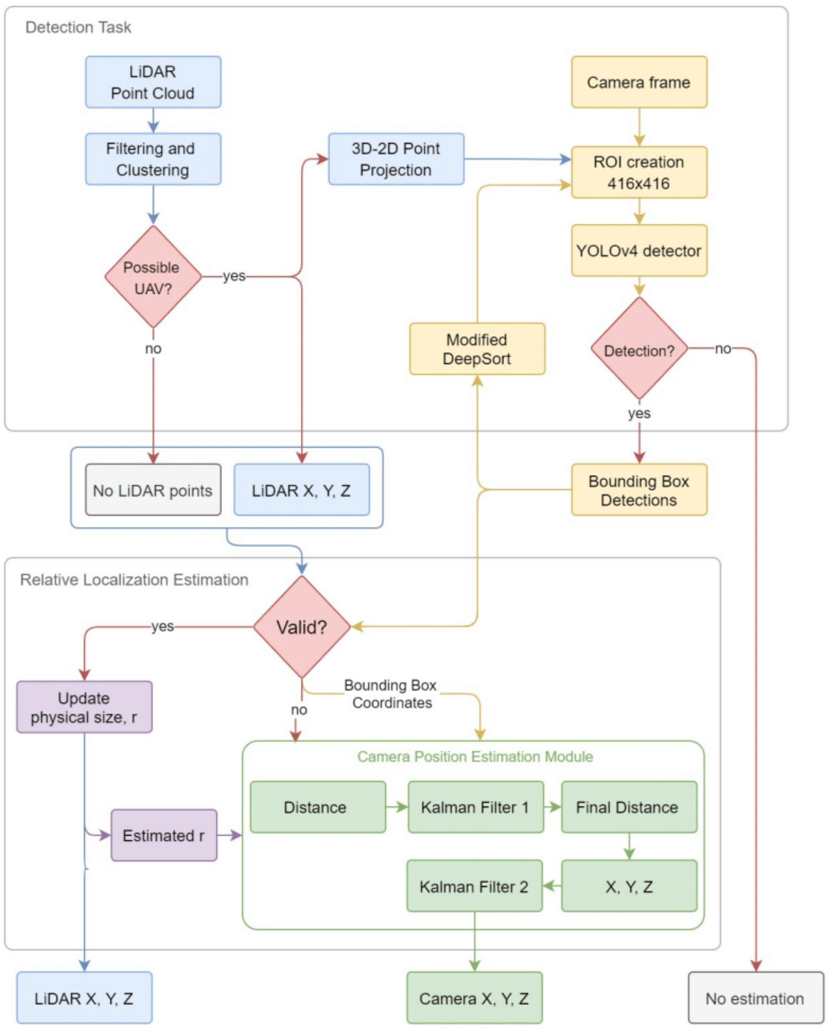


Figure 3.1: Diagram of the previously developed system [33].

The detector chosen for this system was YOLOv4 (the latest YOLO version at the time this algorithm was developed). As for the tracker, Deep SORT was used, however the Re-ID network was not included and the appearance descriptors were always left empty. This edited version, very similar to SORT, was called modified Deep SORT and used the cosine distance to measure the similarity between the bounding boxes from the detector and the tracker.

3.1.2 Limitations

One of the restrictions of this system is that the initialization solely relies on LiDAR data. This is a strong limitation given the fact that the LiDAR available has a small angular resolution in the elevation plane, resulting in a significant volume not covered by this sensor. It is for this reason that the estimation of the physical size of every object being tracked is crucial for the performance of the algorithm, allowing it to estimate distance even without continuous range information.

Another limitation is that the previously developed system does not take into account the camera's or the tracking vehicle's motion, especially, its angular motion: pan or tilt for the camera (or pitch or yaw for the aircraft). These rotations can significantly degrade the performance of the tracker, as the sudden apparent vertical and horizontal shifts in the videos will result in less accurate predictions, hindering the

ID assignment task of the tracker. This can also result in regions of interest that do not contain the targets, leading to lost tracks.

An additional limitation arises from the use of the RGB camera, as the nature of this sensor effectively makes night time tracking impossible, as the detector cannot output reliable detections with dark, featureless images as its input.

3.2 Proposed system overview

In order to tackle these limitations and improve the performance of the overall algorithm, some key changes are proposed. These include:

- Replacing the RGB camera with an IR camera;
- Using image alignment to compensate for camera motion;
- Providing the detector with moving object information by applying image subtraction to consecutive aligned frames;
- Upgrading the tracker from the modified Deep SORT to a new version of the ByteTrack algorithm.

The image detection and tracking subsystem of the overall project, colored in yellow in Figure 3.1, will therefore be completely reworked and taken as the full scope of this dissertation. A diagram of the proposed system is presented in Figure 3.2.

The new system discards the use of regions of interest, as the lower resolution of the infrared images means these can be entirely scanned without completely sacrificing inference speed. Abandoning ROIs also addresses the initialization problem by allowing the algorithm to create new tracks from the entire region captured by the camera.

Since the proposed system does not rely on regions of interest, it only needs the camera stream as its input, and this sequence of frames is used to generate tracks containing all the UAVs in its field of view. To achieve this, the system first runs the ECC algorithm to compute the camera motion matrix between the previous and the current frames, and, using this transformation, both frames are aligned and subtracted. The detector then receives a concatenation of the subtraction result with the current frame and scans this image for possible UAVs using YOLO.

After this step, the detection results are all converted to the same frame of reference, using the accumulated results from the ECC algorithm, and are then given as input to the tracker. For every frame, the tracker receives all the detections (including the ones with very small confidence scores), updates its tracks using the new information and outputs a list with the bounding box and ID of every UAV in the respective frame.

Note that this new subsystem can be integrated into the larger picture with minimal modifications to the pre-existing algorithm, however, this integration will be beyond the scope of this thesis.

3.3 Enhanced Correlation Coefficient (ECC)

The first major step in the proposed method is to compute the transformation matrix that aligns the previous frame with the current one. This is done using the ECC algorithm (described in section 2.2.1), which takes as input both frames and outputs the transformation matrix that can be applied to the previous frame to bring it to the frame of reference of the current camera position. This matrix can have different shapes depending on the selected transformation.

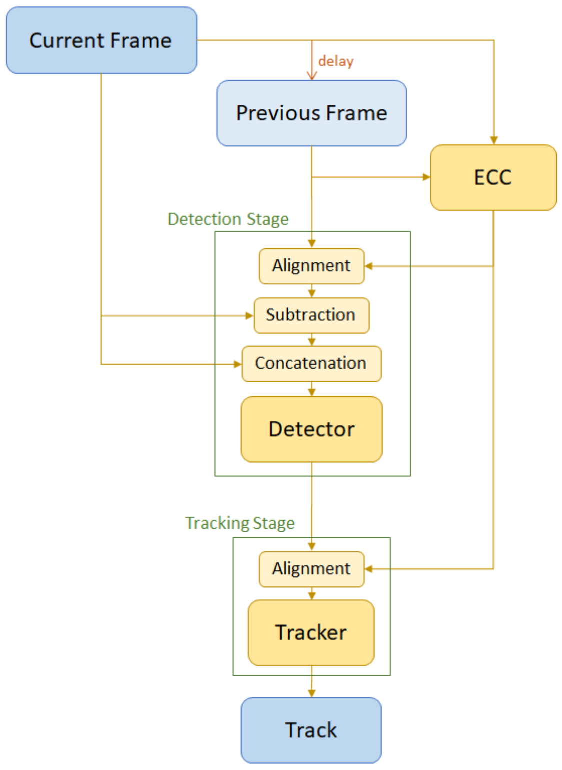


Figure 3.2: Diagram of the proposed system.

As presented in 2.2.1, there are two key hyper parameters necessary to guarantee a good performance of the ECC algorithm: the transformation to be estimated and the scale applied to its input images.

First, in order to select the transformation estimated by the ECC algorithm, an initial analysis of the runtime for the different transformations was conducted, and is presented in Table 3.1. This information was collected by applying a resolution scale of 0.05 to 1000 images from the dataset (with a 640×512 pixels resolution) and taking the average time per iteration.

Table 3.1: Average runtime of the ECC algorithm for different transformations.

Transformation	Runtime (ms)
Translation	0.96
Euclidean	1.70
Affine	2.60
Homography	4.41

The selected affine transformation was the translation for its faster inference speed when compared to the alternatives. This choice also takes into account the apparent movement of the background in the dataset, which didn't present significant rotation or changes in magnification.

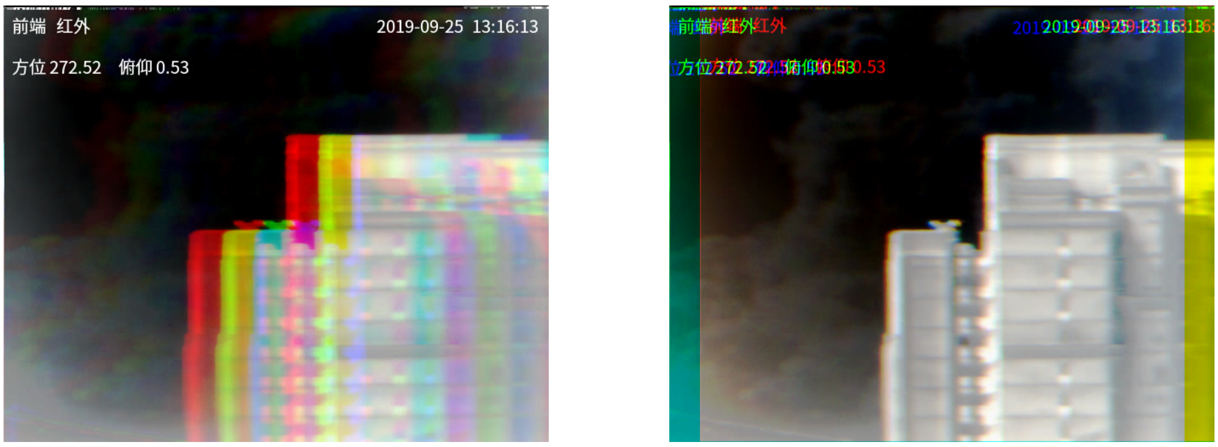
Note that the transformation used to change all the detections to the same frame of reference and the transformation used to align the images for subtraction was kept the same. This increases the overall efficiency of the algorithm since this transformation only needs to be computed once for every frame.

Note also that the runtimes presented in Table 3.1 only refer to the ECC algorithm. For image subtraction it is necessary to transform the images according to the respective warp matrix and for the tracker it is necessary to transform the bounding boxes returned by the detector to the inertial frame of reference.

As for the scale used to reduce the size of the ECC input images, there is an important trade-off to consider. A very small value considerably speeds up the runtime and improves the algorithm convergence, however, it also leads to a lot of information loss resulting in poor estimates. The choice of this parameter is thus vital to achieve good performance.

After some experimentation the scale was set to 0.05. This value ensures that the algorithm converges the vast majority of the times (around 99.99%), keeping the runtime low and still achieving good results on the videos from the Anti-UAV dataset (presented in Section 4.1).

A demonstration of image alignment using the ECC algorithm and the aforementioned parameters is presented in Figure 3.3. This was carried out by taking three consecutive frames from a video in the dataset and displaying them in the red, green and blue channels. In Figure 3.3b the first and third images were offsetted to the frame of reference of the second image.



(a) Before alignment.

(b) After alignment.

Figure 3.3: Demonstration of image alignment using 3 consecutive frames and the ECC algorithm (with a scale of 0.05 and the translation transformation).

Converting images to different frames of reference leads to unspecified borders, often padded with zeros. In Figure 3.3b it is possible to visualize this effect on the left and rightmost parts of the aligned image, where one of the channels is null as there wasn't enough information in the original images to fill these borders completely.

Another aspect to consider is that image alignment relies on having background visual features rich enough to be tracked across frames, which doesn't always happen on the Anti-UAV dataset. Looking into the dataset reveals that some of the videos were recorded with a clear sky as the entire background. This could degrade the performance, as the algorithm could use features from the UAVs for alignment, effectively mistaking drone movement for camera movement. In Figure 3.4 two different frames from the same video are presented, both having a cloudless sky as background.

In the detection stage, this behavior could deteriorate the subtraction results, however, this isn't expected to be a significant problem since the lack of background complexity also tends to make it easier for the detector to find the objects in the images. As for the impact of this effect on the tracker, taking a UAV as reference for the alignment will simply mean that the drone will appear to stay in place, resulting in a perfect position intersection across frames, actually simplifying the assignment process of the tracker. It is therefore expected that this behavior won't significantly impact the tracker results either.

Another factor that could deteriorate the ECC results are features in the images that don't change with camera position. These include characters added to the images, sensor imperfections, dirt, or even

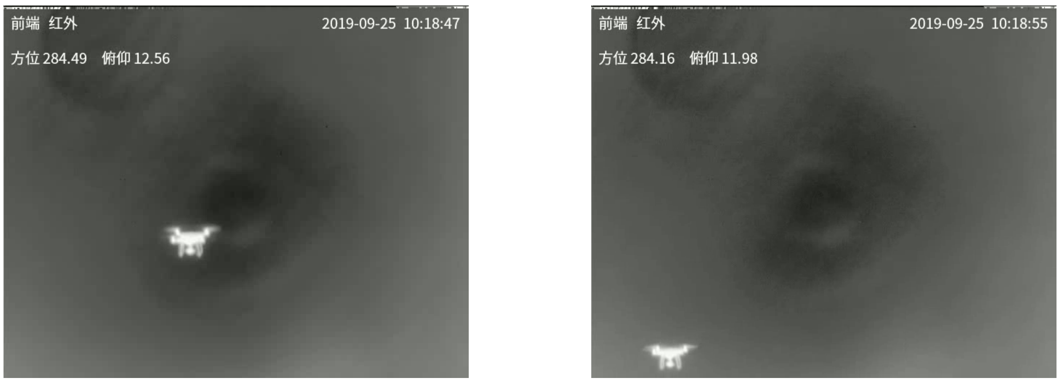


Figure 3.4: Different frames from the same video, taken from different camera attitudes, but with similar apparent backgrounds.

interference caused by the radiation emitted by camera's own optics [61]. In Figure 3.4 it is possible to visualize some of these effects.

To minimize this issue, the uppermost part of the images with the frame and time stamps was cropped out of the ECC algorithm, as this would degrade the alignment results (this procedure was always applied to the videos on this dataset).

Note that throughout this thesis document, single-channel images, including infrared images, are represented in grey-scale effectively having the same information in all three RGB channels.

3.4 Detector

The detectors chosen for this work are some of the latest versions of the popular YOLO series: YOLOR [45] and YOLOv7 [46] . According to [49], these detectors are the current state-of-the-art in real time object detection, thus, even though there are several detectors with better results on the COCO challenge, these were deemed the options with the best speed vs performance trade-off.

The authors of YOLOR present several models, compared in Table 2.1. Out of these, the YOLOR-CSP-X was selected for its cross stage partial connections, allowing it to be more compact (thus having higher image throughput), while maintaining a good accuracy. This architecture was also chosen for being more suited at scanning images with resolution around 640p, similar to the resolution of the images on the Anti-UAV dataset.

Some changes had to be made to its code available on github [62], to adapt its architecture to the intended application. These changes centered around changing the *config* file, which encoded the entire neural network in a human readable format. In this file, the number of classes was changed to one, and, in the final layers, the number of filters (or channels) was changed to 18, in accordance with equation (2.6). Note that the YOLOR architecture deploys three anchor boxes.

The authors of YOLOv7 (some of whom worked on the YOLOR project) also present several models, compared in Table 2.2. Out of these, the YOLOv7-X was chosen for this work, for identical reasons as the YOLOR-CSP-X. Note that YOLOv7 replaces the use of cross stage partial networks with its extend version of the ELAN computational block, E-ELAN, allowing for an even more efficient layer aggregation strategy. Similar to the procedure applied to YOLOR, the number of classes for this model was also changed to one.

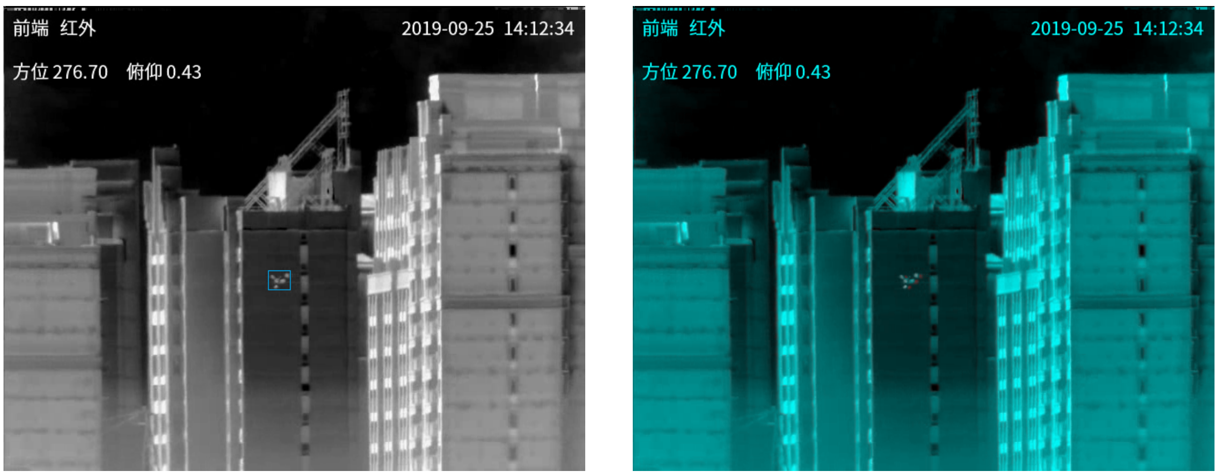
3.4.1 Image Alignment and Subtraction

Since the YOLO variants are already very intricate and optimized, the work developed focused on the input given to the detectors and not on the detectors themselves.

More specifically, to take advantage of the temporal information present in videos, instead of looking at every frame independently, information regarding moving objects was added to the input of the network. In practice, this information took the form of a new channel, concatenated with the original grey-scale image from the current frame to form the new input to the neural network. This extra channel contains the result of subtraction between the matrix with the current pixel values and the same matrix from the previous frame (aligned to the current frame of reference). The absolute value of this matrix was then taken, converting its values back to the 1 byte interval $[0; 255]$ standard in most image formats.

The subtraction information was concatenated with the current image to preserve its spacial relation, making sure any transformations applied to the dataset, like the image augmentation techniques used during training, are applied to all the information in the same way.

This implementation also benefits from the fact that most image formats store pictures in three or four channels, even the ones in grey-scale. It's for this reason, as well as to enable the use of the pre-trained weights, that the final tensors given as input to YOLOR and YOLOv7 had the current frame on the green and blue channels, and the subtraction information on the red one. In Figure 3.5 it is possible to visualize the original frame and the final input given to the network, side by side, and in Figure 3.6 the subtraction channel in presented (re-scaled for visualization purposes).



(a) Original image with respective ground truth bounding box.

(b) Result of concatenation between the original image (on the blue and green channels) and the result of subtraction (on the red channel).

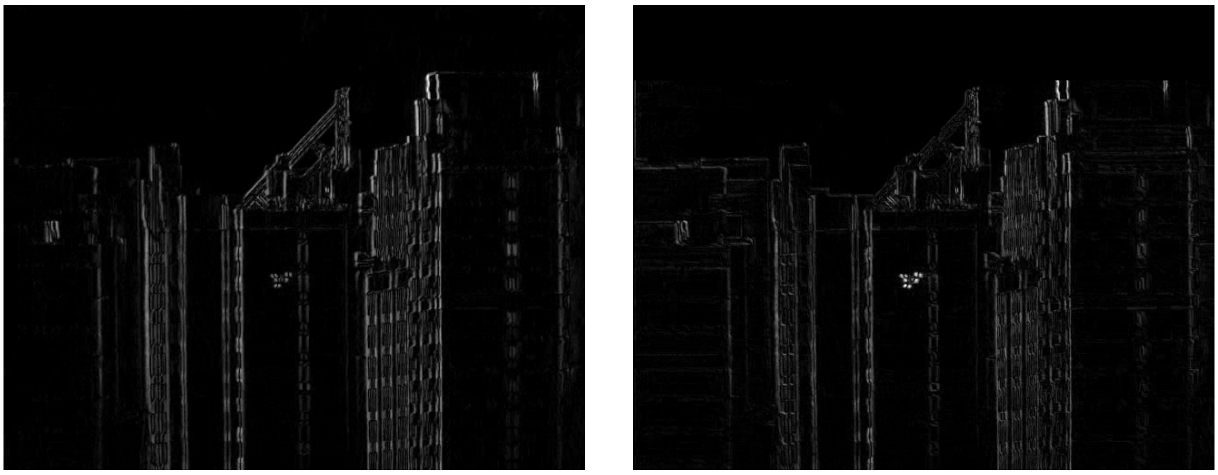
Figure 3.5: Images used to train the neural networks.

Note that the number of input channels of the network doesn't change the size of the channels (or filters) in the following layers, it has therefore a minimal impact on the total inference time.

As stated before, the previous image was aligned to the current frame of reference before the subtraction. The purpose of this alignment is to compensate for the camera motion between both frames, which would result in misaligned backgrounds and a significant apparent noise in the subtraction result. In Figure 3.6 is it possible to visualize the subtraction results with and without the use of frame alignment.

As evidenced by Figure 3.6, compensating for camera motion has a significant positive impact on the subtraction results, highlighting the real moving objects while fading out the edges present in the background.

As discussed in section 3.3, bringing two images to the same frame of reference leads to undeter-



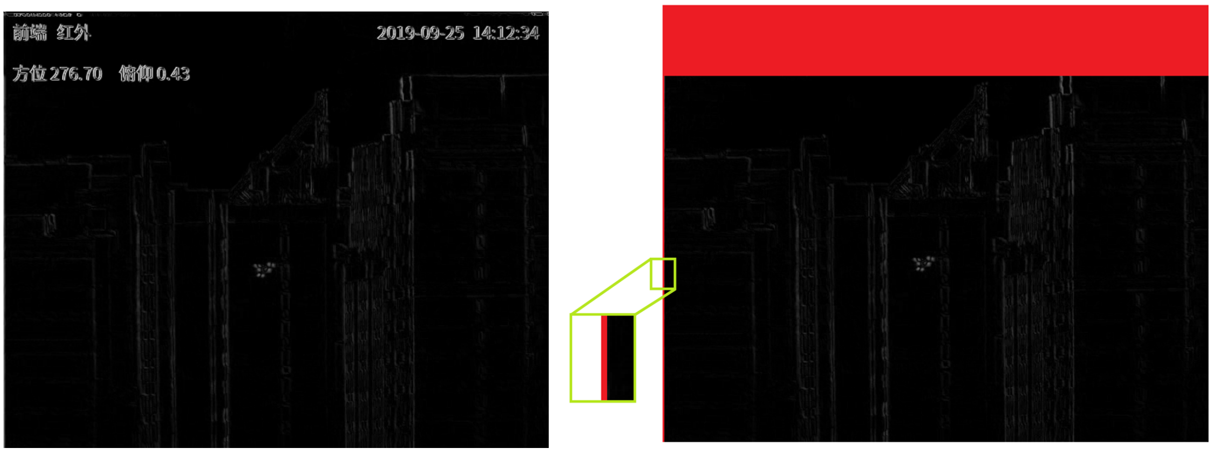
(a) Result of raw subtraction.

(b) Result of subtraction after alignment (padded).

Figure 3.6: Single-channel images containing the subtraction results between two consecutive frames. Note that, for visualization purposes, the pixel values of both images were linearly re-scaled to the interval [0,255].

mined borders on the transformed image, resulting in meaningless subtraction results for these regions. This effect can be suppressed by overriding part of the subtraction matrix with zeros in these positions.

This practice was implemented for all the subtraction results, and the padded area also included the topmost part of the frames containing the stamped time and video information (as these characters would become misaligned and disrupt the subtraction results). In Figure 3.7 it is possible to visualize a result of subtraction before padding, as well as the padded borders; note that this region was colored in red for visualization, however, it should be dark as presented in 3.6b.



(a) Before padding.

(b) After padding. For visualization purposes, the padded borders are colored in red and a section is zoomed in.

Figure 3.7: Result of padding the frame of an image. This time the raw result of subtraction is presented (without re-scaling).

3.4.1.1 Image/Video Format

One obstacle found when storing the results of image subtraction as a channel in the image, was the image/video compression algorithms used to store data more compactly.

Initially the videos were stored using the *mp4* container format, which was empirically shown to significantly degrade the subtraction information. Even the images themselves, used to train the detector, were saved in the *.jpg* format which is another lossy compression file format (meaning information is lost to achieve higher compression rates).

These compression formats altered the values on the red channel of the frames in a way nearly invisible to the naked eye (which is the goal of these algorithms), however, in order to provide the system with the most accurate data possible, all the information was stored and provided using the *png* format, a lossless compression file format.

Note that using the *png* file format increased the size of the dataset by a factor of approximately 5 when compared to the *jpg* file format.

3.4.2 Training

For the process of training the models, first, a new set of videos was created from the original Anti-UAV dataset. The adopted procedure followed the above mentioned methodology and resulted in a replica of the original dataset, except for one of the channels on all the images, which now contained the subtraction results.

Both sets of videos were then converted to images and split into three sets: training, validation and testing, in a respective 0.6, 0.2, 0.2 ratio. Since there are 100 videos in the original dataset, this resulted in: 60 videos for training (56072 images), 20 for validation (19028) and 20 for testing (18147 images) in both models.

Due to the high GPU and memory resources such a large datasets would take to train, only 25% of the training and validation datasets were actually used to train the network. This effectively resulted in two training sets with 14030 images and validation sets with 4730 images. Since there is a very strong correlation between consecutive frames in the videos, taking only one out of every four frames should still preserve most of the diversity present in both these sets.

All the models were trained for 100 epochs starting from the weights provided by the authors (pre-trained on the COCO dataset), which achieved the results presented in Tables 2.1 and 2.2.

3.4.2.1 Image augmentation

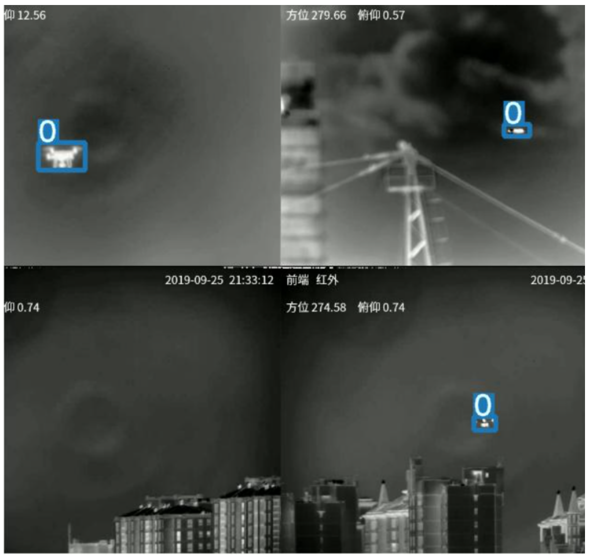
Another factor to take into account is the image augmentation procedures most state-of-the-art detectors, including YOLOR and YOLOv7, use during training . These transformations can be split into two groups: one that changes the intensity of the pixels (i.e. changes in hue, saturation and value) and another that simply changes the position of the pixels in the image (i.e. translation, scale, horizontal flips and mosaic patterns).

Since this practice has been proven to improve the performance of the models, both kinds of transformations were used. In Figure 3.8, it is possible to visualize one example of the image augmentation used by YOLOR during training (using a batch size of four). In this example, four sections were cropped out of their original frames and stitched together to form an augmented image.

3.5 Tracker

For the proposed system, it is necessary for the tracker to be fast and online, in order to be used in real-time applications. Three trackers meet these prerequisites: SORT; the modified version of Deep SORT (used in the previous work), and ByteTrack, the current state-of-the-art in multi-object tracking.

Algorithms with deep association metrics, like the original Deep SORT, were not considered for their significant computing power requirements, and because drones (especially if seen from a far) lack the


(a) By the baseline model.

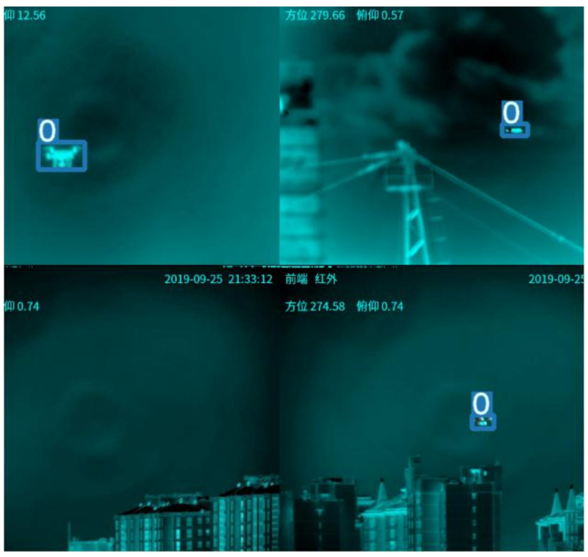

(b) By the proposed model.

Figure 3.8: Augmented images used during training.

necessary visual features for this approach to justify the added complexity. This lack of visual features is even more noticeable in the infrared images, as their single channel nature denies the Re-ID network the necessary colour intricacy expressed in RGB images.

In order to adapt the ByteTrack algorithm to the developed system, some changes were made to its code. First, the original implementation defined a minimum bounding box area of 100 pixels and a maximum aspect ratio for the bounding boxes equal to $width/height = 1.6$. Both these restrictions served to filter out undesirable bounding boxes, improving the results on the pedestrian dataset where it was evaluated (i.e. the MOT17 dataset). On the problem at hand, UAVs tend to appear as smaller objects and with larger aspect ratios than pedestrians, which led to the removal of these constraints.

Apart from these, two other changes are proposed to the ByteTrack algorithm: detection alignment and a new similarity metric for bounding box assignment.

3.5.1 Detection alignment

A brief analysis of the Anti-UAV dataset reveals a considerable amount of motion present in its videos. This motion is not continuous, the camera experiences periods when it is almost completely still, followed by sudden quick movements.

The angular shifts in the camera's attitude, which appear as horizontal and vertical translations in the videos, can completely disrupt the results of the Kalman filter, as well as its predictions, by introducing a volatile component to the every target's apparent speed. Without reliable Kalman filter estimates, it it far more difficult for the tracker to assign the new detections to the right tracks in memory, since the detector's results can be far from the filter's projections.

In order to tackle this problem, the apparent shift in the videos is estimated and used to convert the results from the detector to a single frame of reference, invariable to video shifts. To achieve this, a matrix with the accumulated affine transformation (since the first frame) is computed by applying the ECC algorithm to every frame between the current and the previous images and adding this result to the overall transformation. The inverse of this transformation is then applied to every new detection, effectively converting it to the frame of reference of the first image in the respective video. Doing this ensures that all the inputs to the tracker share the same frame of reference and its results are safeguarded from camera motion.

3.5.2 New Similarity Metric for Bounding Box Assignment

As presented in 2.4.3, the original implementation of ByteTrack used IoU to assign the new detections to the tracks in memory. This poses a problem when dealing with small, fast moving targets, as the IoU can easily reach very small values, or, in the worst case, zero.

As explained in section 2.5.1, the IoU metric is commonly used to measure the accuracy of object detectors, by comparing the similarity between the predicted bounding boxes and their correspondent ground truth. It is therefore applied between annotations and inference data, where a detection is only acceptable if it contains, at least, part of the object.

In the context of multi-object tracking, the situation changes, and now the similarity is used to compare the detections with the expected positions of the targets, predicted by the Kalman filter, which can have very small or even null intersections. In the Anti-UAV dataset, the changes in angular position of the camera, paired with the motion of the UAVs, results in significant object motion throughout the videos, which deteriorates the accuracy of the tracker predictions, making the assignment process of the tracker more difficult.

This problem could be minimized by decreasing the IoU threshold used in the assignment to values closer to zero, allowing for easier matching. However, this comes at and unwanted cost: a large bounding box could be matched with any of the smaller bounding boxes inside it (e.g. a detection with the size of the entire image could me matched to any of the objects being tracked). This "solution" would also have no effect on bounding boxes belonging to the same object which have no intersection, since the IoU would always be zero regardless of how similar the bounding boxes were.

In an attempt to solve the problem of the IoU being null if there is no intersection, regardless of the distance and size of the bounding boxes, a new similarity metric is proposed. Describing every bounding box by the coordinates of its center (x, y), its width (w) and its height (h); and using the subscript d to refer to the detections and t to the tracker predictions for the current frame, the new similarity metric is defined as:

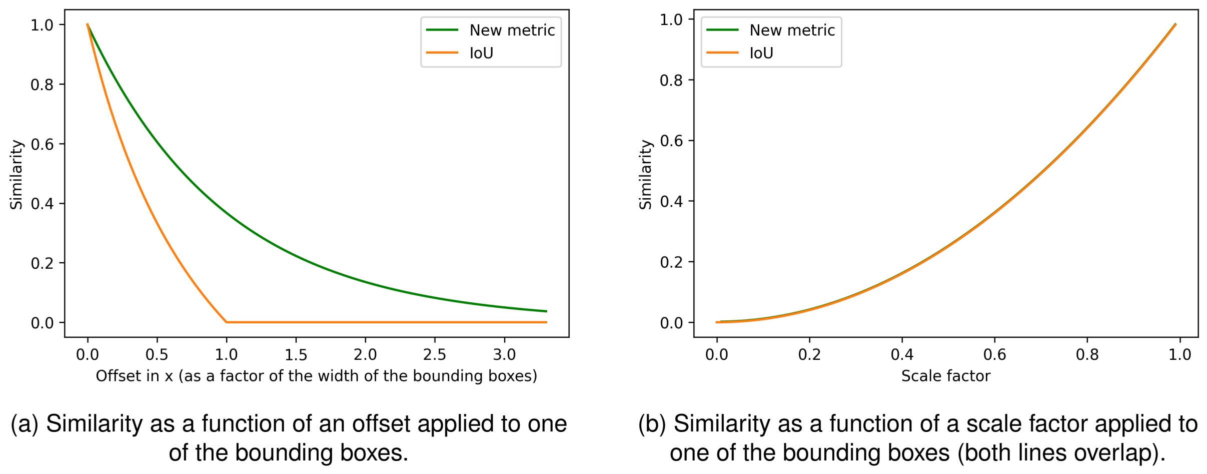
$$\begin{aligned} distance &= \left(\left(\frac{x_d - x_t}{w_t}\right)^2 + \left(\frac{y_d - y_t}{h_t}\right)^2\right)^{\frac{1}{2}}, \\ similarity &= e^{-distance} \cdot \frac{min(w_d \cdot h_d, w_t \cdot h_t)}{max(w_d \cdot h_d, w_t \cdot h_t)}. \end{aligned} \tag{3.1}$$

Equation (3.1) starts by calculating the distance between the centers of the two bounding boxes, relative to the size of the trackers bounding box. The exponential of the symmetric of this value is then used to transform the distance to the interval $]0, 1]$, where one corresponds to a perfect match in position and zero corresponds to an infinite distance between the two boxes. This result is then multiplied by the quotient between the area of the smaller bounding box and the area of the larger bounding box, again, keeping the overall result as a number between zero and one. This multiplication is necessary for the similarity metric to take into account, not only the relative position the the bounding boxes, but also their relative size.

The euclidean distance between the bounding boxes is taken relative to the size of the bounding box of the tracker, since this is the result of all the bounding boxes ever assigned to that object, filtered by a Kalman filter, making it a less noisy estimate of the actual object size, projected to the camera's 2D space.

Keeping the similarity metric in the interval $]0, 1]$ (where one corresponds to a perfect match, and zero to the worst possible match) was desirable to make the choice for the matching threshold easier and more intuitive, as well as to keep consistency between the new metric and the IoU for future comparisons in chapter 4. This choice is also more compatible with the algorithms used in most trackers.

A comparison between the IoU and the new assignment metric is presented in Figure 3.9. This comparison consists on starting with two identical bounding boxes and measuring their similarity as a transformation is applied to one of them: offset (Figure 3.9a) or scale (Figure 3.9b).



(a) Similarity as a function of an offset applied to one of the bounding boxes.

(b) Similarity as a function of a scale factor applied to one of the bounding boxes (both lines overlap).

Figure 3.9: Comparison between the new assignment metric and the intersection over union.

Note that for the same center position, changing the scale of one of the bounding boxes will yield the same result for both the similarity metrics, i.e. the quotient of the two areas (a quadratic function of the scale factor applied to the width and the height of one of the boxes).

3.6 Method Limitations

It is also important to understand some of the limitations of the presented algorithm.

First, the new assignment similarity can be used to relax the assignment constraints, improving the tracking results when there is a relatively small number of objects being tracked. This can however have a negative impact in performance if there are plenty of targets in the video, specially if their paths overlap. The new similarity metric is therefore presented as a way to have more control over the matching threshold and not a complete solution to the ID assignment problem in multi-object tracking.

Second, the added complexity of the proposed algorithm comes at a cost: slower inference speed. The tracker, alignment and subtraction all take a non-negligible amount of time to run which can limit their use in a real life scenario, especially if the algorithm has to run on an edge device (i.e. a CPU or Graphics Processing Unit (GPU) aboard the patrolling UAV). A more comprehensive analysis of the impact of these new features in the overall inference time is carried out in chapter 4.

Third, image subtraction has some inherent limitations caused by non-static backgrounds and/or non-static cameras. Environments with a significant amount of motion caused by vehicles, waves, wind, etc, as well as blurry frames, will result in noisy subtraction results, degrading the overall reliability of the algorithm.

Lastly, infrared cameras tend to be harder to acquire and to have lower resolution that their RGB counter-parts. The images taken from these cameras also tend to be single channel, making them less feature rich, which can cause other flying objects with similar thermal signatures, like birds, helicopters and planes to be mistaken for UAVs. This can also make detection more difficult when working with complex backgrounds.

Chapter 4

Results

The results presented in the previous work were collected using AirSim (an open-source simulator that enables the development and testing of algorithms for autonomous vehicles). This simulator, built upon the Unreal Engine game engine, can render realistic images in real-time, taking advantage of the capabilities provided by dedicated graphics cards. The renderer used in AirSim can only generate RGB images, unsuitable to test the algorithm proposed in this work. For this reason, and since the scope of this dissertation addresses exclusively the detection and tracking part of the existing project, the collection of results was limited to the Anti-UAV dataset, presented bellow.

Real-life tests, even though possible using the TASE200 available at CfAR, were unpractical due to the use of a restricted access, military grade camera, as well as time constraints.

4.1 Anti-UAV Challenge Dataset

The 1st Anti-UAV Workshop and Challenge [63], which took place in Seattle, Washington, between the 14th and 19th of June 2020 resulted in the release of the original Anti-UAV dataset.

The labeled part of this dataset is comprised of 100 fully-annotated RGB and IR unaligned videos, intended to provide a realistic benchmark for object tracking algorithms in the context of drone detection. It contains recordings of six UAV models flying at different lighting and background conditions. In Figure 4.1 some examples of frames taken from these videos are presented.

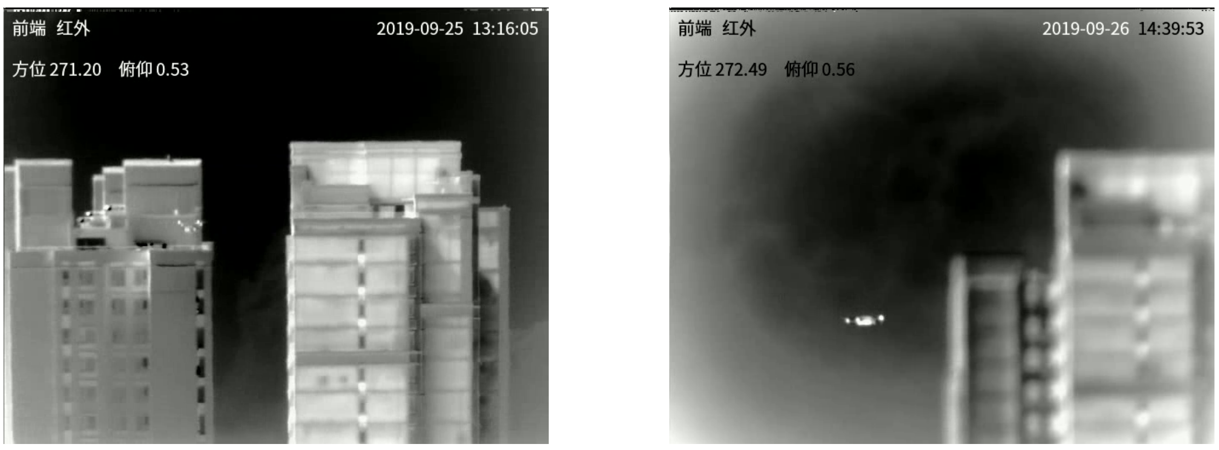


Figure 4.1: Images from the Anti-UAV dataset [63].

4.2 Detector

Both YOLOR and YOLOv7 were trained for 100 epochs using weights pre-trained on the MS COCO dataset [47] and using two sets of data obtained from the Anti-UAV dataset: one with the concatenated subtraction information and one without it.

The weights with the best performance on the validation set were then used to test the models. Detectors usually define the best weights through a weighted combination of different metrics. Both YOLOR and YOLOv7 use the same definition for the best weights: the weights that yield the best combination of mAP@0.5 and mAP@0.5:0.95 on the validation set, using proportions of 0.1 and 0.9, respectively.

The results obtained using the best YOLOR and YOLOv7 weights on the test set are presented in Table 4.1.

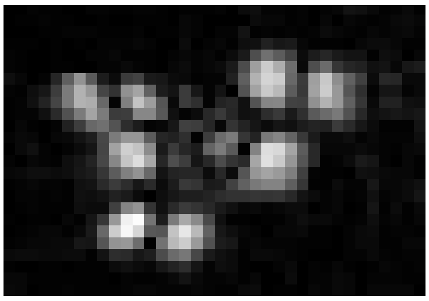
Table 4.1: YOLOR and YOLOv7 test results.

Detector	Data	P	R	mAP@0.5	mAP@[.5:.95]
YOLOR	Without subtraction	0.919	0.963	0.958	**0.468**
	With subtraction	**0.928**	**0.966**	**0.960**	0.462
YOLOv7	Without subtraction	0.913	0.951	0.929	**0.436**
	With subtraction	**0.914**	**0.953**	**0.937**	0.432

These results indicate a good performance of both detectors, with Precision (P), Recall (R) and mean Average Precision using an IoU threshold of 0.5 (mAP@0.5) above 0.9. Increasing the IoU threshold significantly decreases the mAP, as expected, since stricter IoU requirements leads to more false positives and false negatives (also called misses), resulting in a lower mean Average Precision.

Comparing the models with and without subtraction information, it is possible to see a trend in both detectors. Using YOLOR and YOLOv7 the models with subtraction information outperform the models without subtraction information in precision, recall and mAP@0.5, only under performing in the mAP@[.5:.95] metric. This means that the models with subtraction information are overall better at detecting the drones present in the videos leading to fewer false positives and misses, however, the bounding boxes returned are not as tight around the objects leading to poorer IoU between the detections and the ground truth. This inferior IoU matching could be explained by the nature of the subtraction information. As an object moves from frame to frame, subtracting consecutive frames will result in two active areas: one where the object is now and wasn't in the previous frame, but also one where the object isn't now and was in the previous frame. In Figure 4.2b it is possible to visualize these 2 areas, one in white (where the drone is in the current frame) and the other in red (where the drone was in the previous frame). This additional information may be the cause for the lower mAP@[.5:.95], as the subtraction channel might extend the bounding boxes to include not just the object's current location, but also part of the previous one.

Finally, from Table 4.1 it is also possible to conclude that the both detectors yield similar results, with YOLOR having slightly better scores in the relevant metrics.



(a) Result of subtraction, cropped from figure 3.6b.

(b) Result of concatenation between the original infrared image and the result of subtraction, cropped from figure 3.5b.

Figure 4.2: Subtraction information.

4.3 Tracker

To compare the performance of the trackers, these were also evaluated on the test set, comprised of 20 videos - 18147 frames and 17784 objects. The results from this test are presented in Table 4.2.

Table 4.2: Test results using different trackers.

Tracker	IDF1	MOTA	FP	FN	IDS
SORT	0.813	0.950	**40**	819	37
Mod. Deep SORT	0.812	0.969	437	**68**	39
ByteTrack	0.897	0.979	249	113	20
New ByteTrack	**0.929**	**0.980**	249	101	**12**

In this table Mod. Deep SORT refers to the modified version of Deep SORT (without the appearance descriptors); New Bytetrack refers to the Bytetrack using the new assignment metric and FP, FN and IDS refer to false positives, false negatives and ID switches, respectively.

From the results in Table 4.2 it is possible to see that the SORT and Modified Deep SORT present similar performance, trading off false positives for misses. Both the original and the new ByteTrack are able to outperform these trackers, as making use of all bounding boxes, even the ones with very low confidence scores, allows for these trackers to follow the UAVs more consistently.

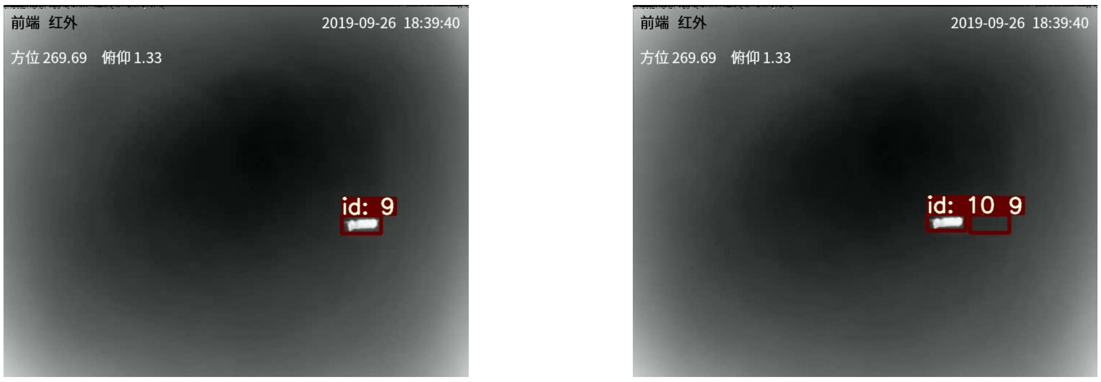
Comparing both versions of the ByteTrack algorithm, it is possible to see a clear improvement in performance as the new assignment metric leads to better matching resulting in fewer misses and ID switches.

It is true that the New Bytetrack is benefiting from looser constraints, allowing the Kalman filter predictions to match more easily with the current detections. To check whether the scores from Table 4.2 are simply a result of using less strict constraints, the IoU thresholds on SORT and Bytetrack were set to the value of 0.01, allowing for simpler matches between the predictions and the detections. The results from this experiment are presented in Table 4.3.

From Table 4.3 it is possible to conclude that lowering the IoU threshold does have a positive impact on both SORT and Bytetrack. This is not enough to outperform the New ByteTrack though, as decreasing the IoU threshold to very small values leads to a limit for false negatives and IDS. This arises from the fact that loosening IoU constraints has no effect on bounding boxes belonging to the same object which have no intersection, like the ones presented in Figure 4.3, where the bounding box predicted by the Kalman filter and the detected bounding box do not overlap, leading to an unavoidable FP and IDS.

Table 4.3: Effect of decreasing the matching threshold on SORT and Bytetrack.

Tracker	IDF1	MOTA	FP	FN	IDS
SORT	0.910	0.953	**40**	774	19
ByteTrack	0.927	0.979	249	110	18
New ByteTrack	**0.929**	**0.980**	249	**101**	**12**



(a) Previous frame.

(b) Current frame

Figure 4.3: Tracker results on two consecutive frames. In this example, the null IoU leads to a track being lost and another being generated.

4.3.1 Image subtraction

To test the effect of using the image subtraction information on a tracker, new data was collected. This data, presented in Table 4.4, was obtained using the new version of the ByteTrack, with the two different detectors (YOLOR and YOLOv7) trained with the two different sets of data (with and without the subtraction channel).

Table 4.4: New ByteTrack results.

Detector	Data	Validation		Test	
		IDF1	MOTA	IDF1	MOTA
YOLOR	Without subtraction	0.980	0.988	0.929	0.980
	With subtraction	**0.982**	**0.989**	0.929	0.980
YOLOv7	Without subtraction	0.967	0.984	0.934	0.981
	With subtraction	**0.981**	**0.989**	**0.960**	**0.991**

From the results in Table 4.4, it is possible to verify that, in accordance with the results presented for the detectors (in section 4.2), the subtraction information does have a positive impact on the tracking results, improving both the IDF1 and MOTA scores of the respective models, with the exception of the YOLOR model on the test set. In fact, the model that performs the best on the test set uses the subtraction information, as well as the YOLOv7 detector, to achieve an impressive IDF1 score of 0.960 and a MOTA of 0.991.

4.3.2 Detection Alignment

To measure the effect of using the ECC results to align the detections before sending them to the tracker, test results were collected with and without this alignment. These results are presented in Table 4.5 and were obtained for both the New ByteTrack and SORT trackers.

Table 4.5: Effect of aligning detections.

Tracker	IDF1		MOTA	
	with alignment	without alignment	with alignment	without alignment
SORT	0.813	0.637 (-21.65%)	0.950	0.932 (-1.90%)
New ByteTrack	0.929	0.921 (-0.86%)	0.980	0.979 (-0.1%)

From these results it is possible to conclude that aligning the detections does have a positive impact on the results from the algorithm, however, for the most part, the trackers can still assign the detections belonging to the same objects to the corresponding track even with the added noise resulting from the shifts in the camera's position.

This improvement is particularly small using the New Bytetrack as the new assignment metric is more forgiving, allowing for easier matching. Using the SORT algorithm, a small camera displacement can significantly reduce the IoU between the expected and the measure positions of an object, making the improvement of aligning detections especially impactful for this tracker.

Note that ID switches are more heavily penalized by the IDF1 metric than by the MOTA, since one IDS will probably lead to many ID False Positives (IDFPs) and ID False Negatives (IDFNs).

Even though the alignment step doesn't boost the performance of the algorithm by a large amount, the cost associated with it is very small. If the ECC results are available, aligning the detections takes two or four summations (depending on the format of the bounding boxes), negligible when comparing to the millions of operations performed by the YOLO algorithm.

4.3.3 Inference time

Finally, to check the impact of all the steps on the total runtime of the algorithm, the partial times were measured. These results are presented in Table 4.6 and were collected using the New ByteTrack tracker and a single graphics processing unit, the NVIDIA GeForce RTX 2070 SUPER. Since ByteTrack, SORT and the modified version of Deep SORT all share the same principles and code base, with the main difference being thresholds and the assignment metric, the runtimes for these three trackers is very similar.

In this table, the term *Inference* refers to the execution of the YOLO algorithm, including both the CNN and the NMS. The term *Subtraction* is used to describe both the transformation applied to the previous frame to bring it to the current frame of reference, i.e. the translation, as well as the actual frame differencing. Of the Subtraction runtime, around 65% is used for the transformation and 35% for the image differencing.

These results corroborate the fast inference speed proposed by the authors of the YOLOv7 algorithm, which takes 53% of the time to run compared to YOLOR. These results also confirm the relatively low impact of performing the alignment and subtraction on the overall runtime, as the detector is still responsible for around 90% of the total execution time.

The final system can be considered real time, with a frame throughput of 16.85 frames per second using the YOLOR detector and 29.4 Frames Per Second (FPS) using YOLOv7 .

Table 4.6: Algorithm's partial runtimes, in milliseconds. The percentage relative to the total is also presented in parenthesis.

Detector	Inference		ECC		Tracker		Subtraction		Total
YOLOR	55.71	(93%)	0.97	(1%)	1.17	(2%)	2.33	(4 %)	59.35
YOLOv7	29.56	(87 %)		(3 %)		(3 %)		(7 %)	34.03

4.3.4 Limitations

Even though the above mentioned models achieved very satisfactory results on the validation and test sets, there are a few key limitations necessary to point out.

The main one arises from the lack of diversity on the Anti-UAV dataset, which results in high validation and test scores, however, that doesn't mean these models will generalize well to data from other datasets. Dynamic background features, such as moving trees and water, will introduce a substantial amount of subtraction noise which the models are not trained to ignore. The drones themselves were also relatively similar, close to the camera and often had a clear sky as background, which simplified the task of their detection and tracking.

Another disadvantage of using a relatively simple dataset, is that it makes all improvements seem small, as even simple solutions can achieve good performances. This was evident when comparing different models approaches, which often presented similar results.

Chapter 5

Conclusions and Future Work

This chapter presents the main conclusions of the work developed in this thesis. It also puts forth its contributions and suggests ideas for future research that can improve the system and its capabilities.

5.1 Conclusions

This thesis proposes a system for the detection and tracking of non-cooperative UAVs using a deep learning approach and a continuous stream of images from an infrared camera. For every frame, this system outputs a set of bounding boxes and their IDs, corresponding to the UAVs captured by the field of view of the camera.

Every iteration of the algorithm starts with two consecutive frames, which are aligned and subtracted, resulting in a representation of the every object motion between the frames. Since this representation shares the same spacial orientation as the original image, they are both concatenated and given as input to a state-of-the-art detector: YOLOR or YOLOv7. The detector scans the images for possible UAVs and provides the system with a list of possible locations (i.e. bounding boxes) and confidence scores. These locations are first aligned to compensate for camera motion and then given to the tracker, paired with the confidence scores. The tracker connects the sequence of detections belonging to the same objects, assigning a unique ID to each one, and rejects all the detections which do not match to any object being tracked nor have enough confidence to start their own new track.

Experiments conducted using the Anti-UAV dataset analyse the impact of all the steps on the overall system and demonstrate the incremental improvements achieved at every stage. More specifically, these tests show that including the subtraction information does improve the detector ability to find the objects of interest in the images, leading to higher precision, recall and mAP@0.5 scores. They also confirm the improvement in performance gained by aligning the detections and by replacing the assignment metric in the tracker by a less strict version of the IoU .

The system with the best performance uses the ByteTrack tracker, with the new assignment metric, and the YOLOv7 detector to achieve a high IDF1 score of 0.960 and a MOTA score of 0.991 on a test subset created from the Anti-UAV dataset. Furthermore, the system achieves real-time capabilities, running on a single GPU (an NVIDIA GeForce RTX 2070 SUPER), at a frame throughput of 29.4 frames per second, significantly higher than the frame rate of the videos on the dataset, filmed at 20 frames per second.

5.2 Contributions

This thesis presents a novel approach to incorporate image subtraction information into a state-of-the-art detector, allowing it to see, not only the objects contained in an image, but also their apparent movement.

A comparison between some of the state-of-the-art detectors and trackers is also presented, providing more insight into their performance and applicability, particularly for the task of tracking small, moving objects. In the end, the state-of-the-art detector (YOLOv7) and tracker (ByteTrack) were integrated, tested and evaluated.

A change to the ByteTrack tracker is also proposed, especially useful in scenarios were a low IoU between detections and tracker predictions is to be expected. This new metric was tested and compared to the original, achieving better results by facilitating the matching process.

The impact of aligning detections before providing them to the tracker was also tested, with the results suggesting a small but consistent improvement.

Finally, the efficacy of using thermographic videos to detect small UAS was confirmed and the real-time capabilities of a detector + tracker set up capable of reliably tracking UAVs were validated, with a system running at nearly 30 FPS on a single GPU.

5.3 Future work

Potential future work includes the use of a more extensive and diverse dataset, that could train a model with a clearer understanding of the nature of the subtraction information, allowing it to more reliably interpret new data. This could be achieved by using an extensive RGB video dataset, such as the MOT17 dataset, used to benchmark multi-object trackers.

The subtracting procedure presented in the methodology chapter is easily generalizable to three channels (RGB) and to the tracking of other classes of objects such as cars and pedestrians. For instance, the new subtraction channel could be generated by the norm of the distance in the red, green and blue intensities for each pixel, effectively achieving the same concept as the subtraction used in this work.

On another note, the way in which motion information is given to the the detector (using a new channel for the image), is not very coherent with how YOLO and other CNNs work. These networks start by looking for low level features (like edges and corners) and move their way into more complex features like shapes and eventually objects. The subtraction information is not only noisy, but also lacks the patterns most convolution filters are trained to recognize. Given the nature of the subtraction information (intended to highlight the objects in motion) it would perhaps make more sense to apply it to the original image using an attention mechanism, in a transformer based approach.

Finally, to improve the synergy between the detector and the tracker, achieving the best balance between the number of misses and false positives, a new study could be conducted to test which metric should the detector optimize during training (e.g AP@50, precision, recall, etc) in order to achieve the best tracking results.

Bibliography

[1] V. U. Castrillo, A. Manco, D. Pascarella, and G. Gigante, “A Review of Counter-UAS Technologies for Cooperative Defensive Teams of Drones,” *Drones*, vol. 6, no. 3, 2022.

[2] U.S. Department of Transportation, “Federal Aviation Administration: Drones by the Numbers.” `https://www.faa.gov/uas/resources/by_the_numbers/`. Accessed: 8.07.2022.

[3] The Guardian, “The mystery of the Gatwick drone.” `https://www.theguardian.com/uk-news/2020/dec/01/the-mystery-of-the-gatwick-drone`. Accessed: 8.07.2022.

[4] The Guardian, “Venezuela’s Nicolás Maduro survives apparent assassination attempt.” `https://www.theguardian.com/world/2018/aug/04/nicolas-maduros-speech-cut-short-while-soldiers-scatter`. Accessed: 8.07.2022.

[5] M. Burgess, “Small Drones Are Giving Ukraine an Unprecedented Edge.” `https://www.wired.co.uk/article/drones-russia-ukraine-war`. Accessed: 27.10.2022.

[6] D. Hambling, “How Ukraine Perfected The Small Anti-Tank Drone.” `https://www.forbes.com/sites/davidhambling/2022/06/01/how-ukraine-perfected-the-small-anti-tank-drone/?sh=49ab20b73171`. Accessed: 27.10.2022.

[7] H.-W. Warnke, “Reconnaissance of LSS-UAS with Focus on EO-Sensors,” in *Sensors and Electronics Technology (SET) Panel Symposium*, 2017.

[8] NATO Communications and Information Agency, “NATO Agency holds exercise to improve counter-drone technology.” `https://www.ncia.nato.int/about-us/newsroom/nato-agency-holds-exercise-to-improve-counterdrone-technology.html`. Accessed: 9.07.2022.

[9] Y. Wang, Y. Chen, J. Choi, and C.-C. J. Kuo, “Towards Visible and Thermal Drone Monitoring with Convolutional Neural Networks,” *APSIPA Transactions on Signal and Information Processing*, vol. 8, p. e5, 2019.

[10] H. Kiani Galoogahi, A. Fagg, and S. Lucey, “Learning Background-Aware Correlation Filters for Visual Tracking,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] S. Wu, K. Zhang, S. Li, and J. Yan, “Joint feature embedding learning and correlation filters for aircraft tracking with infrared imagery,” *Neurocomputing*, vol. 450, pp. 104–118, 2021.

[12] D. Xing, A. Tsoukalas, N. Giakoumidis, and A. Tzes, “Computationally Efficient RGB-T UAV Detection and Tracking System,” in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1410–1415, 2021.

[13] B. Shuai, A. Berneshawi, X. Li, D. Modolo, and J. Tighe, “Siammot: Siamese multi-object tracking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12372–12382, 2021.

[14] D. Xing, N. Evangeliou, A. Tsoukalas, and A. Tzes, “Siamese Transformer Pyramid Networks for Real-Time UAV Tracking,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2139–2148, 2022.

[15] B. K. Isaac-Medina, M. Poyser, D. Organisciak, C. G. Willcocks, T. P. Breckon, and H. P. Shum, “Unmanned aerial vehicle visual detection and tracking using deep neural networks: A performance benchmark,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1223–1232, 2021.

[16] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot MultiBox detector,” in *European Conference on Computer Vision*, pp. 21–37, Springer, 2016.

[18] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv:1804.02767*, 2018.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision*, pp. 213–229, Springer, 2020.

[20] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.

[21] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *International Conference on Image Processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[22] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, “Tracking Without Bells and Whistles,” in *International Conference on Computer Vision (ICCV)*, IEEE, 2019.

[23] R. Ippalapally, S. H. Mudumba, M. Adkay, and N. V. H. R., “Object detection using thermal imaging,” in *IEEE 17th India Council International Conference (INDICON)*, pp. 1–6, 2020.

[24] A. Banuls, A. Mandow, R. Vázquez-Martín, J. Morales, and A. García-Cerezo, “Object detection from thermal infrared and visible light cameras in search and rescue scenes,” in *International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 380–386, IEEE, 2020.

[25] P. Shaniya, G. Jati, M. R. Alhamidi, W. Caesarendra, and W. Jatmiko, “YOLOv4 RGBT Human Detection on Unmanned Aerial Vehicle Perspective,” in *2021 6th International Workshop on Big Data and Information Security (IWBIS)*, pp. 41–46, 2021.

[26] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv:2004.10934*, 2020.

[27] Y. Wang, Y. Chen, J. Choi, and C.-C. J. Kuo, “Towards Visible and Thermal Drone Monitoring with Convolutional Neural Networks,” *APSIPA Transactions on Signal and Information Processing*, vol. 8, 2019.

[28] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2016.

[29] F. Svanström, C. Englund, and F. Alonso-Fernandez, "Real-Time Drone Detection and Tracking With Visible, Thermal and Acoustic Sensors," in *International Conference on Pattern Recognition (ICPR)*, pp. 7265–7272, 2021.

[30] C. Briese, A. Seel, and F. Andert, "Vision-based detection of non-cooperative UAVs using frame differencing and temporal filter," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 606–613, 2018.

[31] H. Bergenroth, "Use of Thermal Imagery for Robust Moving Object Detection," Master's thesis, Linköping University, 2021.

[32] D. A. da Silva Justino, "LiDAR and Camera Sensor Fusion for Onboard sUAS Detection and Tracking," Master's thesis, Instituto Superior Técnico, 2020.

[33] M. R. Santos, "Non-cooperative UAV Detection and Relative Position Estimation," Master's thesis, Instituto Superior Técnico, 2021.

[34] Infiniti Electro-Optics, "Visible Imaging Sensor (RGB Color Camera)." `https://www.infinitioptics.com/glossary/visible-imaging-sensor-400700nm-colour-cameras`. Accessed: 09.09.2022.

[35] View Sheen Technology, "What is NIR/SWIR/MWIR/LWIR/FIR Spectral Range?." `https://www.viewsheen.com/news/nir-swir-spectrum/`. Accessed: 09.09.2022.

[36] J. Wallace, "LWIR cameras are the powerhouse behind thermal imaging," *Laser Focus World*, 2020.

[37] S. Mallick, "Image Alignment (ECC) in OpenCV (C++ / Python)." `https://learnopencv.com/image-alignment-ecc-in-opencv-c-python/`. Accessed: 19.08.2022.

[38] G. D. Evangelidis and E. Z. Psarakis, "Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1858–1865, 2008.

[39] W. S. Jaskirat Kaur, "Tools, techniques, datasets and application areas for object detection in an image: a review," *Multimedia Tools and Applications*, 2022.

[40] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

[41] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving Into High Quality Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

[42] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

[44] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," *arXiv:2105.04206*, 2021.

[46] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv:2207.02696*, 2022.

[47] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision (ECCV)*, p. 740–755, 2014.

[48] 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 Workshops*, pp. 390–391, 2020.

[49] Papers with Code, "Real-Time Object Detection on COCO." `https://paperswithcode.com/sota/real-time-object-detection-on-coco`. Accessed: 05.09.2022.

[50] Papers with Code, "Multi-Object Tracking." `https://paperswithcode.com/task/multi-object-tracking`. Accessed: 11.10.2022.

[51] 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," *arXiv:2110.06864*, 2021.

[52] F. Zeng, B. Dong, T. Wang, X. Zhang, and Y. Wei, "MOTR: End-to-End Multiple-Object Tracking with Transformer," *arXiv:2105.03247*, 2021.

[53] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8844–8854, 2022.

[54] H. Saribas, H. Cevikalp, O. Köpüklü, and B. Uzun, "TRAT: Tracking by attention using spatio-temporal features," *Neurocomputing*, vol. 492, pp. 150–161, 2022.

[55] Papers with Code, "Multi-Object Tracking on MOT17." `https://paperswithcode.com/sota/multi-object-tracking-on-mot17`. Accessed: 20.04.2022.

[56] Glenn-jocher, "Precision-recall curve." `https://github.com/ultralytics/yolov3/issues/898`. Accessed: 17.09.2022.

[57] R. S. Keni Bernardin, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *EURASIP Journal on Image and Video Processing*, 2008.

[58] VisAI Labs, "Evaluating multiple object tracking accuracy and performance metrics in a real-time setting." `https://visailabs.com/evaluating-multiple-object-tracking-accuracy-and-performance-metrics-in-a-real-time-setting/`. Accessed: 25.10.2022.

[59] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision*, pp. 17–35, Springer, 2016.

[60] K. Bernardin, A. Elbs, and R. Stiefelhagen, "Multiple object tracking performance metrics and evaluation in a smart room environment," in *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, vol. 90, Citeseer, 2006.

[61] C. Liu, X. Sui, G. Gu, and Q. Chen, "Shutterless non-uniformity correction for the long-term stability of an uncooled long-wave infrared camera," *Measurement Science and Technology*, vol. 29, no. 2, p. 025402, 2018.

[62] W. Kin-Yiu, “YOLOR (official implementation).” `https://github.com/WongKinYiu/yolor`. Accessed:20.04.2022.

[63] J. Zhao, “ICCV2021 Anti-UAV Challenge Dataset.” `https://anti-uav.github.io/dataset/`. Accessed: 25.04.2022.