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

João Paulo Dinis Lopes
joao.dinis.lopes@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

December 2022

Abstract

Conventional 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. Taking into account 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 work presents a novel approach on the detection and tracking of Unmanned Aerial Vehicles (UAVs), 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, to achieve 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

1. Introduction

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 or UAVs (unmanned aerial vehicles). 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. However, 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. Their size and agility also makes detecting and tracking them a very challenging task.

1.1. Project Overview

This work focuses on the detection and tracking of small UAVs using 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 (Simple Online and Realtime Tracking) and a modified version of Deep SORT are also tested, as well as the You-Only-Learn-One-Representation (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 an object detector to find possible targets in the image. The results from the detector are then aligned to compensate for camera motion and given to a 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).

1.2. Related Work

Recently, empowered by the advances in computer vision, several visual object tracking frameworks have been proposed in the context counter UAV operations. Detection based models are the most common in the literature, mainly due to their simplicity, as they just adapt other networks to the task of object tracking.

In [1] common object detectors, such as Faster

RCNN, SSD, YOLOv3 and DETR are combined with trackers: SORT, Deep SORT and Tracktor to establish a comprehensive performance benchmark for comparing these algorithms in the context of Unmanned Aerial Vehicle visual detection and tracking. In [2] the SSD detector is used to detect persons, bicycles and cars in infrared (IR) videos and in [3] YOLOv3 is trained to detect persons and vehicles in search and rescue scenarios using visible light and infrared images. Similarly, [4] uses YOLOv4 on visible light and thermal images for human detection from an aerial perspective. Finally, [5] uses the Faster-RCNN as a detector and Multi-Domain Network (MDNet) as a tracker for visible and thermal drone monitoring, and [6] uses YOLOv2 and a multi-object Kalman filter tracker, similar to SORT, for real-time drone detection and tracking with visible, thermal and acoustic sensors.

2. Theoretical Background

2.1. Anti-UAV Challenge Dataset

The 1st Anti-UAV Workshop and Challenge (2020) [7] brought the release of the original Anti-UAV dataset. The labeled part of this dataset is comprised of 100 fully-annotated visible light 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 lightning and background conditions.

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. Conventional algorithms try to estimate the parameters of one of the following motion models (in increasing order of complexity): Translation, Euclidean, Affine or Homography.

2.3. ECC Algorithm

One algorithm that aims at estimating the motion model between two given images, is the Enhanced Correlation Coefficient Maximization (ECC) [8]. 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.

2.4. 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.

You Only Learn Once (YOLO) [9], and its variants, are the current state-of-the-art in real-time object detection. 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 that cell. Each cell outputs B bounding box detections, with every detection consisting on the probability that there is an object and the description of that object (i.e. center position (x, y), dimensions (w, h) and 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:

$$F = B(C + 5). \tag{1}$$

Several YOLO variations have been proposed over the years. YOLOR [10] was introduced in 2021 and proposes a network that can encode explicit and implicit knowledge together, through the use of parameters learned during training and incorporated into the CNN using addition, multiplication or concatenation.

YOLOv7 was introduced in 2022 and is now the state-of-the-art in real-time object detection [11]. The main features introduced or improved in YOLOv7 are: model re-parameterization, model scaling and an Extended Efficient Layer Aggregation Network (E-ELAN).

2.5. Multi-object Tracking (MOT)

MOT is the problem of automatically identifying multiple objects in a video and representing them as a set of trajectories with high accuracy.

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 an example of one of these algorithms is Simple Online and Realtime Tracking (SORT) [12]. This algorithm works by describing every track (and, therefore, every object) with a unique ID and a Kalman filter containing the size and position of the object (in pixel coordinates), as well as its derivatives.

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 op-

timally 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.

Deep SORT [13] 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.

ByteTrack [14] is another multi-object tracker heavily inspired by SORT. 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 that a threshold).

2.6. Evaluation Metrics

The Intersection over Union (IoU) 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:

$$IoU = \frac{Area(B_1 \cap B_2)}{Area(B_1 \cup B_2)}. \quad (2)$$

The Precision, P, is defined as the number of correct detections over the total number of detections predicted by the model:

$$P = \frac{TP}{TP + FP}, \quad (3)$$

where TP is the number of true positives and FP is the number of false positives.

The Recall, R, is defined as the number of correct detections over the total number of objects present in an image:

$$R = \frac{TP}{TP + FN}, \quad (4)$$

where TP is the number of true positives and FN is the number of false negatives.

Changing the confidence score threshold of the detector will result in a Precision-Recall trade-off which can be plotted into a curve. The Average Precision, AP, is defined as the area under the Precision-Recall curve:

$$AP = \sum^{n} (R_n - R_{n-1}) \times P, \quad (5)$$

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., $P = max\{P(R) : R \geq R_n\}$.

The mAP is the average across all classes of the AP value per class,

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \quad (6)$$

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.

Switching to the Multi-Object Tracking metrics, the IDF1 is a score that balances identification precision and recall through their harmonic mean, similar to other F1 scores. Is is given by:

$$IDF1 = \frac{2IDTP}{2IDTP + IDFP + IDFN}, \quad (7)$$

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

The MOTA is an alternative metric to evaluate the performance of a tracker. It is given by:

$$MOTA = 1 - \frac{FN + FP + IDS}{GT}, \quad (8)$$

where the denominator, GT, refers to the total number of objects in the ground truth, and IDS refers to the identity switches (occurrences where the track ID for an object changes compared to previous frames).

3. Implementation

3.1. Previous Work

The work presented in this work carries on the counter-UAV project developed at the Center for Aerospace Research (CfAR) in the past two years, with main contributions from Daniel Justino [15] and Mariana Santos [16]. The previously developed system uses a LiDAR ((Light Detection and Ranging) as well as a visible light camera to detect, track and estimate the position of flying UAVs in the world coordinate frame.

The detector chosen for their system was the YOLOv4 (the latest YOLO version at the time their 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.

One of the restrictions of their system is that the initialization solely relied 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. Another limitation is that the previously developed system does not take into account the camera's or the tracking vehicle's motion.

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 visible light camera with an IR camera, using image alignment to compensate for camera motion, providing the detector with moving object information and upgrading the tracker from the modified Deep SORT to a new version of the ByteTrack algorithm. A diagram of the proposed system is presented in Figure 1.

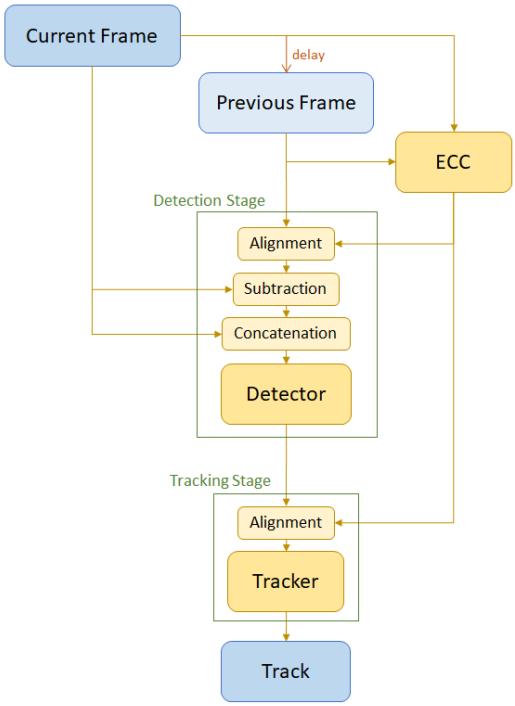


Figure 1: Diagram of the proposed system.

The system starts by running 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, updates its tracks using the new information and outputs a list with the bounding box and ID of every UAV in the respective frame.

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.

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.

The selected 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 did not present significant rotation or changes in magnification.

As for the scale used to reduce the size of the ECC input images, after some experimentation, it 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.

Another aspect to consider is that image alignment relies on having background visual features rich enough to be tracked across frames, which does not 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 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 will not significantly impact the tracker results either.

Another factor that could deteriorate the ECC results are features in the images that do not change with camera position. These include characters added to the images, sensor imperfections, dirt, etc. To minimize this issue, the uppermost part of the images with the frame and time stamps (visible in Figure 2a) was cropped out of the ECC algorithm (this procedure was always applied to the videos in this dataset).

3.4. Detector

The detectors chosen for this work are some of the latest versions of the popular YOLO series: YOLOR [10] and YOLOv7 [17] .

The authors of YOLOR present several models, 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.

Some changes had to be made to its code available on github [18], 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 (1). Note that the YOLOR architecture predicts three bounding boxes per grid cell.

The authors of YOLOv7 (some of whom worked on the YOLOR project) also present several models. Out of these, the YOLOv7-X was chosen for this work, for identical reasons as the YOLOR-CSP-X. Similar to the procedure applied to YOLOR, the number of classes in this model was also changed to one.

3.5. 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.

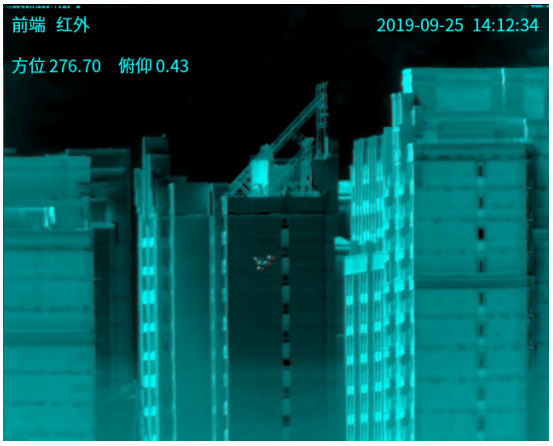
More specifically, to take advantage of the temporal information present in videos, instead of looking at every frame independently, information regarding moving objects was added to the input of the network.

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 one 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 is 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 2 it is possible to visualize the original frame and the final input given to the network.

(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 2: Images used to train the neural networks.

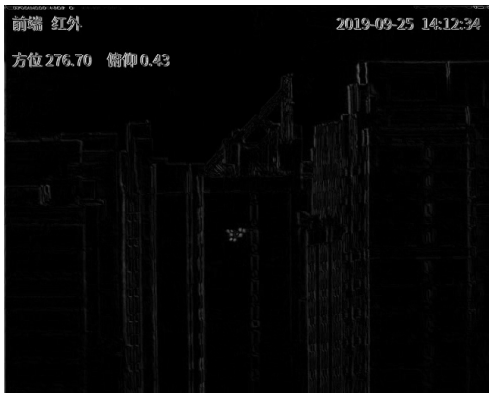
As stated before, the previous image was aligned to the current frame of reference before the subtraction. The purpose of this alignment is to compensate for the camera motion between both frames, which would result in misaligned backgrounds and a significant apparent noise in the subtraction result. In Figure 3 is it possible to visualize the subtraction results with and without the use of frame alignment. As evidenced by this figure, 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.

Bringing two images to the same frame of reference leads to undetermined borders on the trans-

(a) Result of raw subtraction.

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

Figure 3: 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].



(a) Before padding.

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

Figure 4: Result of padding the frame of an image. The raw result of subtraction is presented (without re-scaling).

formed 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 4 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 Figure 3b.

One obstacle found when storing the results of image subtraction as a channel in the images, was the image/video compression algorithms used to store data more compactly, which were empirically shown to significantly degrade the subtraction information. 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.

3.6. 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.

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, each with 14030 images, and two validation sets, each 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 (pretrained on the COCO dataset).

3.7. 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.

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.

3.8. Detection alignment

A brief analysis of the Anti-UAV dataset reveals a considerable amount of motion present in its videos. In order to improve the tracker's results, the apparent shift in the videos is estimated and used to convert all the detections 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.

3.9. New Similarity Metric for Bounding Box Assignment

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.

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.

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:

$$\begin{aligned} distance &= \left(\left(\frac{x_d - x_t}{w_t} \right)^2 + \left(\frac{y_d - y_t}{h_t} \right)^2 \right)^{\frac{1}{2}}, \\ similarity &= e^{-distance} \cdot \frac{min(w_d \cdot h_d, w_t \cdot h_t)}{max(w_d \cdot h_d, w_t \cdot h_t)}. \end{aligned} \tag{9}$$

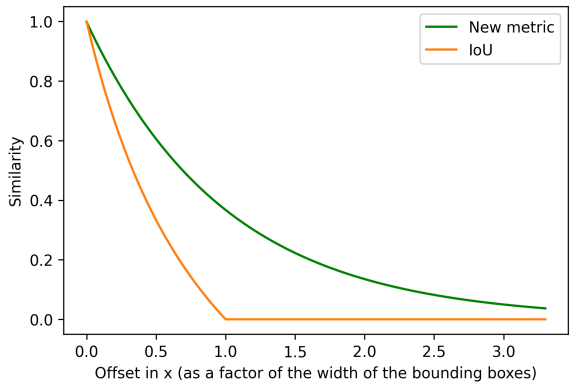
Equation (9) starts by calculating the distance between the centers of the two bounding boxes, relative to the size of the trackers bounding box. The exponential of the symmetric of this value is then used to transform the distance to the interval $]0, 1]$, where one corresponds to a perfect match in 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.

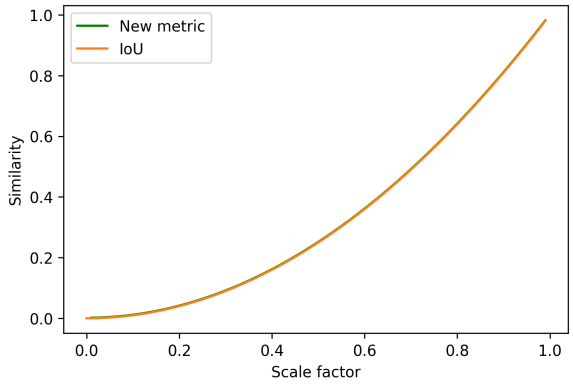
Keeping the similarity metric in the interval $]0, 1]$ (where one corresponds to a perfect match, and zero to the worst possible match) was desirable to make the choice for the matching threshold easier and more intuitive, as well as to keep consistency between the new metric and the IoU for future comparisons. 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 5. 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 5a) or scale (Figure 5b).

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).



(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 5: Comparison between the new assignment metric and the intersection over union.

3.10. 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 small number of objects being tracked. This can, however, have a negative impact in performance if there are plenty of targets in the video, especially 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 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 GPU aboard the patrolling UAV).

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.

4. Results

4.1. Detector

The results obtained using the best YOLOR and YOLOv7 weights on the test set are presented in Tables 1 and 2, respectively.

Table 1: YOLOR test results. *Sub.* refers to whether or not the model has the subtraction information.

Sub.	P	R	mAP@0.5	mAP@[.5:.95]
No	0.919	0.963	0.958	**0.468**
Yes	**0.928**	**0.966**	**0.960**	0.462

Table 2: YOLOv7 test results. *Sub.* refers to whether or not the model has the subtraction information.

Sub.	P	R	mAP@0.5	mAP@[.5:.95]
No	0.913	0.951	0.929	**0.436**
Yes	**0.914**	**0.953**	**0.937**	0.432

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 aren't as tight around the objects leading to poorer IoU between the detections and the ground truth.

4.2. 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 are presented in Table 3.

Table 3: Test results using different trackers.

Tracker	IDF1	MOTA	FP	FN	IDS
SORT	0.813	0.950	**40**	819	37
Mod. D. 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. D. SORT refers to the modified version of Deep SORT (without the appearance descriptors) and New Bytetrack refers to the Bytetrack using the new assignment metric.

From the results in Table 3 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.

4.3. 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, 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: New ByteTrack results.

Model	Sub	Validation		Test	
		IDF1	MOTA	IDF1	MOTA
YOLOR	No	0.980	0.988	0.929	0.980
	Yes	**0.982**	**0.989**	0.929	0.980
YOLOv7	No	0.967	0.984	0.934	0.981
	Yes	**0.981**	**0.989**	**0.960**	**0.991**

From the results in the Table 4, it is possible to verify that 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 IDF1 score of 0.960 and a MOTA of 0.991.

4.4. 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 Tables 5 and 6, and were obtained for both the New ByteTrack and SORT trackers.

Table 5: Effect of aligning detections on the MOTA.

Tracker	with alignment	without alignment
SORT	0.950	0.932 (-1.90%)
New Byte	0.980	0.979 (-0.1%)

Even though the alignment step does not 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.

Table 6: Effect of aligning detections on the IDF1 score.

Tracker	with alignment	without alignment
SORT	0.813	0.637 (-21.65%)
New Byte	0.929	0.921 (-0.86%)

4.5. 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 7 and were collected using the ByteTrack tracker and a single graphics processing unit, the NVIDIA GeForce RTX 2070 SUPER.

In Table 7, the term *Inference* refers to the execution of the YOLO algorithm, including both the CNN and Non-Maximum Suppression. 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.

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 fps using YOLOv7 .

4.6. Limitations

Even though the above mentioned models achieved very satisfactory results on the validation and test sets, there is a key limitation necessary to point out. This limitation arises from the lack of diversity on the Anti-UAV dataset, which results in high validation and test scores, however, that does not mean these models will generalize well to data from other datasets. Dynamic background features will introduce a substantial amount of subtraction noise which the models were not trained to ignore. The drones themselves were also relatively similar, often close to the camera and with a clear sky as background, which simplified the task of their detection and tracking.

5. Conclusions

The presented work 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.

The model 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, corrob-

Table 7: 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

orating the accuracy and reliability of the presented algorithm. 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.

References

[1] 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.

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

[3] "Object detection from thermal infrared and visible light cameras in search and rescue scenes," in *Internationa Symposium on Safety, Security, and Rescue Robotics*, pp. 380–386, 2020.

[4] 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*, pp. 41–46, 2021.

[5] 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.

[6] 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*, pp. 7265–7272, 2021.

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

[8] 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.

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

[10] 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.

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

[12] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *International Conference on Image Processing*, pp. 3464–3468, 2016.

[13] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *International Conference on Image Processing*, pp. 3645–3649, 2017.

[14] 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.

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

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

[17] 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.

[18] W. Kin-Yiu, "YOLOR." `https://github.com/WongKinYiu/yolor`. Accessed:20.04.2022.