# Non-cooperative UAV Detection and Relative Position Estimation

Mariana Ricardo Santos
marianaricardo@tecnico.ulisboa.pt

Instituto Superior Técnico, Univesidade de Lisboa, Portugal

November 2021

**Abstract**

The presented work proposes a system designed to run onboard a UAV (Unmanned Air Vehicle) and capable of detecting and estimating the relative position of non-cooperative UAVs. The system utilizes both LiDAR (Light Detection and Ranging) measurements and images from a camera to detect and track nearby UAVs using a deep learning approach. Instead of manually labelling a dataset to train the chosen object detector - YOLO (You Only Look Once), synthesized images were automatically created and annotated using the open-source software AirSim. YOLO was trained with 4761 training and 530 validation images. YOLO outputs bounding box coordinates that are combined with measurements given by the LiDAR to estimate the relative position of the encountered UAVs. Kalman Filtering is used to smooth the obtained estimations. The system can be used in situations where conventional localization systems are not a good solution such as sense and avoid.
**Keywords:** Camera, LiDAR, YOLO, AirSim, Kalman Filter

## 1. Introduction

As the technological capabilities of UAVs increase, these aircraft are becoming more popular due to their wide and various applications. These vehicles present a robust solution for many industries and are capable of serving several commercial and individual needs. However, UAVs still have limitations in practical scenarios due to their lower payload capacity, shorter endurance and operational range. The potential of UAVs could be further explored if autonomous operations were possible, without the need for a human pilot. For this, UAVs need to be equipped with AI algorithms capable of handling difficult problems and reacting in complex scenarios.

### 1.1. Project Overview

The proposed system is part of a research performed at the CfAR (Centre for Aerospace Research) of the University of Victoria. The task consisted in continuing the work developed in the previous year by Daniel Justino [9] where a sensing system detects and tracks potential intruder UAVs using a LiDAR and a Camera. A LiDAR can provide 3D point clouds with representations of the environment at ranges over $100m$ at high update-rates $(5 - 20Hz)$ in low visibility conditions. However, the 3D point clouds provided are sparse and detections worsen in the presence of rain, fog, snow or dust. Cameras can provide rich texture-based and colour-based information. By combining both measurements from a LiDAR and Camera it is possible to improve the deficiencies from each individual system.

The system should fly onboard a UAV and be used for integration with avoidance trajectory planning algorithms for sense and avoid applications. Therefore, the method presented aims to continue the previous work by estimating the relative position of the encountered UAVs.

### 1.2. Related Work

Numerous approaches to vision-based UAV detection and tracking have been presented in the literature. Some of the most effective approaches utilise multi-stage-detection pipelines, machine learning and deep learning [8].

More recently, deep learning approaches have been explored for the vision-based detection and relative localization estimation of objects. In [19] a single camera and the deep learning algorithm YOLO [15], are used to estimate the distance to a MAV (Micro Aerial Vehicle) given its known physical size. A similar approach is presented in [16] where the algorithm can be used for a larger number of drones by training YOLO with a dataset that is created while systematically flying a quadcopter in front of a static camera and applying background subtraction to the images. In [11], a monocular camera and YOLO are used in a collision avoidance system where a nested Kalman Filter is used to improve the estimations of distance and velocity. All

these methods use single cameras and require a priori information about the vehicles, so they cannot be applied to non-cooperative UAVs.

An example of a LiDAR Obstacle Warning and Avoidance System (LOWAS) employed in a system for UAS (Unmanned Aerial System) sense and avoid applications is presented in [14]. In [20], a single camera and a LiDAR are used for the detection of beacons, with results improving by using information from both sensors. A neural network is trained with data from the LiDAR and the camera so that it could estimate the position of the encountered beacons. These estimations were compared with LiDAR 3D point clouds and fuzzy logic was used to compute the score of the final outputted distance.

## 2. Background

### 2.1. Visual Object Detection

Humans are capable of looking at an image and instantly know which objects are present and their positions. One of the fundamental computer vision problems is to completely understand an image by trying to precisely estimate the concepts and locations of the objects contained in each image. The current state-of-the-art deep learning object detector YOLO is a CNN (Convolutional Neural Network) that can be trained to detect different classes of objects in an image. The output of YOLO is a set of predicted bounding boxes that consist of 5 predictions: $x$, $y$, $w$, $h$, and confidence. The $(x, y)$ coordinates represent the centre of the box and $(w, h)$ its width and height. YOLO keeps only the bounding boxes with the highest confidence scores. Since the release of the first version of YOLO, many improvements have been made to the object detector. The presented work implements version 4 of YOLO that corresponds to the one with the best performance and higher speed.

### 2.2. Multi-Object Tracking

The objective of MOT (Multi-Object Tracking) is to analyse videos to identify and track objects without any prior knowledge about their appearance and number.

Deep SORT (Simple Online and Realtime Tracking with a Deep Association) proposes a tracking-by-detection framework for the MOT problem which can be used for real-time applications. It integrates appearance information to improve performance and assumes the camera is not calibrated and there is no ego information available. For each new UAV assignment, the algorithm creates a new track that stays in a tentative state until $A_{max}$ consecutive associations are successful. Afterwards, the track transitions into an active state. Otherwise, the track is deleted. The track remains in an active state for as long as observations keep being assigned to them. When no associations are made, the track transitions into a lost state where the Kalman Filter from Deep SORT continues to predict the target position. If it associates one observation with a track, this track transitions back to the active state, but if no associations are made for $N$ iterations, the track is removed.

### 2.3. Geometric Camera Parameters

The pinhole perspective projection model is a mathematically convenient and simple approximation of the imaging process. Let $P$ denote a scene point with coordinates $(X, Y, Z)$, $p$ denote its image with coordinates $(x, y, z)$ and $O$ the ideal pinhole of the camera, as shown in Fig. 1. Since $p$ is a point

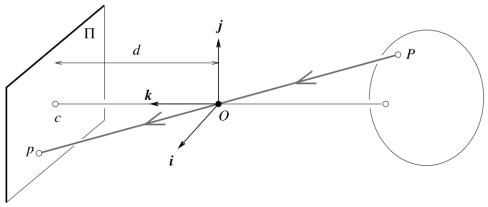



Figure 1: Collinearity of the point $P$, its image $p$, and the pinhole $O$ from which the perspective projection equations are derived. [6]

in the image plane, $z = d$ and because $p$, $P$ and $O$ are collinear yields,

$$\begin{cases} x = \lambda X \\ y = \lambda Y \\ d = \lambda Z \end{cases} \Leftrightarrow \begin{cases} x = d\frac{X}{Z} \\ y = d\frac{Y}{Z} \end{cases} . \tag{1}$$

Consider a coordinate vector P to be $(X, Y, Z)^T$ in $\mathbb{R}^3$ in some fixed world coordinate system. Its homogeneous coordinate vector is $\mathbf{P} = (X, Y, Z, 1)^T$ in $\mathbb{R}^4$ and the correspondent vector $\mathbf{p} = (x, y, 1)^T$ of its image $p$ in the camera's reference frame are related by the perspective projection equation [6],

$$\mathbf{p} = \frac{1}{Z}\mathcal{M}\mathbf{P}, \tag{2}$$

where $\mathcal{M}$ is the matrix that provides an algebraic representation of the perspective projection process.

Consider a normalized image plane parallel to the image plane but located at a distance $d = 1$ from the pinhole, $O$. Let $\hat{\mathbf{p}} = (\hat{x}, \hat{y}, 1)^T$ be the homogeneous coordinates of the projection of point $P$ into the normalized image plane. Equation (1) can be written as

$$\begin{cases} \hat{x} = \frac{X}{Z} \\ \hat{y} = \frac{Y}{Z} \end{cases} \Leftrightarrow \hat{\mathbf{p}} = \frac{1}{Z}\begin{bmatrix} Id & 0 \end{bmatrix}\mathbf{P}. \tag{3}$$

The relation between the vector of homogeneous coordinates in the image frame and the homogenous

coordinates in the normalized image plane is obtained through the intrinsic camera calibration matrix $\mathcal{K}$:

$$\mathbf{p} = \mathcal{K}\hat{\mathbf{p}} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}, \tag{4}$$

and, consequentially, equation (2) can be rewritten as

$$\mathbf{p} = \frac{1}{Z}\mathcal{K}\begin{bmatrix} Id & 0 \end{bmatrix}\mathbf{P}. \tag{5}$$

The pinhole model provides only a simple approximation and the effect of radial and tangential lens distortion should also be taken into account. The camera intrinsic calibration matrix can be augmented with the radial and tangential distortion coefficients $k_1, ..., k_n$, $p_1$ and $p_2$. Let $\tilde{\mathcal{K}}$ be the augmented matrix of $\mathcal{K}$ yields,

$$\mathbf{p} = \frac{1}{Z}\tilde{\mathcal{K}}\begin{bmatrix} Id & 0 \end{bmatrix}\mathbf{P}. \tag{6}$$

Finally, extrinsic camera parameters are necessary to transform the world coordinates to the camera centred coordinate frame. Let ${}^{C}\mathbf{P}$ denote the vector of homogeneous coordinates of point $P$ expressed in the camera coordinated frame, $C$ and ${}^{W}\mathbf{P}$ the same vector expressed in the world coordinate frame, $W$. It is possible to write

$$\mathbf{p} = \frac{1}{Z}\tilde{\mathcal{K}}\begin{bmatrix} \mathcal{R} & \mathbf{t} \end{bmatrix}{}^{W}\mathbf{P}, \tag{7}$$

where $\mathcal{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector.

### 2.4. LiDAR Principles

LiDAR sensors use information acquired by the light emitted by a laser beam or pulse. The emitted power must be kept within a safety envelope and an adequate frequency must be selected to guarantee safety in human environments [5]. Usually, a very short but intense pulse of laser radiation is used to compute distances by measuring its time of flight accordingly with,

$$d = \frac{c \cdot (t_r - t_t)}{2}, \tag{8}$$

where $d$ is the distance between the LiDAR and the object, $c$ is the speed of electromagnetic radiation and $t_r$ and $t_t$ are the received and transmitted time measurements, respectively.

The relationship between the distance measurements and the LiDAR coordinate frame is given by

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = d \begin{bmatrix} cos(\psi) \cdot cos(\theta) \\ cos(\psi) \cdot sin(\theta) \\ sin(\psi) \end{bmatrix}, \tag{9}$$

where $\psi$ and $\theta$ are the elevation and azimuth angles of the emitted laser beam, respectively [18].

### 2.5. Kalman Filtering

Consider a linear, discrete-time system having dynamics [7]

$$x_{k+1} = A_k x_k + B_k u_k + G_k w_k, \quad t \geq 0 \tag{10a}$$
$$y_k = H_k x_k + v_k \tag{10b}$$
$$x_0 = x_0 \tag{10c}$$

where $x_k$, $u_k$, $y_k$, $w_k$ and $v_k$ represent the state, control, measurements and measurements noise vectors, respectively, and $A_k$, $B_k$, $G_k$ and $H_k$ represent the system transition, input distribution, system noise and observation matrices, respectively. $w_k$ and $v_k$ are Gaussian, white noises processes satisfying

$$\begin{aligned} &E\{w_k\} = 0, \quad E\{v_k\} = 0, \quad E\{w_k v_j^T\} = 0, \\ &E\{w_k w_j^T\} = \begin{cases} 0 & k \neq j \\ Q_k & k = j \end{cases}, \\ &E\{v_k v_j^T\} = \begin{cases} 0 & k \neq j \\ R_k & k = j \end{cases}, \end{aligned} \tag{11}$$

and have covariance matrices $Q_k$ and $R_k$, respectively, both positive definite [7].

The Kalman Filter performs two steps: prediction, with $\hat{x}_{k+1}^-$ representing the predicted estimate and update, with $\hat{x}_{k+1}^+$ representing the updated estimate.

**Prediction Step:**

$$\hat{x}_{k+1}^- = A_k \hat{x}_k^+ + B_k u_k, \tag{12a}$$
$$P_{k+1}^- = A_k P_k^+ A_k^T + G_k Q_k G_k^T. \tag{12b}$$

**Update Step:**

$$\hat{x}_{k+1}^+ = \hat{x}_{k+1}^- + K_{k+1}[y_{k+1} - H_{k+1}\hat{x}_{k+1}^-], \tag{13a}$$
$$P_{k+1}^+ = [I - K_{k+1}H_{k+1}P_{k+1}^-], \tag{13b}$$

where $K_{k+1}$ is given by

$$K_{k+1} = P_{k+1}^- H_{k+1}^T [H_{k+1} P_{k+1}^- H_{k+1}^T + R_{k+1}]^{-1}. \tag{14}$$

The Kalman filter only needs to remember the previous step making it computationally efficient.

### 2.6. Evaluation Metrics

The precision, $P$, translates the capacity to perform correct detections since it gives the relationship between the number of objects detected versus the total number of detections. It is defined as

$$P = \frac{TP}{TP + FP}, \tag{15}$$

where $TP$ and $FP$ are equal to the number of true positives and false positives detected, respectively.

The recall, $R$, translates the ability to find all the objects in the scene since it gives the relationship

between the number of objects detected versus the total number of objects. It is defined as

$$R = \frac{TP}{TP + FN}, \tag{16}$$

where $FN$ corresponds to the number of false negatives in a set.

The RMSE (Root Mean Squared Error) has been used as a standard statistical metric to measure model performance. It is a measure of accuracy and it is always non-negative. It is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\hat{x}_n - x_n\right)^2}, \tag{17}$$

where $N$ is the total number of measurements, $\hat{x}_n$ is the estimated value of the $n^{th}$ predicted measurement and $x_n$ is its correspondent actual value.

Finally, the MaxAE (Maximum Absolute Error) is given by

$$MaxAE = max(|\hat{x}_n - x_n|), \tag{18}$$

where $\hat{x}_n$ is the estimated value of the $n^{th}$ predicted measurement and $x_n$ is its correspondent actual value.

## 3. Implementation

### 3.1. Previous Work

The work presented carries on the work previously developed by Daniel Justino in [9] during his time at CfAR. Therefore, an overview of the algorithm that he developed for the task of detection and tracking of non-cooperative UAV is presented.

Using an object detector such as YOLO to inspect a whole image and detect UAV is computationally expensive and can lead to the acquisition of a large number of false positives. For this reason, one of the largest contributions of the previously developed work was the generation of ROIs (Regions of Interest) in the captured images, in which YOLO searches for UAVs.

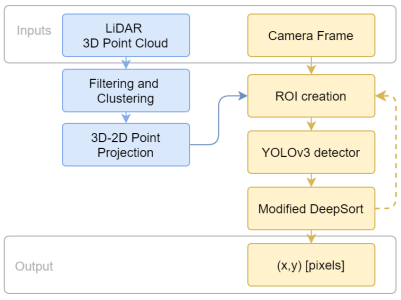



Figure 2: Previously proposed architecture. The dashed line represents a link between consecutive iterations [9].

Note the scheme present in Fig. 2. The LiDAR acquires a UAV by giving a 3D point cloud that is clustered and filtered by the algorithm to obtain just the points relative to the encountered UAV. These points are projected into the 2D image using the geometric camera parameters and equation (7). Afterwards, a ROI centred in the 2D projected point is created with a fixed size of 128x128 pixels. The object detector YOLOv3 scans this ROI and tries to detect the UAV inside it. If a UAV is detected, the tracking algorithm based on Deep SORT is used to predict the position of the UAV in the following frame. For this reason, in the next frame YOLO will be inspecting ROIs created by the LiDAR 3D-2D projection and ROIs predicted by the modified Deep SORT. The modified Deep SORT tracker is especially important when LiDAR measurements are not available. Deep SORT was planned for the tracking of pedestrians, where CNN appearance descriptors add value to the algorithm but, for the specific case of small UAVs, the appearance descriptors were discarded and the algorithm is called modified Deep SORT.

This architecture detects and tracks UAVs in an image but is not capable of providing relative position estimations of the detected UAV, $(X, Y, Z)$, in the world coordinate frame. The work presented in this thesis aims to continue the work discussed thus far and find a solution for the problem of relative position estimation of non-cooperative UAVs using a LiDAR and a camera.

### 3.2. System Overview

The method proposed can be divided into two different steps: detection and relative localization estimation, as presented in Figure 3.

For the detection task, most of the work previously developed was kept. For each frame, the 3D point cloud given by the LiDAR is filtered and clustered. When there are points likely to be related to an encountered UAV, a ROI is created using the 3D-2D projection. YOLO looks for UAVs in the ROIs created by the projection and by the modified Deep SORT tracker. If for a specific frame no ROI is created, the output of that frame is empty. The same happens if YOLO does not detect a UAV in any of the ROIs created for that frame. The modified Deep SORT is used to predict the position of the UAV in the following frames. The algorithm constructs a track with the previous detections and uses this information to predict the position of the UAV in the following frame.

The main work is focused on the second task, relative localization estimation. The position of the UAV is estimated with two different methods. When YOLO detects a UAV with a correspondent LiDAR measurement, i.e., if there is a LiDAR 2D projection inside the bounding box returned by YOLO, the correspondent LiDAR 3D coordinates are used directly. If there are no correspondent Li-

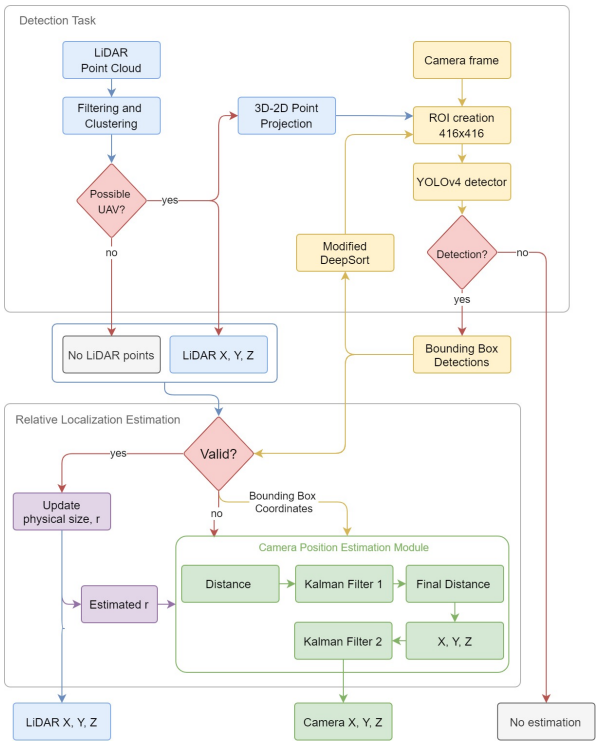



Figure 3: Diagram of the proposed system.

DAR points for the detected UAV, position estimation is performed using the camera.

Because the LiDAR 3D point clouds are sparse, most of the estimations will be based on information from the camera.

To estimate the UAV's position with the camera, it is necessary to estimate its relative distance first. The method used to achieve this requires knowing the apparent physical size of the target, $r$, which will be discussed in detail later, the bounding boxes returned by YOLO and the geometric camera parameters. The value of $r$ is called apparent since it changes accordingly with variations in the attitude of the UAV. A Kalman Filter is used to smooth the estimates of the distance that afterwards is used to compute the relative 3D coordinates of the UAV in the LiDAR's coordinate frame.

The proposed system works for any number of UAVs present in the scene and without any a priori information or assumption.

## 4. Detection Task

For the detection task, some key updates were implemented. Firstly, YOLO was updated to the last released version at the time of the training, YOLOv4 [2], since this version was computationally faster and provided better detection results. Also, the size of the ROI was changed to $416 \times 416$ pixels, which is the standard size of the input images of YOLOv4.

The Deep SORT tracker was configured to keep a new track in the tentative state until $A_{max} = 3$ consecutive associations are successful and to remove this track if no associations are made for $N = 9$ iterations.

The coordinates of the bounding boxes returned by YOLO are used in the estimation of distance and 3D coordinates when using only the camera. These bounding boxes need to fit around the target almost perfectly to achieve good results. For this reason, it was necessary to create a dataset that satisfied this requirement. The open-source software AirSim [17] was used to this end.

### 4.1. AirSim

AirSim [17] is an open-source simulator that enables the development and testing of algorithms for autonomous vehicles and facilitates the collection of annotated data for the training of object detectors such as YOLO. AirSim was built on Unreal Engine [10] which offers realistic simulations, enabling the realization of Software-in-the-loop (SITL) and Hardware-in-the-loop (HITL) experiments.

Amongst other functionalities, it is possible to modify the vehicles and onboard sensors, interact with the vehicle programmatically, using Python code or use external controllers such as PX4 [13]. For trajectory planning, it is possible to integrate software such as QGroundControl [1].

Two UAVs flew in every simulation so that the first UAV (carrying one camera) could record images of the second UAV in 4 different environments. It was decided to train YOLO with images of two UAVs: DJI Inspire 1 and Mavic Pro considering that these were the drones available at CfAR and used in the experiments of previous years. The specifications of these drones can be found in Table 1.

Several flights were performed to capture the drones with different attitudes and at different distances. The camera images were programmed to have $1920 \times 1080$ pixels, a horizontal FOV (Field of View) of 90° and a vertical FOV of 50.6°. Alongside the scene camera frames, segmented images were collected. A Python code was created to change the colours of the segmentation images so that every object in the environment would appear black and only the drones in white. This facilitated the process of finding the bounding boxes of the drones present in the image, using the Python library OpenCV [3]. These points were written to a text file accordingly to the YOLO darknet format. A total of 4761 training images and 530 validation images were gathered to train the object detector.

## 5. Relative Localization Estimation

LiDARs are capable of giving 3D point clouds with high accuracy and at high frame rates but these point clouds are sparse and the UAVs are not

Table 1: Properties of the vehicles used in the AirSim simulations.

| | Dimensions [cm] | Weight [g] | Nº of Motors | Autonomy [min] |
|---|---|---|---|---|
| DJI Inspire 1 | $44 \times 45 \times 30$ | $2,845$ | 4 | 18 |
| DJI Mavic Pro | $32 \times 24 \times 8$[1] | 734 | 4 | 21 |

(a) Scene image from *Landscape Mountains*.

(b) Boundary points of the UAV in (b).

Figure 4: Process for the creation of a dataset in the YOLO Darknet format.

detected when flying between the LiDAR beams. In addition, LiDAR beams can have small ranges. Cameras can capture sequenced frames in which the UAV is always visible and YOLO can always perform detections in these frames. However, single camera position estimation state-of-the-art methods require an a priori knowledge of the encountered UAVs and are usually used in cooperative UAV missions. By using information from the previous LiDAR detections as the necessary a priori knowledge for the implementation of single camera methods, these can be expanded to non-cooperative flying vehicles.

### 5.1. Position Estimation Using LiDAR

LiDAR measurements are used directly whenever available and validated by YOLO, i.e., if the LiDAR 3D-2D projection results in points that are inside the bounding box returned by YOLO. For this reason, the algorithm presented is fully dependent on the detections performed by YOLO. This decision was made taking into account that only an object detector can distinguish a UAV from a bird or the tip of a tree.

### 5.2. Position Estimation Using Camera

Whenever there are no measurements from the LiDAR, the relative position estimation is computed using only the camera. This task can be divided into two phases: distance estimation and 3D coordinate estimation.

It is possible to notice from Fig. 5 that the distance $d$, from the centre of the UAV and the origin of the camera coordinate system, $O_c$, is estimated from the angle $\alpha$ which corresponds to the angle between vectors $a_1$ and $a_2$. These are directional vectors of 3D lines intersecting with $O_c$ and the centres of the horizontal edges of the bounding box of the detected UAV.

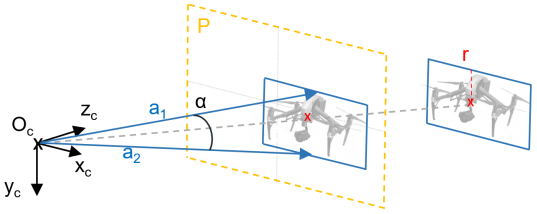



Figure 5: Projection of the UAV in the camera projection plane, $P$. The UAV is specified by the physical size $r$.

The angle $\alpha$ between vectors $a_1$ and $a_2$ is given by

$$\alpha = acos\left(\frac{a_1 \cdot a_2}{\|a_1\| \cdot \|a_2\|}\right), \tag{19}$$

and the approximate distance, $d$, to the detected object is given by

$$d = \frac{r}{tan(\alpha/2)}, \tag{20}$$

where $r$ is the estimated physical size of the UAV.

Since the system should work for any non-cooperative UAV, it is necessary to find a way to estimate the target's physical dimension, $r$. To accomplish this, the last LiDAR measurement before losing the LiDAR information is used. Since the distance given by the LiDAR from frame $t-1$ and the corresponding bounding box returned by YOLO at $t-1$ are known, yields,

$$r(t-1) = d(t-1) \cdot tan(\alpha(t-1)/2). \tag{21}$$

In subsequent camera frames, at $t$, $t+1$ and so on, it is assumed $r$ constant and equal to the last estimate, until a new LiDAR measurement occurs and the value of $r$ is updated. The performance of the distance estimation improves as the number of LiDAR measurements increases since the value of $r$ is updated more often and the algorithm takes into account more changes in the UAV's attitude.

Vectors $a_1$ and $a_2$ are directional vectors with origin in the camera coordinated frame $(x_c, y_c, z_c)$ and end in the image plane. Thus far, the only information available is the coordinates $\mathbf{p} = (x, y)$ of the two points of the horizontal edges of the bounding boxes. To use this method to compute the 3D coordinates in the world coordinate frame, it is necessary to know the value of the coordinate $z$ as well.

However, it is impossible to calculate this value because the distance from the image plane to the camera origin, $O_c$, is unknown. For this reason, vectors $a_1$ and $a_2$ are obtained from the normalized image plane, which is at the known distance $\hat{z} = 1$. The relation between the vector of homogeneous coordinates in the image frame and the homogeneous coordinates in the normalized image plane was given by equation (6) and therefore, the points in the normalized plane can be obtained using it.

The position of the encountered UAV, in 3D coordinates and relative to the camera coordinate frame, is obtained using vectors $a_1$, $a_2$ and distance, $d$. Let vector $a_c$ be the vector from the camera origin towards the centre of the detected UAV, $C$, yields:

$$a_c = \frac{a_1 + a_2}{2}, \quad C = d\frac{a_c}{\|a_c\|}. \tag{22a}$$

The coordinates of point $C$ are given with respect to a coordinate system centred in the camera. It is necessary to transform these coordinates to the LiDAR coordinate frame using the extrinsic camera parameters.

### 5.3. Method Limitations

The bounding boxes returned by YOLO can vary sudden and significantly in size because the bounding box does not always fit around the detected UAV in the same way, inducing a random and unpredictable error. In addition, changes in the attitude of the UAV during its flight can lead to estimation errors. In some cases, with very complex backgrounds, YOLO can detect only a part of the UAV, not bounding its body entirely. Since the distance estimation is dependent on the height of the bounding box returned by YOLO, which is given in pixels, there exists a quantization phenomenon associated with the smallest measurement unit - 1 pixel. For distances of $10m$, a variation of 1 pixel corresponds to an absolute error in distance of $30cm$ but for a distance of $40m$, a variation of 1 pixel corresponds to an absolute error of $6.8m$ in distance, which is significantly larger. Kalman Filtering was implemented to smooth all the estimation variations that might occur due to the phenomena discussed.

### 5.4. Kalman Filtering

For the distance estimation, it was considered a very simple linear, discrete-time system having dynamics given by equation (10). The considered state transition matrix is based on a linear velocity model,

$$A_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}_{2\times 2}, \tag{23}$$

and the input $u = 0$ since there are no control inputs. The Kalman Filter is initialized with estimates of $\hat{x_0}$, and predicted (a priori) estimate covariance, $P_0$, given by

$$\hat{x_0} = \begin{bmatrix} d_{t=0} \\ \dot{d}_{t=0} \end{bmatrix}_{2\times 1}, \quad P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{2\times 2}, \tag{24a}$$

where $d_{t=0}$ is computed with the last available LiDAR measurement and $H_k = \begin{bmatrix} 1 & 0 \end{bmatrix}_{1\times 2}$. The covariance of the measurements noise matrix, $R_k$, and the process noise matrix, $Q_k$, were tuned manually.

For the 3D coordinates stabilization, it was also used a linear, discrete-time system of the form presented in Equation 10. The state transition matrix is based, once again, in a linear velocity model with no control inputs ($u = 0$). The Kalman Filter is initialized with estimates of $\hat{x_0}$, and predicted (a priori) estimate covariance, $P_0$, given by

$$\hat{x_0} = \begin{bmatrix} x_{t=0} & y_{t=0} & z_{t=0} & \dot{x}_{t=0} & \dot{y}_{t=0} & \dot{z}_{t=0} \end{bmatrix}^T_{6\times 1}, \tag{25a}$$

$$P_0 = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \\ 0 & & 1 \end{bmatrix}_{6\times 6}, \tag{25b}$$

where $x_{t=0}$, $y_{t=0}$ and $z_{t=0}$ are given by the last available LiDAR measurement and

$$H_k = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & & 1 & \dots & 0 \end{bmatrix}_{3\times 6}. \tag{26}$$

Matrices $Q_k$ and $R_k$ were tuned manually.

## 6. Implementation for Multi-UAV

To detect and estimate the position of several UAVs in a sequence of images, it is necessary to find a method to associate each detected UAV in a new frame with its previous detection in the last frame. This is an assignment problem that will be solved using a simple distance metric.

The algorithm creates a list containing the information of all the UAVs present in the scene. For each frame, the algorithm searches for UAVs in the list that have been lost for more than LOSS_THRESHOLD frames and removes them.

For each ROI, YOLO tries to detect UAVs. If a new detection is confirmed, the algorithm checks if the list of UAV is empty. When it is, the new detected UAV is added to the list, if not, the algorithm tries to assign the detected UAV with the ones already present in the list.

The assignment is performed based on the coordinates of the centre of the bounding box returned by YOLO, $(c_x, c_y)$, in pixels. These coordinates are compared to the centres of the bounding boxes of all the other UAVs present in the list, according to

$$s = \sqrt{(c_x - c_x^i)^2 + (c_y - c_y^i)^2}, \tag{27}$$

where $(c_x^i, c_y^i)$ represents the centre of the bounding box of the UAV with correspondent index $i$ in the list of vectors. The algorithm chooses the UAV correspondent to the smallest value of $s$ and if $s \leq$ DETECTION_THRESHOLD, the assignment is validated. If not, the detection is considered to be a new UAV that just entered the scene and is added to the list. Because the ROIs have a size of $416 \times 416$ pixels, they may capture more than one UAV. Therefore, after the assignment, the algorithm verifies if the UAV has already been updated for that frame and if so, chooses the estimation that presents the closest distance.

The algorithm only needs to store information regarding the last detection (for each UAV) making it computationally inexpensive.

## 7. Simulations

### 7.1. AirSim Setup

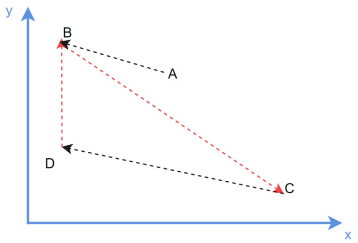
The simulations in AiSim were planned using QGroundControl and transmitted to AirSim by the PX4 controller. The UAV used was a DJI Inspire 1. The chosen environment was different from the ones used to train YOLO and it is composed of a landscape with a clear sky, where a UAV should be easily detected and mountains simulating complex backgrounds.

A UAV carrying a camera and a LiDAR stayed hovering at a fixed position while the other represented a non-cooperative encountered UAV. The LiDAR was simulated using a built-in tool from AirSim and presented 8 detection layers, 420000 points per second, a range of $60m$, a horizontal FOV of 360°, a vertical FOV of 21° $(+3°/-18°)$ and 10 rotations per second. Python code was developed to capture LiDAR measurements, ground truth and camera images during the flights, using APIs. The trajectory was planned so that the maximum velocity of the flying UAV was equal to $4m/s$. The camera was configured to output images of size $1280 \times 720$ since this is the resolution of the camera available at CfAR. It presented a horizontal FOV of 90° and a vertical FOV of 50°. The simulations were configured to give a data rate of $20Hz$.

It was necessary to estimate the camera's intrinsic and extrinsic parameters using the Calibrator App from Matlab [12]. For the extrinsic calibration between the LiDAR and the camera, IPPE (Infinitesimal Plane-based Pose Estimation) [4] was used with points gathered manually.

### 7.2. Yaw Simulation

This first flight was simulated to study the influence of attitude variations in the algorithm's predictions. QGroundControl only allows to control the yaw of the flying UAVs but the results were already enough to infer several conclusions regarding the influence of attitude variations. The UAV flew along the $Z$-axis at positions ranging from 10 to $50m$ in intervals of $10m$. For each distance, the UAV follows the trajectory represented in Figure 6(a), in the $OXY$ plane, following the sequence A-B-C-D-B.




(a) Trajectory followed in the $OXY$ plane.

Figure 6: Trajectory followed by the UAV for the yaw simulation.

The UAV presented yaw equal to 0° from point A to B and from point C to D and yaw equal to 90° from point B to C and D to B.

Position estimations in the $OXY$ plane present good results, with RMSE values lower than $4.34m$ for estimations in $X$ and $1.58m$ for estimations in $Y$. The error in the $Z$-axis is the most affected, presenting a MaxAE of $23.21m$ and a RMSE of $7.38m$ for $Z \approx 50m$. Overall, it can be concluded that the algorithm performs well for positions such that $Z \leq 40m$ but for higher values of $Z$ the estimates start to deviate significantly from the ground truth values. It was concluded that the attitude of the UAV influences the estimations for distances along the $Z$-axis up to $30m$. For further values of $Z$, the UAV is too small and the results are mostly affected by variations in the height of the bounding box.

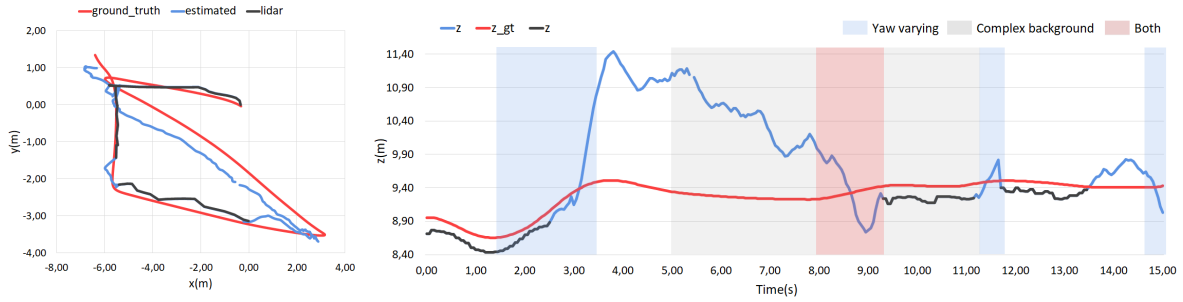
The precision and recall of YOLO were equal to 100% and 93.50%, respectively. YOLO never missed the detection of the UAV for more than LOSS_THRESHOLD consecutive frames for positions with $Z \leq 40m$ but it was lost at $Z \approx 50m$ for $10s$ which corresponds to 3.3% of the flight. An empty frame was defined as the fraction between the number of frames without a position estimation output and the number of total frames. YOLO presented a speed of $1.73Hz$ when running with CPU and a speed of $9.5Hz$ when running with GPU.

### 7.3. Camera Resolution Simulation

To study the impact of the camera resolution in the algorithm's estimations, a flight simulation with a $1080p$ camera was performed. The simulated flight presented the same trajectory as the yaw simulation. When comparing the results obtained with a camera of resolution equal to $1080p$, it is concluded that the results improve for all the values of $Z$ except for $Z \approx 40m$. It was concluded that

Table 2: RMSE and MaxAE results obtained for the yaw simulation with a $720p$ camera.

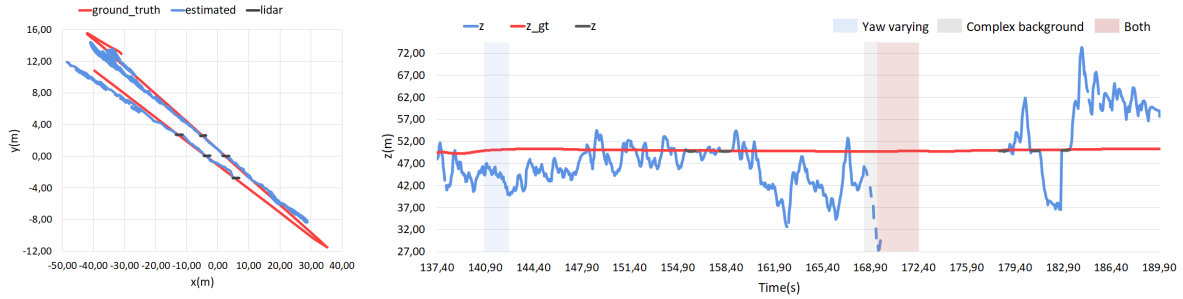
| | RMSE[m] | | | | MaxAE[m] | | | | Empty Frames[%] |
|---|---|---|---|---|---|---|---|---|---|
| | $X$ | $Y$ | $Z$ | $d$ | $X$ | $Y$ | $Z$ | $d$ | |
| $10m$ | 0.43 | 0.19 | 0.80 | 0.86 | 1.25 | 0.39 | 1.93 | 2.19 | 0.33 |
| $20m$ | 0.80 | 0.34 | 1.28 | 1.45 | 2.93 | 1.21 | 3.78 | 4.73 | 8.84 |
| $30m$ | 2.00 | 1.13 | 3.56 | 4.17 | 5.53 | 3.67 | 10.27 | 12.18 | 9.93 |
| $40m$ | 2.12 | 1.30 | 4.03 | 4.62 | 7.11 | 3.83 | 10.53 | 12.91 | 1.29 |
| $50m$ | 4.35 | 1.58 | 7.36 | 8.60 | 16.94 | 5.99 | 23.21 | 28.76 | 17.70 |




(a) Results in the $OXY$ plane.

(b) Results in the $OXZ$ plane.

Figure 7: Estimation results for the yaw simulation at $Z \approx 10m$.




(a) Results in the $OXY$ plane.

(b) Results in the $OXZ$ plane.

Figure 8: Estimation results for the yaw simulation at $Z \approx 50m$.

the system's results improved when using a camera with higher resolution and used to estimate the position of UAVs up to $Z \leq 50m$.

YOLO continued to present precision of 100% and recall improved to 94.42%. It presented a frequency of $1.64Hz$ when running with CPU (Central Processing Unit) and $11.92Hz$ when running with GPU (Graphics Processing Unit).

### 7.4. Multi-UAV Simulation

A simulated flight with two UAVs was performed to exemplify the response of the algorithm when more than one UAV is present in the scene. The minimum distance between the two UAVs was 50 pixels, which is smaller than the DETECTION_THRESHOLD defined (64 pixels). The system was capable of detecting and tracking multiple UAVs in the frames without ever failing to assign the two UAVs correctly.

## 8. Conclusions

The presented work proposes a system for the detection and relative position estimation of non-cooperative UAVs using the state-of-the-art object detector YOLO and data from a LiDAR and an EO camera. The system is capable of detecting and estimating the relative position of any number of encountered UAVs without any a priori knowledge. Overall, the system provided good estimation results for values of $Z$ up to $40m$ whenever using a camera with a resolution of $720p$ and up to $50m$ when using a camera with $1080p$. The maximum RMSE value obtained for $Z \approx 40m$ and a camera with $720p$ was equal to $4.62m$, with correspondent MaxAE equal to $12.91m$. For the case of the camera with $1080p$, the maximum RMSE obtained for $Z \approx 50m$ was $7.80m$ with correspondent MaxAE $18.36m$. For the multi-UAV simulation, the system was capable of detecting and tracking multiple

Table 3: Results obtained for the $1080p$ camera simulation.

| | RMSE[m] | | | | MaxAE[m] | | | | Empty Frames[%] |
|---|---|---|---|---|---|---|---|---|---|
| | $X$ | $Y$ | $Z$ | $d$ | $X$ | $Y$ | $Z$ | $d$ | |
| $10m$ | 0.44 | 0.18 | 0.65 | 0.71 | 0.96 | 0.42 | 1.73 | 1.81 | 0.00 |
| $20m$ | 0.83 | 0.48 | 0.97 | 1.23 | 2.05 | 1.65 | 2.83 | 3.50 | 0.59 |
| $30m$ | 1.40 | 0.84 | 1.94 | 2.36 | 4.06 | 2.68 | 7.28 | 8.49 | 8.03 |
| $40m$ | 2.60 | 1.04 | 5.86 | 6.34 | 6.80 | 3.91 | 14.05 | 14.76 | 11.39 |
| $50m$ | 3.56 | 1.00 | 7.01 | 7.80 | 12.55 | 4.07 | 16.34 | 18.36 | 3.62 |

UAVs in the same frame without ever failing to assign the two UAVs correctly.

**Acknowledgements**

The author would like to thank Professor Afzal Suleman and Professor Rita Cunha for the guidance provided throughout this work.

**References**

[1] QGC - QGroundControl - Drone Control.

[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. apr 2020.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] T. Collins and A. Bartoli. Infinitesimal plane-based pose estimation. *International Journal of Computer Vision*, 109(3):252–286, 2014.

[5] C. V. Dolph, M. J. Logan, L. J. Glaab, T. L. Vranas, R. G. McSwain, Z. R. Johns, and K. Severance. Sense and avoid for small unmanned aircraft systems. *AIAA Information Systems-AIAA Infotech at Aerospace*, (10), 2017.

[6] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2 edition, 2003.

[7] G. F. Franklin, M. L. Workman, and D. Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., USA, 3rd edition, 1997.

[8] J. James, J. J. Ford, and T. L. Molloy. Learning to Detect Aircraft for Long-Range Vision-Based Sense-and-Avoid Systems. *IEEE Robotics and Automation Letters*, 3(4):4383–4390, 2018.

[9] D. Justino. LiDAR and Camera Sensor Fusion for Onboard sUAS Detection and Tracking. Master's thesis, Instituto Superior Técnico, 2021.

[10] B. Karis. Real shading in unreal engine 4 by. 2013.

[11] Q. Lim, Y. He, and U.-X. Tan. Real-Time Forward Collision Warning System Using Nested Kalman Filter for Monocular Camera. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, dec 2018.

[12] MATLAB. *9.6.0.1072779 (R2019a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[13] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. PIXHAWK : A System for Autonomous Flight using Onboard Computer Vision. pages 2992–2997, 2011.

[14] S. Ramasamy, R. Sabatini, A. Gardi, and J. Liu. LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. *Aerospace Science and Technology*, 55:344–358, 2016.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:779–788, 2016.

[16] F. Schilling, F. Schiano, and D. Floreano. Vision-Based Drone Flocking in Outdoor Environments. *IEEE Robotics and Automation Letters*, 6(2):2954–2961, apr 2021.

[17] S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim : High-Fidelity Visual and Physical. pages 1–14.

[18] J. Shan and C. Toth. *Topographic Laser Ranging and Scanning: Principles and Processing, Second Edition*. CRC Press, 2018.

[19] M. Vrba and M. Saska. Marker-Less Micro Aerial Vehicle Detection and Localization Using Convolutional Neural Networks. *IEEE Robotics and Automation Letters*, 5(2):2459–2466, 2020.

[20] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford. LiDAR and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. *Electronics (Switzerland)*, 7(6), 2018.