Deep learning assisted visual tracking of evader-UAV

Athanasios Tsoukalas[1], Daitao Xing[2], Nikolaos Evangeliou[1], Nikolaos Giakoumidis[3] and Anthony Tzes[1]

***Abstract*— In this work the visual tracking of an evading UAV using a pursuer-UAV is examined. The developed method combines principles of deep learning, optical flow, intra-frame homography and correlation based tracking. A Yolo tracker for short term tracking is employed, complimented by optical flow and homography techniques. In case there is no detected evader-UAV, the MOSSE tracking algorithm re-initializes the search and the PTZ-camera zooms-out to cover a wider Filed of View. The camera's controller adjusts the pan and tilt angles so that the evader-UAV is as close to the center of view as possible, while its zoom is commanded in order to for the captured evader-UAV bounding box cover as much as possible the captured-frame. Experimental studies are offered to highlight the algorithm's principle and evaluate its performance.**

I. Introduction

Long term target detection poses several issues and methods that relying on Deep Learning can discriminate the target without requiring prior frame information [1]. The downside is that given certain visual noise conditions or target scale, the target may not be identified or not be properly classified. This results in a target loss, necessitating the utilization of long term tracking techniques.

Similar works on the concept of Deep Learning based tracking, assisted by other non deep learning oriented methods have recently appeared in the literature. In [2] a homography based method is used to validate the matching of the vehicles identified by Yolo tracker in subsequent frames. In [3] a method based on Homography and Optical flow is used to model the background versus the moving objects and assist the Deep Learning algorithm detection on smaller objects by providing regions of interest of enhanced resolution to the Deep Learning input. In [4] a deep learning approach is used to determine the optical flow between frames and use it in order to model the background versus the foreground.In [5] an assisted Deep Learning technique is used in order to reduce the calculations for object identification and tracking, for efficient use on mobile devices.

Deep Learning have also been used for UAV-detection and tracking. In [6] the YOLOv3 is used running on GPU for UAVs tracking, in order to avoid potential risk of collision. In [7] a segmentation network is employed for detection of stationary aircraft below the horizon, rejection of moving ground vehicles and simultaneous detection of multiple aircraft. In [8] a method generates semi-synthetic training data, for training a deep convolutional network to detect non cooperative UAVs from video sequences. The work in [9] compares different single-shot detectors and presents an optimized architecture. The work in [10] introduces a tiny Single Shot Detector [11], which is optimized to minimize the model size, while retaining detection performance. However, those methods either require powerful GPU devices to achieve high performance or run on CPU at low speed, which is insufficient for on-board experimentation.

In this article, a tracking PTZ camera is used on a pursuer UAV as shown in Figure 1 top left with the purpose to detect and track an evader UAV. To increase the robustness of the Yolo-identification and tracking method, a Correlation [12] technique is employed followed by a second one relying on Homography and Optical Flow between frames [13]. These methods may track the target based on correlation and motion detection when the Yolo tracker fails to identify a target.

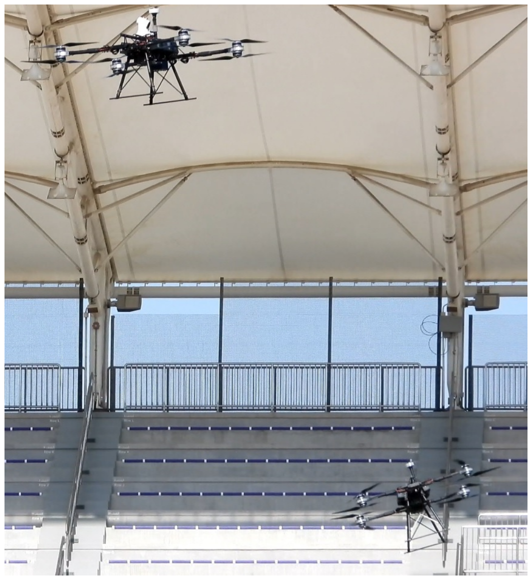

Fig. 1. Tracker and pursuer UAV with mounted PTZ Camera

In our method, various techniques are deployed in order to enhance the performance of the deep learning identification of a target UAV in both computation power and identification accuracy. In order to solve the issue of a missed identification by the Yolo tracker, a long term tracker based on foreground extraction after background subtraction using a homography and optical flow based method is deployed. This enhancement provides possible regions where the UAV target may be located when the target is in motion, and focuses the deep

[1]Robotics and Intelligent Systems Control (RISC) Lab, Electrical & Computer Engineering, New York University Abu Dhabi, Abu Dhabi, 129188, United Arab Emirates.

[2]Department of Computer Science and Engineering, New York University, 11201, New York, USA.

[3]Kinesis Lab facility, Core Technology Platforms(CTP), New York University Abu Dhabi, Abu Dhabi, 129188, United Arab Emirates.

Corresponding author's email: `anthony.tzes@nyu.edu`

learning tracker on those regions resulting in a focused area for UAV identification. In order to maximize performance, the Yolo algorithm is executed in a cropped window from the full image frame, around the region of interest as given by the previous frame identification, either by Yolo or our long term tracking scheme. YOLOv4-tiny is run in CPU mode in order to bypass the CUDA GPU requirement which is not available in the small form PC systems used in our UAVs and in order to decrease the execution time the method is used in combination with CPU related performance enhancements as the core Deep Learning framework.

II. Visual Tracking Algorithm

The target identification process looks at the current full frame using the Yolo tracker. If a target is found and classified as UAV, Yolo is run in a cropped window around the found target in the next frame and the process continues in the cropped space for as long as a target is identified with a certainty exceeding 50%. If the target is lost, then the optical flow and homography based tracking is deployed in the whole frame. If a possible UAV-target is provided, then Yolo runs in the cropped window to validate the target.

A. *Yolo tracker*

In this section, the deep learning based target identification stage based on YOLOv4-tiny is introduced. YOLOv4-tiny method is designed based on YOLOv4 method to make it have faster performance in object detection, thus making it ideal for on-board UAV deployment. The detector consists of three parts: (1) a backbone network for feature extraction, (2) the feature pyramid network for extracting multi-scale features for detection and (3) the YOLO detection head to output the detection results. The YOLOv4-tiny method uses the CSPDarknet53-tiny network as a backbone network to replace the CSPDarknet53 network in YOLOv4 method, in order to boost the detection speed on low-end GPU devices. The backbone network consists of two basic block types: ConvBlock and CSPBlock.The input images are fed into two ConvBlocks, where each block contains a Convolution layer followed by a Batch Normalization Layer, an activation function and a max pooling function. The feature maps with reduced resolution are fed into three CSPBlocks to learn features in different scales. The CSPBlock module divides the feature map into two parallel tracks and combine them in a residual way to increase gradient flow propagation. The CSPBlock module achieves similar learning ability of convolution network in a parameter efficient way. Compared with the original CSPBlock in YOLOv4, it removes the computational bottlenecks and replace the ReLU activation function with a more efficient LeakyRelU function to further simplify the computation process. The LeakyReLU function is: $y_i = \begin{cases} x_i & x_i \geq 0 \\ \frac{x_i}{a_i} & x_i < 0 \end{cases}$, where $a_i \in (1, +\infty)$ is a constant parameter.

Before feeding the features from the backbone network into the detection head, those features are fused by a feature pyramid network in order to extract feature maps with different scales and increase object detection speed. Considering the detection of UAVs in variable sizes, ranging from a short to a long distance, we employ three different scale feature maps 13×13, 26×26 and 52x52 to predict the detection results, under the assumption of a 416x416 image size. To enhance the feature representation ability of the network, we further fuse the features from different scales by inserting an interpolation function. Low level features with higher resolutions are down sampled and appended to the higher level features. Meanwhile, upsampling is deployed in a reversed way. The interpolation and fusion process significantly increases the receptive field with almost no reduction of the network speed.

As in YOLOv4 method, the prediction head follows the anchor-based detection mechanism to match the ground truth bounding boxes with the feature points on feature maps. Even though the size of UAVs can vary in real scenarios, the aspect ratio of the bounding boxes is relatively fixed compared with general objects around the target. Hence, the optimal anchors settings are pre-calculated according to the training data, and then the number of anchor boxes is also decreased from 9 to 2. Specifically, two different scale anchors with aspect ration of 1:1 and 1:2.25 are used for the detection head of each feature map from the pyramid network.

The prediction process of Yolov4-tiny method contains a bounding box regression branch and a classification branch. The feature points on feature map are labeled as positive if the corresponding anchor has an IOU score higher than a threshold. Otherwise, the points are labeled as negative. Meanwhile, the positive points are expected to predict the offsets between the ground truth box and the corresponding anchors. The network predicts the shift distances from the corner position of anchor to the corresponding corner of the ground truth box as a vector $d^* = (d_{x1}, d_{y1}, d_{x2}, d_{y2})$, where $x1, y1, x2, y2$ represent the coordinate of the left-top corner and right-bottom corner respectively. The final prediction bounding box can be calculated by adding the offset to the original coordinate of the anchor.

Finally, the objective function consists of two parts: 1) classification loss and 2) bounding box regression loss. Let $x_{ij} = \{1, 0\}$ be an indicator for matching the i-th default anchor to the j-th ground truth box. c_i^0 and c_i^1 are the negative and positive prediction probability respectively. The classification loss is:

$$\text{loss}_{cls} = -\sum_{i \in \text{Pos}}^{N} x_{ij} \log\left(\hat{c}_i^1\right) - \sum_{i \in Neg} \log\left(\hat{c}_i^0\right)$$

where $t \in \{0, 1\}$ is the class number (non-UAV vs UAV) and $\hat{c}_i^t = \frac{\exp(c_i^t)}{\exp(c_i^0) + \exp(c_i^1)}$. The bounding box loss is defined as:

$$\begin{aligned} \text{loss}_{reg} &= 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \\ &+ \frac{16}{\pi^4} \frac{\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^4}{1 - IOU + \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2}, \end{aligned} \quad (1)$$

where the IOU is the intersection between the ground truth bounding box and the predicted bounding box, $\frac{\rho^2(b,b^{gt})}{c^2}$ denotes the Euclidean distance between the center position of the predicted bounding box and the ground truth bounding box, and c is the minimum diagonal distance of the box that can contains the predicted bounding box and truth bounding box. The training data set is collected from a website which provides 9000+ images containing drones. Also 1000 negative images which does not contain a UAV from the ImageNet data set are randomly selected. During training, the images are resized into 416×416 after random augmentation including flip, rotation and random crop.

B. *Long term motion based tracker*

The YOLO 4 implementation has a few weaknesses that prohibit the sole use of the system for long term tracking, as it is possible to lose the target completely and with local consistency for multiple frames depending on the background of the object, its distance from the camera and the used training set. Inhere, a long term tracker as described in [13] is used in combination with the YOLO tracker to discriminate motion from the background and keep tracking the object up to the point YOLO finds it again; its underlying process is described in Figure 2.

The identification starts by implementing YOLO in the whole first frame. If a target is found then runs the subsequent search in a cropped version of the next frame, around the previously identified target. If the identification in the cropped window fails, the system reverts to running Yolo in the whole image and if a target is not found reverts to running the long term tracker. A correlation filter based tracker is also employed in order to follow the object, initialized around the YOLO found window when confidence is high enough, otherwise around the motion based tracking identified window after a certain confidence.

At all times the correlation tracker runs around the currently found by the Yolo or Optical Flow-Homography methods target and will track the target further if both those methods fail to deliver a result that make sense in the sense of be close to the region of the last know target position. A flow diagram of the method is presented in Figure 2.

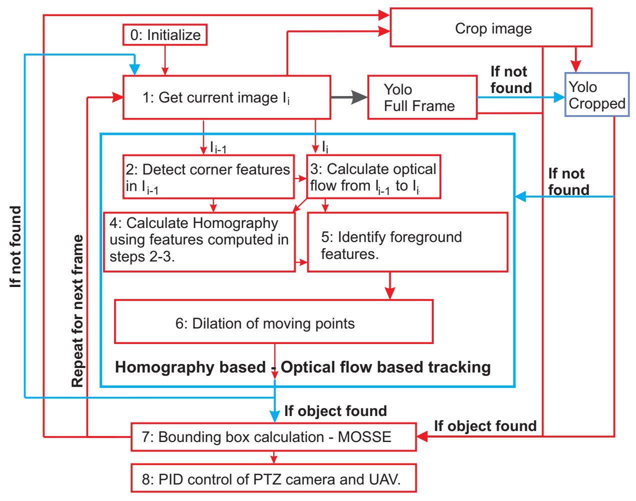
0: Initialize
Crop image
1: Get current image I_i
Yolo Full Frame
If not found
Yolo Cropped
I_{i-1}
I_i
2: Detect corner features in I_{i-1}
3: Calculate optical flow from I_{i-1} to I_i
If not found
4: Calculate Homography using features computed in steps 2-3.
5: Identify foreground features.
6: Dilation of moving points
Homography based - Optical flow based tracking
If not found
Repeat for next frame
If object found
If object found
7: Bounding box calculation - MOSSE
8: PID control of PTZ camera and UAV.

Fig. 2. Long term tracker using Assisted Deep Learning

C. *Object tracking using optical flow and homography*

The long term moving UAV identification is based on optical flow and homography calculations [14] between two sequential frames. A group of prominent image features [15], [16] is first identified in the previous image frame and optical flow is used to estimate the motion of the features in the current frame as shown in Figure 2 in the cyan block.

Assuming the static background holds the majority of the image features, the homography between the feature points in the previous frame and optical flow estimated new positions of these features in the current frame is calculated. The homography is then used to estimate the feature points position in the current frame and the positions are compared to those estimated by the optical flow and the background points are those with a distance below a threshold, as seen in Figure 3.In the figure, the black dots indicate the feature points identified in the previous frame and the cyan and red dots the estimated next frame position of those features based on Optical flow and Homography respectively. The purple dots correspond to the estimated points from the two methods that have a distance below a threshold, indicating that the points belong to the background as their optical flow determined motion closely matches the background motion estimated by the Homography. The points where the distance is above the threshold are considered as the moving object.

The optical flow is calculated using the Lucas - Kanade method [17] with the an extension using a pyramidal scheme with variable image resolutions [18]. The basic optical flow premise is to discover the position of an image feature from previous frame, in the current frame.

The output of the optical flow calculations is the estimated features position in the current frame. An example of the feature matching between the current and previous frames is presented in Figure 4.

D. *Homography & Optical Flow*

The homography or perspective transform can be formulated as a 3×3 operator H on homogeneous coordinates

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = H\tilde{x} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}. \quad (2)$$

The resulting normalized homogeneous coordinates x_i', y_i' are

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}, \quad y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}, \quad (3)$$

while the operator H is calculated such that minimizes the back projection error

$$\sum_i \left(x_i' - \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \right)^2 + \left(y_i' - \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \right)^2.$$

The computed homography matrix is refined further with the non linear Levenberg-Marquardt optimization [19], [20] method to further reduce the re-projection error.When the tracker UAV is moving, static camera based background subtraction algorithms cannot provide an accurate UAV-tracking, and another background subtraction method is

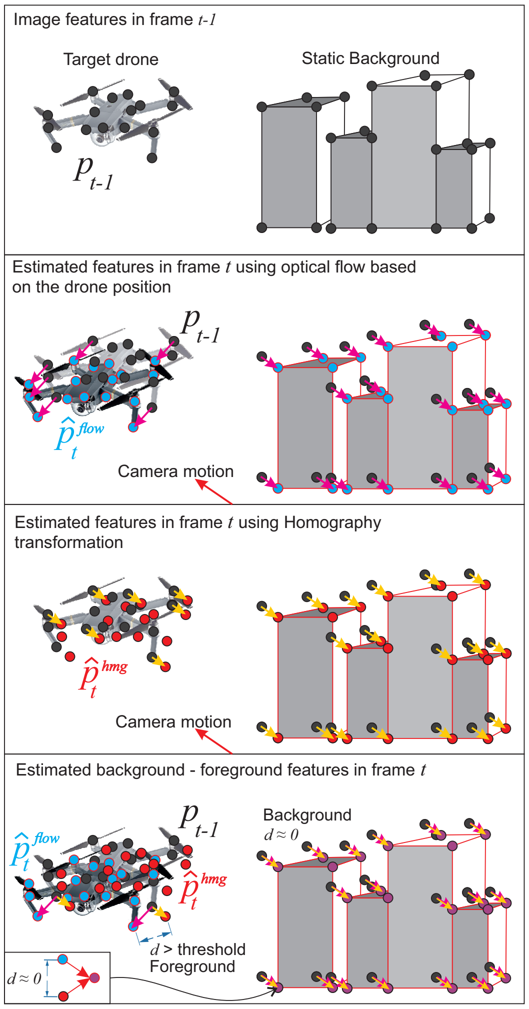
Image features in frame t-1
Target drone
Static Background
p_{t-1}
Estimated features in frame t using optical flow based on the drone position
p_{t-1}
$\hat{p}_t^{flow}$
Camera motion
Estimated features in frame t using Homography transformation
$\hat{p}_t^{hmg}$
Camera motion
Estimated background - foreground features in frame t
p_{t-1}
$\hat{p}_t^{flow}$
$\hat{p}_t^{hmg}$
Background
d ≈ 0
d > threshold
Foreground
d ≈ 0

Fig. 3. Foreground identification method

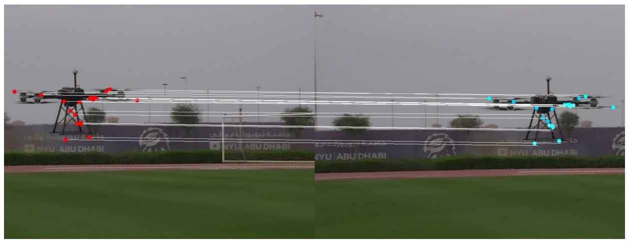

Fig. 4. Image feature matching between successive frames

needed for this case. The method involves the discovery of special image areas with specific characteristics in the camera image, which we note as the image feature set F_{s_t} at time t. The feature set is a collection of pixel points in the camera image $p_t = [px_t, py_t]$. Various algorithms are available for the discovery of image features [15].

For each of the discovered image features of the previous image frame, the optical flow F_{v_t} is derived as a collection of velocity vectors that correspond to each of the image features. For the optical flow estimation of the discovered image features, the OpenCV function “calcOpticalFlowPyrLK” is used. The method operates on a sparse feature set using the iterative Lucas-Kanade method with pyramids and the inputs are the current and previous frames as well as the image features identified from the previous frame. The output is the estimated feature position in the current frame. The algorithm also provides a status flag for each point that describes whether a flow was found for the specific point.

The long term tracking method using the above homography and optical flow principles, differentiates the foreground to the background image features discovered in the previous frame. The estimated background points are given by the transformation of the strong image feature points using a homography matrix and comparison of those positions to the estimated positions by the optical flow method, in the current frame.

Given an image g_{t-1} in time $t-1$, we derive the estimation of the positions of the selected image feature points set F_{s_t} in the image g_t in time t, based on the homography calculated between the strong feature points of the previous frame and their estimated positions by an optical flow method as described in Section II-C in the current camera frame, as described in the previous sections.

The optical flow provides the estimation $\hat{p}_t^{flow}$ of the position of a feature point in the previous frame g_{t-1}, in the current frame g_t for the p_t point in the previous frame and an estimation of the point $\hat{p}_t^{Hmg}$ is also provided from the homography based transformation on the feature points set of the previous frame g_{t-1}. The distance $b_t = ||\hat{p}_t^{flow} - \hat{p}_t^{Hmg}||$ is evaluated and if $b_t(p_t) \geq b_{bkgd}$, $\forall p_t \in F_{s_t}$ (image feature point set), then the point p_t is considered as moving, otherwise as a background one.

The foreground points correspond to those describing the UAV; a dilation operator is followed by the computation of a predicted bounding box around the blob of points that surrounds the UAV, as shown in Figure 5 bottom left window. In the top left window the strong image features are represented by black dots, in the top right window the foreground features are separated from background and are then plotted against a black background with a dilation operation as shown in the bottom left window, then a bounding box is calculated around the dilated blob as shown in bottom right window to identify the moving target. When the background has static objects across a wide range of distances to the tracker UAV, some points may be identified as foreground, thus a second background layer is assumed and the homography is recalculated using this new subset, in order to identify the remaining background points. It should be noted that this layer can be computed extremely fast due to the lack of need to compute the optical flow.

E. Correlation based long-term tracker

One downside of the Yolo and optical flow - homography techniques is that when the tracked object remains static and blends with the background, or is not directly identified by the deep learning network, can be missed, thus a correlation short term tracker relying on the MOSSE tracking algorithm [12] is used to cover for that case, keeping the tracking active by simple correlation check between frames.

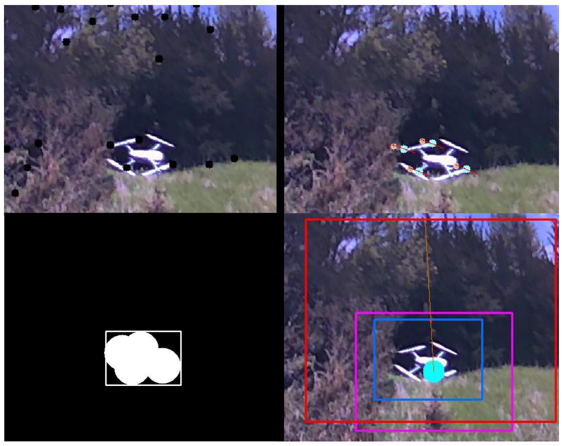
Fig. 5. Estimated image features on tracked UAV

III. Experimental Studies

In the ensuing studies, one evader UAV was moving in a space shown in Figure 6 while the pursuer (tracker) UAV [21] monitored the space. The tracker octarotor UAV is equipped with a 20X optical zoom PTZ-camera and an Intel i7-system running ROS and communicating with the base through Mavlink. Inhere, Yolo runs once in high deep learning identifier resolution (800px) and if there is no result, it runs in the same frame in a lower resolution also in the whole image (416px). If a solution is found in the next frame Yolo runs in a cropped window. If all Yolo methods fail to provide a result, the optical flow method may provide a viable solution and MOSSE runs on the place dictated by it. The MOSSE window will be initiated by either the Yolo or optical flow based tracking window and keep the tracking alive using correlation if both other methods fail to provide a result. The system uses the MOSSE window to move the PTZ camera pan and tilt in order to center the object and order the camera to zoom in when the target UAV is near the center of the camera frame. The results are showing the MOSSE tracking window position and width-height difference to the manually annotated actual UAV bounding box in every video frame. The actual bounding box is considered as the box that fully covers the UAV, including the propellers and landing gear. The pursuer and target UAVs are following the trajectory as shown in Figure 6 in red and blue respectively. A snapshot of the two UAVs flying is shown in Figure 1.

In the following Figure 8 and Figure 9 the blue dots correspond to the identification of the bounding box of the UAV by the Yolo algorithm and the red to the identification by the optical flow and homography based algorithm. The count of the points is mentioned in the ”Yolo” and ”Assist” title in the graphs, correspondingly.

In Figure 8 the error in pixels of the estimated bounding box around the UAV X-Y top left corner from the actual bounding box is shown. The error increases as the zoom factor of the PTZ camera is also increased, due to the larger displacements and bounding box widths when the UAV is taking a much larger area of the camera image. Therefore we use a normalized representation of the error divided by

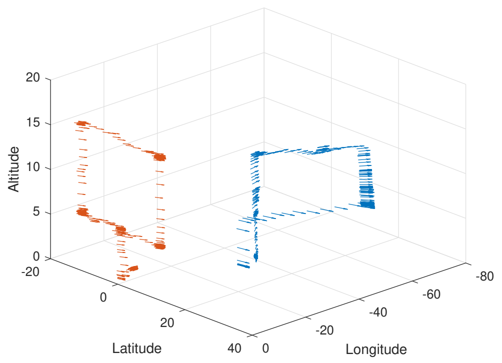

Fig. 6. Tracker (red) and pursuer (blue) UAV trajectories

the zoom factor, in order to correctly evaluate the error in the various zoom stages as if the zoom factor was one. In Figure 9 the error of the estimated bounding box width and height is shown in pixels, while the the zoom factor of the camera in focal length units (mm) is shown in Figure 7.

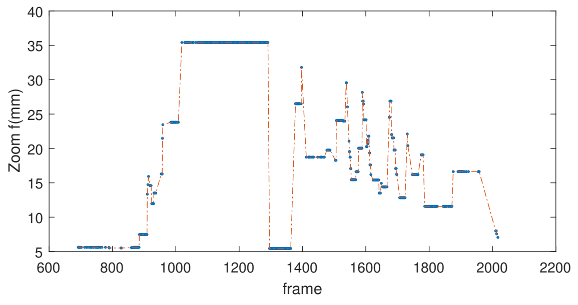

Fig. 7. Camera Zoom factor focal length (in mm)

In the following table a summary of the identification is presented for the method used to realize the long term tracking. A threshold is used to eliminate the outlier estimations based on the previous successful Yolo estimated bounding boxes, so that the PTZ camera will not be directed towards a false positive bounding box. From the results is evident

TABLE I
Enhanced Identification of Flying UAV

Yolo 4	Homography -Optical Flow	% Assisted	Total Frames
831	95	10.25	926

that out of the 1675 captured frames the sole Yolo identifier managed to identify the evader UAV in 89.75% while the homography/optical flow enhanced version assisted in the remaining 10.25% of the frames. Whenever there was no identification of any UAV, the MOSSE algorithm was used to re-initialize the video-tracking while at the same time the camera was zooming out to cover a wider FoV.

The performance of the system varies depending on the identification state, the Frames Per Second (FPS) analyzed on the i7 Intel NUC PC used on the UAV with the camera that performed the online tracking and PTZ camera control,

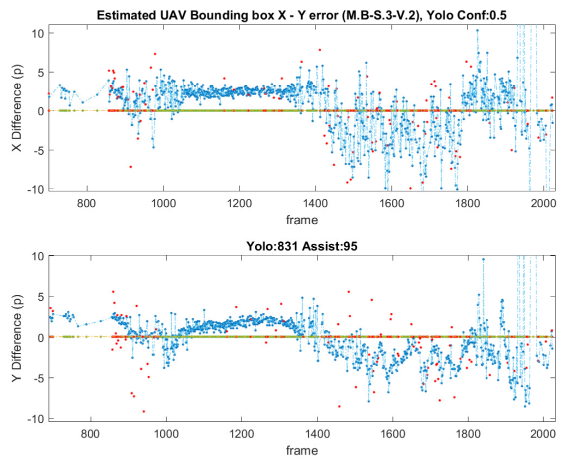

Fig. 8. Error distances of estimated UAV bounding box corner normalized by Zoom factor

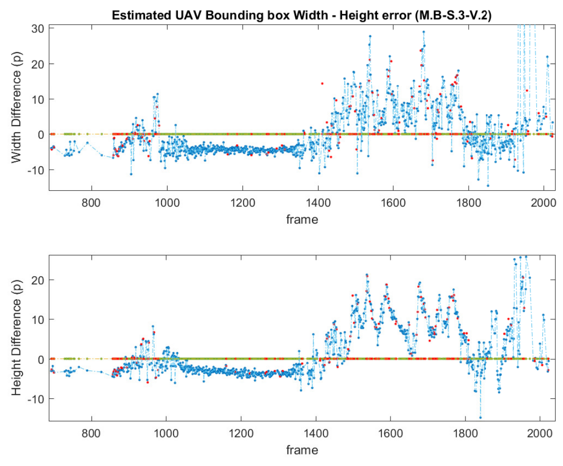

Fig. 9. Error distances of estimated UAV bounding box dimensions normalized by Zoom factor

were at 10-15fps average while a target was searched in the full image and at 15-25fps when the search was done in a cropped window around the previously found target in a previous frame, after a successful identification.

IV. Conclusions

In this work a long term UAVs tracking and detection system was developed for use on board of a UAV equipped with a computer-controlled PTZ camera. The method uses a long term tracker based on Homography and Optical Flow in order to compensate for the case where the Deep Learning detection algorithm may fail, resulting in a more robust tracking method. The PTZ-camera adjusts its parameters so that the tracked UAV spans a large portion of the frame while being centered as much as possible. The selected Deep Learning algorithm is adapted to run efficiently on a CPU-based system. The experimental results showcase how the two long term tracking and Deep Learning methods complement each other for better tracking performance.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[2] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review*, vol. 11, 12 2019.

[3] J. Li, Y. Dai, C. Li, J. Shu, D. Li, T. Yang, and Z. Lu, "Visual detail augmented mapping for small aerial target detection," *Remote Sensing*, vol. 11, no. 1, 2019.

[4] J. Huang, W. Zou, J. Zhu, and Z. Zhu, "Optical flow based real-time moving object detection in unconstrained scenes," *ArXiv*, vol. abs/1807.04890, 2018.

[5] H. Qiu, X. Liu, S. Rallapalli, A. J. Bency, K. Chan, R. Urgaonkar, B. S. Manjunath, and R. Govindan, "Kestrel: Video analytics for augmented multi-camera vehicle tracking," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018, pp. 48–59.

[6] Z.-Y. Huang and Y.-C. Lai, "Image-Based Sense and Avoid of Small Scale UAV Using Deep Learning Approach," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 545–550.

[7] J. James, J. J. Ford, and T. L. Molloy, "Below horizon aircraft detection using deep learning for vision-based sense and avoid," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 965–970.

[8] C. Briese and L. Guenther, "Deep learning with semi-synthetic training images for detection of non-cooperative UAVs," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 981–988.

[9] C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. Bouganis, "DroNet: Efficient convolutional neural network detector for real-time UAV applications," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 967–972.

[10] A. Womg, M. J. Shafiee, F. Li, and B. Chwyl, "Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection," in *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 2018, pp. 95–101.

[11] 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*. Springer, 2016, pp. 21–37.

[12] D. Bolme, J. Beveridge, B. Draper, and Y. Lui, "Visual object tracking using adaptive correlation filters," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 06 2010, pp. 2544–2550.

[13] A. Tsoukalas, N. Evangeliou, N. Giakoumidis, and A. Tzes, "Airborne Visual Tracking of UAVs with a Pan-Tilt-Zoom Camera," in *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems, ROBOVIS 2020, Budapest, Hungary, November 4-6, 2020*. SCITEPRESS, 2020, pp. 90–97.

[14] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[15] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, March 2018, pp. 1–10.

[16] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[17] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, vol. 2, 1981, pp. 674–679.

[18] J. Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[19] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Math.*, vol. 2, no. 2, pp. 164–168, 1944.

[20] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[21] N. Evangeliou, A. Tsoukalas, N. Giakoumidis, S. Holter, and A. Tzes, "Development of a Versatile Modular Platform for Aerial Manipulators," *Service Robotics, IntechOpen*, 2020.