Various Object Trackers for UAV Chasing of Counter UAV Systems

Kyubin Kim, Jaehong Kim, Han-Gyeol Lee, Jingon Joung
School of Electrical and Electronics Engineering, Chung-Ang University
Seoul 06974, South Korea
{kkb0623; kjhct9606; forener; jgjoung}@cau.ac.kr

***Abstract*—In this study, we examine eight different object trackers for UAV chasing. The chasing algorithm employs version three of you only look once version (YOLOv3) and the considered object tracker. Following an object tracking benchmark, success rate, precision, and complexity were compared. From the numerical results, it was shown that the proposed UAV chasing algorithm with a tracker achieves improved chasing performance with less computational complexity.**

***Index Terms*—Unmanned aerial vehicles (UAV), object tracker, YOLO**

I. Introduction

The unmanned aerial vehicle (UAV) chasing is one of the essential functions of counter UAV systems (CUSs) [1], in which a pursuer/counter UAV (pUAV) detects and chases an evader/attack UAVs(eUAV). A vision-based method processes every image of a moving eUAV to detect, classify, and chase. Here, a region with convolution neural network (R-CNN)-based object detection is the most well-known object detection algorithm [2], yet, its high complexity hinders CUS for detecting eUAV in real time. To circumvent this issue, one-stage detectors, e.g., you look only once (YOLO) and a single shot detector (SSD), are employed to process object localization and classification at once [3], [4]. Specifically, YOLOv3, the third version of YOLO, is one of the most popular algorithms because of its fine performance and easy implementation. YOLOv3 uses darknet-53 to extract target object features and constructs three different scaled feature maps. Therefore, YOLOv3 can detect small-size objects like UAVs. On the other hand, object trackers (e.g., trajectory tracking, correlation filter, neural network) were developed rapidly to track target objects.

In this study, various object trackers have been examined for chasing eUAV. A vision-based UAV chasing algorithm is proposed by combining YOLOv3 with various object trackers. Performances of proposed algorithms are evaluated and discussed by object tracking benchmark (OTB) performance metrics [5]. As the result, it is shown that the proposed algorithms can improve detection performance and reduce the complexity of a system with YOLOv3 only. An extended version of this paper is available in [6].

The remainder of this paper is organized as follows. Section II describes the system model of the proposed UAV chasing system, and Section III introduces the performance evaluation metrics. Simulation results are presented in Section IV. Section V concludes this paper.

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-00874, Development of Next Generation Wireless Access Technology Based on Space Time Line Code, 25%, and 2022-0-00635 Development of 5G industrial terminal technology supporting 28GHz band/Private 5G band/NR-U Band, 25%), by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT) (2021R1A4A2001316 & 2022R1A2C1003750)

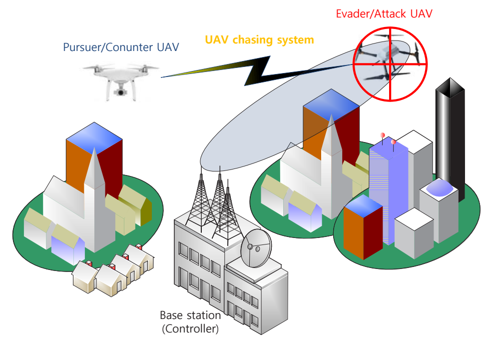

Fig. 1. Counter UAV system with chasing single eUAV by single pUAV.

II. Proposed UAV Chasing System

A static ground station platform is considered for a UAV chasing system as shown in Fig. 1, in which a single pUAV chases a single eUAV following the control of the ground station after detecting and recognizing the eUAV. The chasing scenario is as follows: i) pUAV takes eUAV images by using the equipped camera in real-time and sends the images to the controller at the ground station, ii) the controller processes a deep learning-based object detection algorithm, i.e., a you only look once (YOLO), to detect the eUAV, iii) the controller sends control signals to pUAV to chase the eUAV. Concretely, a YOLOv3 based UAV detection is considered and various OpenCV trackers are examined to chase the eUAV after the detection with low computational complexity, as briefly summarized in Algorithm 1, where f and c are flag and counter, respectively.

First, YOLOv3 detects eUAV from the input image. If YOLOv3 succeeds to detect eUAV and returns coordinates of the bounding box, the tracker gets ready to track eUAV at the next image($c = 1$). When the next image comes through, the object tracker tracks eUAV until it is failed to track and

Algorithm 1 Proposed UAV chasing algorithm.

```
f = 0 and c = 0
while true do
    frame ← new image
    if mod(c, 20) ≠ 0 or f = 0 then YOLOv3(frame)
        if YOLOv3 detects eUAV then f = 1 end if
    otherwise tracker(frame)
        if tracker fails to track eUAV then f = 0 end if
    end if
    c ← c + 1
end while
```

return the bounding box. After the tracker failed, the YOLOv3 recovered tracking by detecting eUAV. However, most of the object tracker has a fixed bounding box regardless of the location of the eUAV until YOLOv3 recovers the tracking. To solve this problem, we periodically run YOLOv3 in every 20 images and configure the bounding box to fit the real-time eUAV position. For this overall UAV chasing algorithm, we are going to use various OpenCV trackers as follows:

- **Boosting**: The boosting tracker can adapt its classifier thus being good to handle appearance changes of the object (AdaBoost). Furthermore, the online trained classifier uses background images as negative examples in the update [7].
- **MIL**: The multiple instance learning (MIL) tracker uses a bag of positive and negative images for training classifiers and updates the next location of the target by using a greedy strategy. MIL tracker can track well when a partial occlusion exists [8].
- **MOSSE**: The minimum output sum of the squared error (MOSSE) tracker models target appearance by adaptive correlation filters, uses convolution for tracking, and is robust to variation of light, scale, and pose. However, the MOSSE tracker provides poor tracking performance when the object's appearance changes [9].
- **Median Flow**: The Median Flow tracker uses forward-backward trajectory error to track the object, and is good to predict a motion. However, it is hard to track a high mobility object [10].
- **TLD**: The tracking-learning-detection (TLD) tracker decomposes the process of long-term tracking as tracking, learning, and detection. Tracker follows object from frame to frame, detector localizes all objects that have been observed in past, learning estimates detector's error and updates to avoid errors [11].
- **KCF**: The kernelized correlation filter (KCF) uses a circulant matrix to deal with redundant training data. By using this method, the KCF tracker can reduce data storage and complexity. KCF tracker can't recover tracking after failure or occlusion [12].
- **GOTURN**: The generic object tracking using regression networks (GOTURN) tracker uses offline-trained CNN networks that are trained with many video sequences and images. GOTURN tracker is good at tracking various objects, but it only works well when target objects are included in off-line trained neural network [13].
- **CSRT**: The channel and spatial reliability tracking (CSRT) uses spatial and channel reliability to constrain tracking areas. By using this method, the filters can enlarge the search area and the tracker has the robustness to track non-rectangular shaped targets [14].

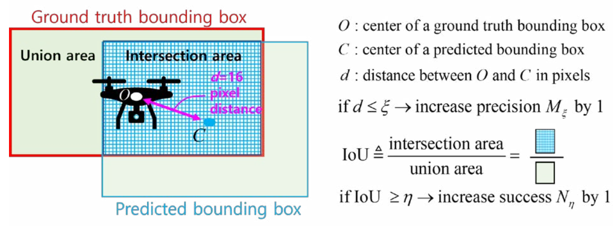

Fig. 2. Examples to depict precision P_ξ and success $S(\eta)$.

III. Performance Metrics

To evaluate the performances of each UAV chasing algorithm, the precision and success rate from OTB are employed.

The precision rate is defined as a ratio of the number of images (frames) as follows:

$$P_\xi = \frac{M_\xi}{K}, \tag{1}$$

where M_ξ represents the number of images whose bounding box center, C, is close to the center of ground truth, O, which is shorter than a predefined threshold ξ-pixel distance; and K is the total number of the predicted frames. Following [5], we set $\xi = 20$-pixel distance in this study. For example of the precision in Fig. 2, the distance d between O and C is 16-pixel distance, which is shorter than ξ, and thus, this predicted frame is included in M_ξ-pixel distance. As we can see in Fig. 2, however, the precision value does not capture the accuracy of distance between O and C in a perpendicular direction to the boxes.

To resolve this ambiguity, a success rate is employed. Let us define an intersection of union (IoU) value as a ratio of the intersection area to the union area of ground truth and predicted bounding boxes as depicted in Fig. 2. The success rate is then defined for the given threshold η as follows:

$$S(\eta) = \frac{N_\eta}{K}, \tag{2}$$

where N_η is the number of images whose IoU value is greater than η. To represent the success rate for various η, area under the curve (AUC) score, i.e., $\bar{S}$ of $S(\eta_l)$ over various $\eta_l = \frac{l}{10}$, $l \in \{1, 2, \ldots, 10\}$, is employed, which is calculated by the trapezoidal method as follows:

$$\bar{S} = \frac{1}{10} \sum_{l=0}^{9} \left(\frac{S(\eta_l) + S(\eta_{l+1})}{2} \right). \tag{3}$$

In Fig. 3, the AUC of TLD is depicted as example of how to calculate the sum of trapezoidal areas according to (2) and (3).

TABLE I

PERFORMANCE OF UAV CHASING ALGORITHMS : $\bar{S}$ IS AUC SCORE OF SUCCESS RATE, P_{20} IS PRECISION RATE WITH $\zeta = 20$, C IS COMPLEXITY.

	YOLOv3-416			+Boosting			+MIL			+MOSSE			+Median Flow			+TLD			+KCF			+GOTURN			+CSRT		
Dataset	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C	$\bar{S}$	P_{20}	C
E1	0.451	0.699	0.074	0.49	0.808	0.028	0.496	0.825	0.055	0.363	0.557	0.035	0.452	0.805	**0.024**	0.330	0.741	0.064	0.438	0.708	0.027	**0.556**	0.811	0.045	0.511	**0.827**	0.036
E2	0.563	0.770	0.071	0.582	0.814	0.027	0.532	0.838	0.052	0.363	0.468	0.037	0.529	0.795	0.023	0.405	0.768	0.067	0.523	0.786	**0.022**	0.610	0.849	0.04	**0.626**	**0.865**	0.029

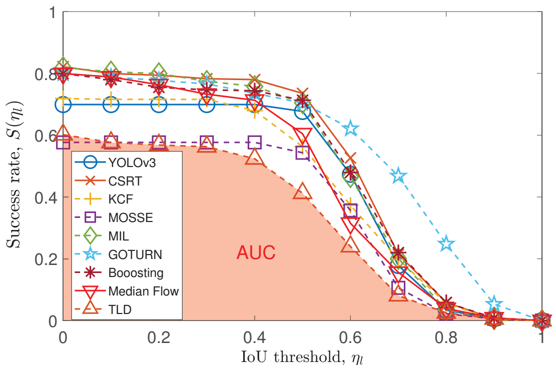

Fig. 3. Example of AUC calculation by (3). Here, the AUC of TLD is depicted in OPE success rate plot of E1.

IV. SIMULATION AND DISCUSSION

We evaluate UAV chasing performances by two datasets, namely, E1 and E2. The E1 represents an occasional obstacle environment and consists of 360 low-resolution images (i.e., 640×480, $K = 360$). The E2 represents the environment with variable illumination, consisting of 370 high-resolution images (i.e., 960×720, $K = 370$). The eUAV images in both datasets are obtained from iPhone 11. For the simulation, we use intel core i7-8700K CPU, NVIDIA GeForce GTX 1070 Ti GPU, 32GB RAM, and python with OpenCV. The model of eUAV is DJI Tello edu. YOLOv3 is trained, validated, and tested by a total of 6063 images of eUAV (4000 for training 1000 for validation, and 1063 for test, respectively.). For the training, the Adam optimizer is used with an initial learning rate 0.001. The maximum epoch is 200, while the early stopping method stops the training at 110 epochs. The complexity C refers to the average time taken to process one image frame and its unit turns to be $\mathrm{s/frame}$.

The UAV chasing performances are evaluated by $\bar{S}$ and P_{20} through one pass evaluation [5]. The result of the UAV chasing simulation is summarized in Table I. The underlined values indicate the higher performance than YOLOv3 of each dataset and the bolded values indicate the best performance. In E1, since the occlusion and low-resolution environment, the YOLOv3 can not track eUAV well. However, among the proposed UAV chasing algorithms, the YOLOv3 with GOTURN shows the highest $\bar{S}$, and CSRT shows the highest P_{20}. Similarly, in E2, the YOLOv3 shows the lower $\bar{S}$ and P_{20} than some proposed UAV chasing algorithms due to the illumination variation. Among the UAV chasing algorithms, the YOLOv3 with CSRT shows the highest $\bar{S}$ and P_{20} for E2. Unlike YOLOv3, the object trackers can utilize the previous tracking information in current tracking, which can compensate YOLOv3 when the current image is under bad condition. Also, the proposed UAV chasing algorithms show lower computation complexity than YOLOv3, because of the object trackers have less computation complexity than the deep learning-based YOLOv3.

V. CONCLUSION

This study proposed the UAV chasing algorithms, combining YOLOv3 with object tracker, for the UAV chasing in CUS. The success and precision rates were calculated to evaluate the eUAV chasing performance. The numerical results showed that the proposed UAV chasing algorithm can effectively improve the success and precision rates of YOLOv3 with less complexity.

REFERENCES

[1] H. Kang, J. Joung, J. Kim, J. Kang, and Y. S. Cho, "Protect your sky: A survey of counter unmanned aerial vehicle systems," *IEEE Access*, vol. 8, pp. 168 671–168 710, Sep. 2020.

[2] O. Hmidani and E. M. Ismaili Alaoui, "A comprehensive survey of the R-CNN family for object detection," in *Proc. Int. Conf. Adv. Commun. Technol. Netw. (CommNet)*, Dec. 2022, pp. 1–6.

[3] A. Tripathi, M. K. Gupta, C. Srivastava, P. Dixit, and S. K. Pandey, "Object detection using YOLO: A survey," in *Proc. Int. Conf. Contemp. Comput. Inform. (IC3I)*, Mar. 2022, pp. 747–752.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, Dec. 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0_2

[5] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[6] K. Kim, J. Kim, H.-G. Lee, J. Choi, J. Fan, and J. Joung, "UAV chasing based on YOLOv3 and object tracker for counter UAV systems," *IEEE Access*, vol. 11, pp. 34 659–34 673, Apr. 2023.

[7] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting." in *Proc. British Mach. Vis. Assoc. (BMVA)*, Sep. 2006, pp. 47–56. [Online]. Available: http://dblp.uni-trier.de/db/conf/bmvc/bmvc2006.html/GrabnerGB06

[8] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Aug. 2009, pp. 983–990.

[9] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Aug. 2010, pp. 2544–2550.

[10] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Proc. Int. Conf. Pattern Recognit.*, Oct. 2010, pp. 2756–2759.

[11] ——, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jan. 2012.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[13] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," Aug. 2016. [Online]. Available: https://arxiv.org/abs/1604.01802

[14] A. Lukežic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nov. 2017, pp. 4847–4856.