A Hybrid Method for Tracking of Objects by UAVs

Hasan Saribas[1], Bedirhan Uzun[2], Burak Benligiray[1], Onur Eker[2], Hakan Cevikalp[2]
[1]Eskisehir Technical University, [2]Eskisehir Osmangazi University
[1]2 Eylul Kampusu, 26470, Eskisehir, Turkey, [2]Meselik Kampusu, 26480, Eskisehir, Turkey
{hasansaribas48, eee.bedirhan, bbenligiray, onureker34, hakan.cevikalp}@gmail.com

Abstract

Object tracking remains one of the fundamental problems of computer vision since it becomes difficult under some realistic conditions such as fast camera movement, occlusion and similar of objects to the tracked target. As a real-world application, tracking objects using cameras mounted on unmanned aerial vehicles (UAVs) has become very popular. With the increasing availability of small single board computers with high parallel processing power capabilities, tracking of objects by using onboard computers within UAVs in real time has become feasible. Although these onboard computers allow a wide variety of computer vision methods to be executed on a UAV, there is still a need to optimize these methods for running time and power consumption. In this paper, we propose a hybrid method for a UAV to detect and track other UAVs efficiently. To detect the target UAV at the beginning of the video and in the case where the tracked UAV has been lost, we use the deep learning-based YOLOv3 and YOLOv3-Tiny models, which provide one of the best trade-offs between speed and accuracy in the literature. To track the detected UAVs in real time, a kernelized correlation filter is used. Combining these two methods provides high accuracy and speed even on onboard computers. To train the neural nets and test our method, we have collected a new dataset composed of videos of various UAVs in flight, captured from another UAV. The performance of the proposed method has been compared with other state-of-the-art methods in the literature on this dataset. Additionally, we also tested the proposed trackers on aerial videos captured from UAVs. Experimental results show that the proposed hybrid trackers achieve the state-of-the-art performance on all tested datasets. The code is available at https://github.com/bdrhn9/hybrid-tracker.

1. Introduction

Visual tracking is the task of determining the spatial location of a specific object across sequential frames captured by a camera. Tracking is one of the fundamental problems of computer vision, and therefore it has been studied by researchers for a long time. Although these studies have resulted in methods that perform satisfactorily under controlled conditions, it is difficult to claim the same for difficult conditions such as fast moving cameras and objects, occlusion, and complex backgrounds. Therefore, tracking remains an open problem of computer vision.

Visual tracking has many real-world applications, mostly in monitoring and surveillance. Due to their medium-low flight altitude, unmanned aerial vehicles (UAV) are an ideal platform to mount cameras on for these purposes. With the recent emergence of single-board computing modules that can be mounted on UAVs, onboard visual tracking has become a feasible option. The resulting implementations can be utilized in surveillance security systems, for traffic monitoring or for some military applications.

The majority of the earlier studies followed either a generative or a discriminative approach for tracking. Generative approaches use the location of the object in the previous frame to infer its location in the current frame [27, 33]. This is done by producing various windows in the current frame around the location of the object returned in the previous frame, and choosing the window that delivers the highest similarity to the object representation according to the metric of choice. These methods require the location of the object to be given at the beginning and are not trained beforehand.

Unlike generative methods, discriminative methods are trained in advance to recognize the object to be tracked [35, 15, 18]. The trained model can then be used to detect the object and track it across the frames. Discriminative methods tend to be able to deliver a higher accuracy compared to generative methods. However, they also have some comparative shortcomings, such as only being able to track objects that they were trained for, not being able to recognize the object in the case of appearance transformations unless the model is updated, and running slower compared to generative methods.

The most recent tracking methods that deliver state of

Figure 1. Example images from the UAV Tracking Dataset which is created to test the UAV tracking methods.

the art performance use either deep neural nets or correlation filters. Although later layers of deep CNN models are successful at producing high-level representations, they fail at capturing fine details. In comparison, earlier layers of CNN architectures are more sensitive towards the changes in appearance of the objects, which provide better localization. For this reason, shallower CNN architectures are preferred in tracking applications [29, 22, 26]. Furthermore, it is not feasible to train deeper CNN architectures with a lot of parameters online in real-time because of speed issues.

The correlation filter-based tracking methods on the other hand solve a ridge regression problem in the frequency space to locate the object to be tracked [4, 16, 9]. To locate the object in the next frame, the learned filter is applied around the region of interest and the candidate that returns the highest correlation filter response is chosen. Then, the filter is updated with this new position. This approach is considerably faster compared to deep learning-based methods.

Visual object detection is determining the locations and scales of examples of a general object type in an image. There are two major differences between the visual object detection and tracking problems. In detection, all instances of a general object type are desired to be localized, while tracking is focused on a specific object instance. Furthermore, since tracking is done on sequential frames, information from previous frames can be leveraged to better localize the object in the current frame. This is not possible in the detection problem, as the images are typically independent, and not sequential frames of a video. Similar to visual tracking, visual detection is a challenging problem under difficult conditions. Various transformations such as scale and viewing angle differences cause the objects to appear very differently on the image. Moreover, some objects, such as animals, can undergo non-rigid transformations, and modeling all resulting appearances as the same object type is hard. In addition to these, similar to all vision problems, various illumination conditions and complex backgrounds make detection much more difficult.

Object detection methods are typically composed of two stages: extracting features that represent the objects and learning to detect objects by using a machine learning algorithm. HOG [5], LBP [1] and CNN [11] features are commonly used for detection. After extracting these features, the second part is the training of detectors for detection. The classic approach to detection is to train a classifier such as support vector machines (SVM), artificial neural networks (ANN), nearest neighbors or decision trees to distinguish examples of objects from examples of the background using the extracted features, and using this classifier to classify candidate regions in the image. Alternatively, deep learning-based methods provide state of the art results by training a CNN and a detector in an end-to-end fashion, and these methods can run in real-time on GPUs. Some popular examples to this approach can be given as YOLO [30], SSD [24] and Faster R-CNN [32].

In this study, we propose to utilize a detector alongside a tracker to mitigate its shortcomings. Specifically, the detector acts as a localization initializer and a self-correction mechanism whenever the tracker loses the target. The resulting method both delivers good tracking accuracy, and is also lightweight enough to run onboard within a UAV. The performance of this method at tracking other UAVs is tested with a dataset that we have created. Additional experiments are provided for tracking of ground objects in videos captured from UAVs. The results of these experiments indicate that the proposed method is suitable for tracking applications by using onboard computers with UAVs.

2. Proposed Method

In this study, we propose a hybrid method to track UAVs in real-time. To this end, the kernelized correlation filter (KCF) [16], which is particularly fast at tracking, is used along with the accurate and relatively fast detection models YOLOv3 and YOLOv3-Tiny [31]. To train the YOLO models and test the proposed methods, we have created a new dataset of videos with various kinds of UAVs (see Fig. 1).

2.1. Kernelized Correlation Filter Tracker

Correlation is a metric of similarity between two patterns, where more similar patterns are more highly correlated. Visual tracking methods based on correlation filters are designed to produce the highest response when the filter is applied on the object to be tracked. To reduce the computational complexity of the learned filter, and thus to speed up the method, the properties of circulant matrices [10] are used for KCFs [16]. In addition, HOG features [5] are used to improve tracking accuracy.

Now, let y_i represent a target and $f(\mathbf{z}) = \mathbf{w}^\top \mathbf{z}$ be the function that minimizes the squared error over samples $\mathbf{x}_i$ and their regression targets y_i. The KCF briefly solves the following ridge regression optimization problem,

$$\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i) - y_i)^2 + \lambda ||\mathbf{w}||^2, \tag{1}$$

where λ is a regularization parameter that controls overfitting. The solution of the problem in the frequency domain corresponds to

$$\widehat{\mathbf{w}} = \frac{\widehat{\mathbf{x}}^* \odot \widehat{\mathbf{y}}}{\widehat{\mathbf{x}}^* \odot \widehat{\mathbf{x}} + \lambda}, \tag{2}$$

where $\widehat{\mathbf{w}}$ indicates the Fourier transform of $\mathbf{w}$, $\widehat{\mathbf{x}}^*$ is the complex conjugate of $\widehat{\mathbf{x}}$, and $\odot$ denotes the element-wise product. We can easily obtain $\mathbf{w}$ in the spatial domain by using inverse Discrete Fourier Transform (DFT). KCF also utilizes the kernel trick to improve the accuracy.

Since the problem is solved in the frequency domain, the method is extremely fast which makes the method ideal for practical applications. Consequently, this method is one of the fastest in the literature, robust against to translation and scale variances, and provides high accuracy in general. However, it only searches for the object in a region of interest, and hence it fails when the object is occluded or has left the field of view, as it does not have a self-correction mechanism. In addition, as in the other correlation filter based trackers, it is not straightforward to estimate aspect ratios of the target object bounding boxes (To the best of our knowledge, there is only a single study [21] that addresses this problem).

2.2. YOLO Detector

Multi-stage object detectors generate object proposals and classify these [13]. Although these object proposals can be generated using another method, a neural net can be trained for this purpose as well [32]. Unlike these methods, YOLO is a single-stage object detector network. The model predicts bounding boxes and their probabilities in a single pass. The probabilities are then used to weight the bounding boxes. While multi-stage object detectors tend to run a pass for every candidate, YOLO does a single pass for the whole image, which results in a considerable speed-up [30].

We should emphasize that YOLO is a fast object detection method, rather than an object tracking method. It is not possible to use YOLO by itself for tracking a specific object with other objects of the same class in the scene, as it would not attempt to distinguish the target object from the others. Additionally, it is not designed to return a detection response for all frames, which may result in a discontinuity in tracking.

With UAV-specific applications of vision, running time, which also affects power consumption, becomes much more critical. While YOLO runs relatively fast on high-end GPUs compared to multi-stage detectors, it is not comparable to KCF, which can run with 100+ FPS on a CPU. However, this slowness can be alleviated somewhat by using the smaller version of YOLO, YOLO-Tiny, which has a smaller number of convolutional layers. In this study, we use YOLOv3 and YOLOv3-Tiny, which include minor improvements over the original model [31].

2.3. The Proposed Hybrid Tracker

YOLO object detector is good at returning the locations of the general object categories, but it cannot be used entirely for tracking. Because, the same kind of object instances such as aeroplanes and pedestrians can be considered as both the target and background in a given frame. It also becomes slow for onboard computers. KCF tracker on the other hand needs initialization in the first frame and in case of tracking failures. It has also shortcoming that it is not robust to changes in aspect ratios of the target bounding boxes. Considering the advantages and limitations of

these two methods described above, it can be seen that they complement each other. Therefore, we introduce a hybrid tracker to take advantage of the benefits of these two methods. The proposed hybrid tracker (UAVH) uses YOLO to detect the object in the first frame and recover in the cases where the tracker fails, and it uses KCF to maintain the tracking of the object. The resulting method is a hybrid tracker that embodies the advantages of both approaches.

We first need to train a YOLO model to detect objects in order to use the proposed hybrid method. To this end, the YOLO model is trained to detect UAVs with a dataset composed of UAV images in one of our experiments and we used MS-COCO dataset for training YOLO in the other tests. For optimal performance, we compiled the original C implementation of the YOLO method as a dynamic library and used the related functions in our Python code[1].

From this point on, the terms tracker and detector are used for the KCF and YOLO methods, respectively. Once the YOLO is trained for the specific categories that will be used for tracking, we need a mechanism to switch between the detector and tracker methods.To this end, we rely on the tracker scores: If the filter score of the KCF tracker is below a determined threshold value, the YOLO detector is called to detect all UAVs (or any other tracked object instance) in the image. Next, we have to decide if the object of interest is in the scene or not. We rely on object detection scores for this purpose and if the scores are higher than a selected threshold, we consider the returned positions as potential tracked object locations. To determine the final position of the tracked object, we use the distances from the last known position of tracked object and the candidate locations. As seen in Eq. 3, among the k candidate detections (P_{D_i}), the one with the highest IoU (Intersection-over-Union) ratio with the latest output of the tracker is chosen as the new position of the target and then it is used to train the KCF tracker.

$$\max_{i \in k} \left(\frac{|P_T \cap P_{D_i}|}{|P_T \cup P_{D_i}|} \right) \tag{3}$$

Note that sudden movements may cause the detected bounding boxes to not overlap with the earlier tracked object position. In that case, as seen in Eq. 4, the Euclidean distances between the center coordinates of the k candidate detections (x_{D_i}, y_{D_i}) and the center coordinates of the latest output of the tracker (x_T, y_T) are compared, and the position with the closest distance is chosen as the new target position.

$$\min_{i \in k} \left(\sqrt{(x_T - x_{D_i})^2 + (y_T - y_{D_i})^2} \right) \tag{4}$$

By implementing the above procedure, the tracker is enabled to recover from a failure and update the aspect ratio dynamically. Since the detector is only called when the filter score of the tracker is below a certain threshold, the proposed method runs faster than YOLO. As a result, the proposed method inherits its speed, and the abilities to lock on a specific object instance and produce a continuous output from KCF, and its abilities to recover from failure and update the location and aspect ratio of the bounding box from YOLO.

[1]Our code is available at https://github.com/bdrhn9/hybrid-tracker.

Table 1. The comparison of the methods on the UAV Tracking Dataset. Red, green and blue indicate the best results in descending order.

Method	Success	Precision	FPS
UAVH (ours)	**0.561**	**0.773**	53.5
UAVH-Tiny (ours)	**0.524**	**0.737**	**69.2**
YOLOv3 [31]	**0.485**	**0.653**	16.5
YOLOv3Tiny [31]	0.461	0.630	47.1
CSRT [25]	0.232	0.402	**111**
TLD [18]	0.205	0.265	20.1
MIL [2]	0.215	0.379	12.3
KCF [16]	0.126	0.196	**115**
BACF [19]	0.213	0.339	25.8

3. Experiments

Although there are many datasets to test methods for tracking objects in videos such as OTB (Object Tracking Benchmark) [34] and VOT (Visual Object Tracking) [20], these datasets are not solely composed of UAV videos. For this reason, we labeled 7500 frames to create a dataset for our specific application, which is tracking of UAVs by using a camera mounted on another flying UAV. 2000 frames from this dataset are used to train the YOLOv3 models, while the remaining 5500 frames from 15 videos are used for testing. In addition to this dataset, we have also conducted experiments on the UAV123 and UAV20L datasets [28], which are composed of videos shot from UAVs. Differently from our dataset, the objects to be tracked in these videos are not UAVs, but various objects on the ground. The YOLO models used in the UAV123 and UAV20L experiments were pretrained with the COCO dataset [23].

For a fair comparison between the methods, the OTB evaluation protocol is used [34]. In this protocol, the results are presented with both success and precision scores. While calculating the success score, the overlap between the ground-truth bounding box (R_{GT}) and the bounding box predicted by the method (R_P) is considered.

$$IoU = \frac{|R_{GT} \cap R_P|}{|R_{GT} \cup R_P|} \tag{5}$$

When the IoU ratio is above a certain threshold value, the result is considered to be successful, and the success score

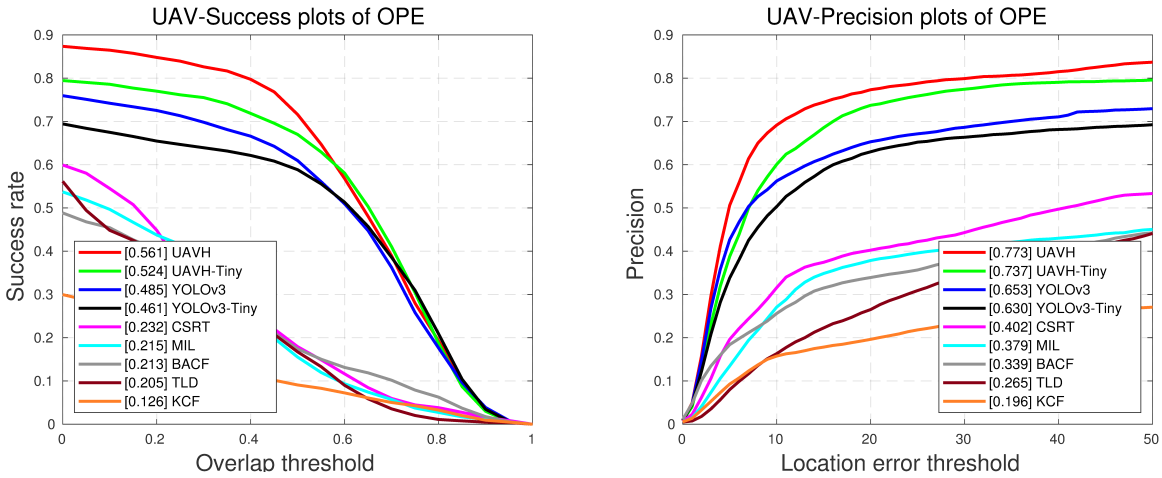

Figure 2. Success and precision scores of the methods on the UAV Tracking Dataset using one pass evaluation (OPE) protocol.

is calculated as the ratio of the number of frames that the method was successful to the total number of frames. This calculation is repeated for all videos in the dataset to calculate an overall success score for the dataset. To calculate the second metric, precision, the distance between the ground-truth center of the bounding box is compared with the bounding box returned by the method. The ratio of the number of frames where this distance is less than 20 pixels to the total number of frames gives the precision score.

The experiments are repeated with both YOLOv3 and YOLOv3-Tiny. YOLOv3-Tiny is chosen particularly to be able to run the proposed method on a Jetson TX2 onboard within a UAV, and the resulting method is denoted as UAVH-Tiny. To determine the thresholds for YOLO and KCF, we experimented on a small validation dataset and set the YOLO threshold score to 0.5 and KCF filter score to 0.45. The models are trained with an NVIDIA Quadro P5000 GPU and the experiments are done on a notebook computer with an Intel Xeon E-2186M CPU and NVIDIA Quadro P4200 GPU. We also run the tests of the proposed trackers on a Jetson TX2 module.

3.1. UAV Tracking Dataset

The proposed methods, UAVH (using YOLOv3) and UAVH-Tiny (using YOLOv3Tiny), are trained with the training set that we have collected. These are compared on our test set with the following recent studies that have achieved high tracking performance: BACF [12], KCF [16], TLD [18], MIL [3], CSRT [25], YOLOv3 [31], and YOLOv3-Tiny [31]. It should be noted that all these methods are trackers except YOLOv3 and YOLOv3-Tiny.

The success and precision scores of the tested methods on our dataset are given in Table 1 and Fig. 2. The proposed hybrid methods deliver the highest success and precision scores, while being the third fastest method. It can be seen that the faster KCF and CSRT methods have performed very poorly on our dataset. If we compare the proposed hybrid method to its components, YOLOv3 and KCF, we can see that it has outperformed both of them.

3.2. UAV123 Dataset

The UAV123 dataset consists of 123 challenging high-resolution aerial video sequences [28]. These video sequences are classified under 12 different scenarios, such as fast moving objects, substantial changes in aspect ratio and scale, illumination variation, viewpoint change, similar object and so on. In both the UAV123 and UAV20L datasets, we compared the proposed methods with state of the art methods in the literature (some only appear on the plots): Note that for the UAV123 and UAV20L experiments, the proposed method is pretrained with the COCO dataset.

The success and precision plots of the methods are given in Fig. 3. It can be seen that UAVH provides the best results again, but UAVH-Tiny performs comparatively worse. See Table 2 for various performance metrics. UAVH is both the best performing method, and it is also in the best top-3 methods in terms of frame rates. Although UAVH-Tiny is not in the best top-3 methods in success and precision, it runs significantly faster than UAVH. Both methods can be considered to run in real-time on a Jetson TX2, which can be mounted on a UAV to do onboard tracking. The average speeds of the proposed methods that run on Jetson TX2 modules are given in the last column of Table 2.

The success plots of the tested methods for the 12 scenarios are given in Fig. 4. UAVH outperforms all other methods in 9 out of 12 scenarios. It especially provides a high success score under aspect ratio and scale variations because of the adaptability YOLO provides. The performance of the proposed method seems to have suffered in background clutter, similar object and low resolution scenarios, likely because the object is searched for by applying YOLO in a region of interest around the previous loca-

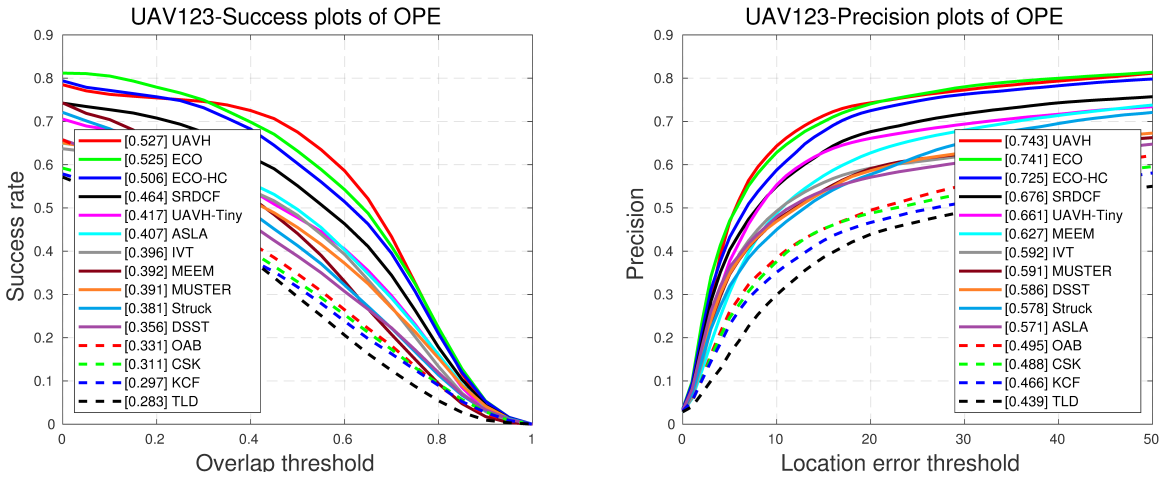

Figure 3. Success and precision scores of the methods on the UAV123 Dataset using one pass evaluation (OPE) protocol.

Table 2. The comparison of the methods on the UAV123 and UAV20L Dataset. Red, green and blue indicate the best results in descending order.

	UAV123		UAV20L		FPS	
Methods	**Success**	**Precision**	**Success**	**Precision**	**PC**	**Jetson TX2**
UAVH (ours)	**0.527**	**0.743**	**0.557**	**0.784**	**62.86**	**25**
UAVH-Tiny (ours)	0.417	0.661	**0.445**	**0.761**	**71.9**	**34**
ECO [6]	**0.525**	**0.741**	**0.435**	**0.604**	8	-
ECO-HC [6]	**0.506**	**0.725**	-	-	60	-
SRDCF [8]	0.464	0.676	0.343	0.507	5.37	-
MUSTER [17]	0.391	0.591	0.329	0.514	0.975	-
DSST [7]	0.356	0.586	0.270	0.459	25.4	-
OAB [14]	0.331	0.495	0.317	0.490	0.459	-
Struck [15]	0.381	0.578	0.288	0.437	17	-
KCF [16]	0.297	0.466	0.223	0.339	**125**	-
TLD [18]	0.283	0.436	0.197	0.336	13.8	-

tion. Object detection is particularly difficult in these cases, which has degraded our performance. This effect is most emphasized in the similar object scenario, where ECO has outperformed the proposed tracker by nearly 5%. This is expected since YOLO cannot discriminate between the instances of the same object category. Compared to UAVH, UAVH-Tiny seems to show more variance in performance in different scenarios, indicating a comparative lack of robustness.

3.3. UAV20L Dataset

UAV20L is a subset of UAV123 designed to test for long-term aerial tracking [28]. It consists of 20 video sequences and more than 58K frames. The success and precision plots of the tested methods can be seen in Fig. 5 . Similar to the experiments on UAV123, UAVH delivers the best performance. While UAVH-Tiny is close to UAVH in precision, there is a significant difference in success.

The scores for different performance metrics are given in Table 2. This time, UAVH-Tiny is among best performing top-3 methods, indicating that a shallower detector network can be used for long-term tracking. This is reasonable, as with shorter video sequences, the success of the YOLO model initializing the object location becomes more critical. On the other hand, long-term tracking depends on the KCF more, which closes the gap between UAVH and UAVH-Tiny.

3.4. Visual Comparison

Finally, we present a visual comparison of the proposed methods along with others from the literature on the UAV Tracking Dataset. See Figure 6 for examples. It can be seen that the proposed methods both estimate the general region that the UAV is in, and also returns a better bounding box with an appropriate scale and aspect ratio.

4. Conclusion

In this study, YOLOv3 and KCF are used to design a hybrid visual tracker to detect and track UAVs. The main idea is to combine a tracker and a detector to take advantages of

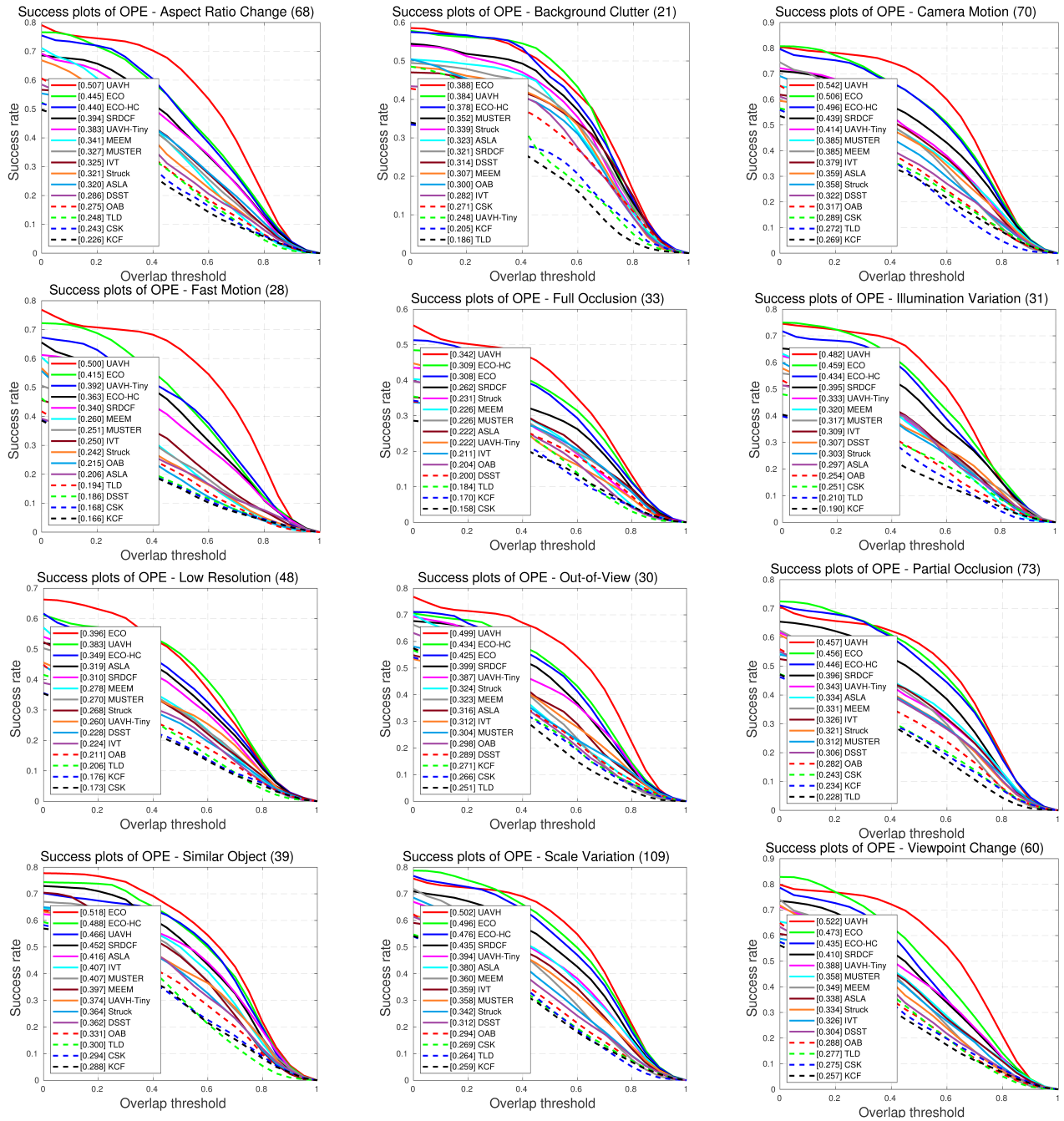

Figure 4. Success scores of the methods on the different scenarios in the UAV123 Dataset using one pass evaluation (OPE) protocol.

these two methods and reduce the limitations of the resulting tracking system. To compare the proposed method with the trackers in the literature that provide high performance in real-time, a new dataset containing UAVs has been collected. The proposed hybrid tracker (UAVH) outperformed both YOLOv3, KCF, and the other methods that it has been compared to.

To improve the speed of the proposed method, it has also been implemented with the YOLOv3-Tiny model. The resulting UAVH-Tiny method has achieved a good trade-off between accuracy and speed, and can provide real-time performance on a single board computer mounted within a UAV.

Acknowledgments. This work was funded by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant number EEEAG-116E080. We also thank NVIDIA for the donation of Quadro P5000 GPUs used in this study.

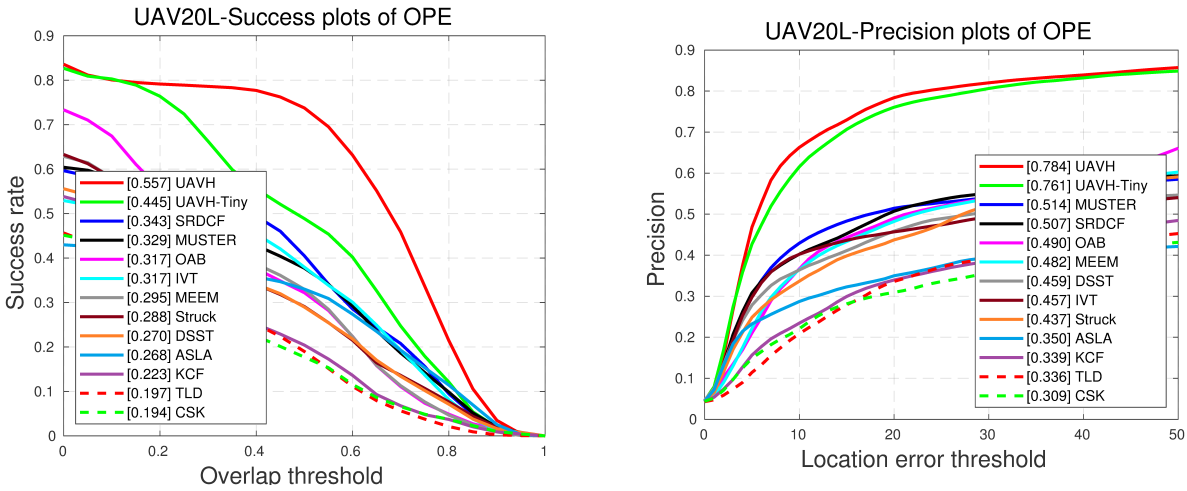

Figure 5. Success and precision scores of the methods on the UAV20L Dataset using one pass evaluation (OPE) protocol.

Figure 6. A visual comparison of the methods on the UAV Tracking Dataset. Bounding box colors are as follow: Black–ground truth, red–UAVH (ours), blue–UAVH-Tiny (ours), green–CSRT, yellow–KCF.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006. 2

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 983–990, 2009. 4

[3] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 5

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010. 2

[5] H. Cevikalp and B. Triggs. Visual object detection using cascades of binary and one-class classifiers. *International Journal of Computer Vision*, 123(3):334–349, 2017. 2, 3

[6] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6638–6646, 2017. 6

[7] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, 2017. 6

[8] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proc. IEEE International Conference on Computer Vision*, pages 4310–4318, 2015. 6

[9] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Proc. European Conference on Computer Vision*, pages 472–488, 2016. 2

[10] P. J. Davis. *Circulant Matrices*. American Mathematical Soc., 2012. 3

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proc. International Conference on Machine Learning*, pages 647–655, 2014. 2

[12] H. K. Galoogahi, A. Fagg, and S. Lucey. Learning background-aware correlation filters for visual tracking. In *Proc. IEEE International Conference on Computer Vision*, volume 3, page 4, 2017. 5

[13] R. Girshick. Fast R-CNN. In *Proc. IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 3

[14] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. British Machine Vision Conference*, volume 1, page 6, 2006. 6

[15] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016. 1, 6

[16] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 2, 3, 4, 5, 6

[17] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. MUlti-store Tracker (MUSTer): A cognitive psychology inspired approach to object tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 749–758, 2015. 6

[18] Z. Kalal, K. Mikolajczyk, J. Matas, et al. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409, 2012. 1, 4, 5, 6

[19] H. Kiani Galoogahi, A. Fagg, and S. Lucey. Learning background-aware correlation filters for visual tracking. In *Proc. IEEE International Conference on Computer Vision*, pages 1135–1143, 2017. 4

[20] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016. 4

[21] F. Li, Y. Yao, D. Z. P. Li, W. Zuo, and M.-H. Yang. Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *ICCV Workshops*, 2017. 3

[22] H. Li, Y. Li, and F. Porikli. DeepTrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, 25(4):1834–1848, 2016. 2

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. European Conference on Computer Vision*, pages 740–755, 2014. 4

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proc. European Conference on Computer Vision*, pages 21–37, 2016. 2

[25] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017. 4, 5

[26] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proc. IEEE International Conference on Computer Vision*, pages 3074–3082, 2015. 2

[27] X. Mei and H. Ling. Robust visual tracking using l1 minimization. In *Proc. IEEE International Conference on Computer Vision*, pages 1436–1443, 2009. 1

[28] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for UAV tracking. In *Proc. European Conference on Computer Vision*, pages 445–461, 2016. 4, 5, 6

[29] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016. 2

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 2, 3

[31] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 3, 4, 5

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Information Processing Systems*, pages 91–99, 2015. 2, 3

[33] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008. 1

[34] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 4

[35] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *Proc. European Conference on Computer Vision*, pages 470–484, 2012. 1