On Detecting and Handling target occlusions in Correlation-filter-based 2D tracking

Iason Karakostas, Vasileios Mygdalis, Anastasios Tefas and Ioannis Pitas
Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece
E-mail: {iasonekv, mygdalisv, tefas, pitas}@csd.auth.gr

***Abstract*—This paper focuses on the application of 2D visual object tracking in Unmanned Aerial Vehicles (UAV) for the coverage of live outdoor events, by filming moving targets (e.g., athletes, boats, cars etc.). In this application scenario, a 2D target tracker visually assists the UAV pilot (or cameraman) to maintain proper target framing, or it is employed for autonomous UAV operation. It should be expected that in such scenarios, the 2D tracker may fail due to target occlusions, illumination variations, fast 3D target motion, etc., thus, the 2D tracker should be able to recover from such situations. The proposed long-term 2D tracking algorithm solves exactly this problem, by detecting occlusions from the 2D tracker responses. Moreover, according to the immensity of the occlusion, the tracker may stop updating the tracker model or try to re-detect the target in a broader frame region. Experimental results indicate that our proposed tracking algorithm outperforms state-of-the art correlation filter trackers in UAV orientated visual tracking benchmarks, as well as in realistic UAV cinematography applications.**

***Index Terms*—2D visual object tracking, Occlusion-detection, Fast motion change.**

I. Introduction

Aerial live outdoor event coverage that relies on camera equipped unmanned aerial vehicles (UAVs), or drones, has revolutionized media production during the past years. Commercial UAVs employ 2D tracking algorithms to assist the media production crew to maintain proper target framing or even enable autonomous flight. During autonomous UAV flights that heavily rely on the 2D visual tracker (especially when the target is not equipped with a GPS or other localization aid device) it is of high importance that the 2D target tracking module is robust and even when it fails, it should be able to inform the UAV pilot/cameraman or take actions to recover. Most of the 2D visual target tracking algorithms assume that the target re-appears in successive frames, perhaps slightly translated, rotated or scaled. In realistic application scenarios, the framed target should be expected to disappear from the camera frame due to rapid target/camera motion, or may be occluded by obstacles. In fact, target occlusions have been identified as the most common cause of tracking failure [1].

State-of-the-art 2D tracking methods employ the so-called tracking-by-detection approach, i.e., they learn a discriminating function able to detect the target within a search area/window, i.e., a sub-frame region around the target detection ROI at the previous video frame. Tracking algorithms mostly differ in the optimization procedure followed to train and update the object detection model, and the employed target template representations. Although, 2D tracking algorithms that employ deep architectures and can perform real-time exist [2], [3], these methods struggle to perform real-time in UAV embedded systems. Perhaps the most successful family of real-time 2D tracking algorithms suitable for UAV applications are the correlation filter-based trackers [4]–[6].

Related work in 2D tracking that focuses on handling target occlusions have been proposed [7]–[12]. In order to detect target occlusions, heuristic methods are commonly employed, e.g., empirically setting thresholds between successive frame tracking metrics or metrics based on statistical features of the tracker response distribution, such as the maximum tracker response value [9], [10] or the Peak-to Sidelobe Ratio (PSR) [7]. During occlusions, an additional target detector may be employed, which is trained separately from the target tracker [9], [10], or the tracker itself may be applied in video frame regions selected by an object proposal algorithm [7]. However, most of these tracking methods have not been specifically designed for UAV implementation and computational complexity has not always been considered as a limitation during the algorithm design process, resulting in methods that fail to perform real-time 2D visual tracking on a UAV system.

This paper presents a 2D tracking method that detects and handles target occlusions (DHO), incorporating three components: a) a baseline 2D visual target tracker, b) a lightweight occlusion detector and c) an occlusion handling algorithm. In order to detect occlusions, an offline detector based on Support Vector Machines [13], [14] is trained from the distributions of the 2D tracker responses when no, full or partial target occlusions occur, obtained by annotated videos [15]. The tracker responses are employed as a measure of tracking quality and handled by the occlusion handling algorithm which decides whether to stop 2D tracking model training or to trigger target re-detection in a broader search region, using the tracker model as a detector. This prevents the tracker model to be updated/polluted during occlusions. Therefore, instead of training an additional object detector [9], [10] to be employed for object re-detection, the proposed tracker employs the tracker model itself for object detection [7] in a sub-frame search region, in order to maintain real-time tracking performance.

This work has received funding from the European Union' Seventh Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

II. Correlation filter-based tracking

Let $\boldsymbol{X} \in \mathbb{R}^{N \times N}$ be a data matrix where each row of $\boldsymbol{X}$ contains the vectorial representation of a target ROI translation $\boldsymbol{x}_i$. Correlation-filter based methods focus on learning a discriminating filter $\boldsymbol{w}$, that regresses the data matrix $\boldsymbol{X}$ to a Gaussian distribution $\boldsymbol{y} \in \mathbb{R}^N$, corresponding to all available ROI translations, where the original target template (not translated) is mapped to the peak y_i value. The maximum response value index of this filter with all possible ROI translations indicates the new ROI position of the target in the next frame. This discrimination filter is optimized by solving a Ridge Regression problem:

$$\min_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 + \lambda \|\boldsymbol{w}\|^2, \tag{1}$$

where λ is a regularization parameter. This problem has the following closed-form solution:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{2}$$

where $\boldsymbol{I}$ is an identity matrix of appropriate dimensions. Since $\boldsymbol{X}$ contains all the available translations of the first sample $\boldsymbol{x}_1$, it is circulant [16], having the following property:

$$\boldsymbol{X} = \boldsymbol{F}^H diag(\boldsymbol{F}\boldsymbol{x}_1)\boldsymbol{F}, \tag{3}$$

where $\boldsymbol{F}$ is the so-called Discrete Fourier Transform (DFT) matrix and $\boldsymbol{F}^H$ is its Hermitian transpose. By replacing (3) in (2), $\boldsymbol{w}$ can be obtained in the Fourier domain:

$$\hat{\boldsymbol{w}}^* = \frac{\hat{\boldsymbol{x}}^* \odot \hat{\boldsymbol{y}}}{\hat{\boldsymbol{x}}^* \odot \hat{\boldsymbol{x}} + \lambda}, \tag{4}$$

where $\odot$ denotes element-wise operations, $\hat{\cdot}$ and $\hat{\cdot}^*$ denote the DFT transform and its complex-conjugate, respectively. Finally, the response map $\boldsymbol{R}$ in the spatial domain for a test image ROI feature matrix $\boldsymbol{Z}$ at frame i, can be obtained by:

$$\boldsymbol{R} = \mathcal{F}^{-1}\left(\hat{\boldsymbol{w}} \odot \hat{\boldsymbol{z}}\right), \tag{5}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier Transform. The target center is located at the position corresponding to the maximal filter response. When employing multiple descriptor channels, the obtained response maps can be averaged, or linearly combined using a descriptor reliability metric [17].

After successful tracking, the test feature image ROI and all its translations $\boldsymbol{Z}$ are employed to update the correlation filter, in the following manner:

$$\hat{\boldsymbol{w}}_i = \frac{A_i}{B_i}, \tag{6}$$

where:

$$\begin{aligned} A_i &= (1-h)A_{i-1} + h\left(\hat{\boldsymbol{z}}^* \odot \hat{\boldsymbol{y}}\right) \\ B_i &= (1-h)B_{i-1} + h\left(\hat{\boldsymbol{z}}^* \odot \hat{\boldsymbol{x}} + \lambda\right), \end{aligned}$$

where $0 < h \leq 1$ is the so-called learning rate, which adapts the filter to increase its peak response at frame i.

III. Detecting and handling target occlusions

A. Occlusion detection

Let us assume a 2D tracking dataset containing ROI annotations of a target depicted in F video frames. Also let us assume that along with the target ROI annotation, there is some annotation $o_i = \{-1, 1\}, i = 1, \ldots, F$ that marks the frame occlusions or when the target is out-of-view e.g., partially or fully or even does not appear in the frame due to rapid camera motion ($o_i = 1$). In fact, 2D tracking benchmarks (e.g., VOT 2017 dataset [1], [15]), provide such information.

Given the $2D$ response $\boldsymbol{R}_i$ of the tracker, we extracted a patch around the center of size 11×11 pixels, in order to construct a feature vector $\boldsymbol{r}_i$ per frame, as depicted in Figure 1. The extracted $2D$ matrix was vectorized, in order to construct a $121-$dimensional feature vector and is labeled with 1 if it refers to an occluded frame and -1 otherwise, according to the dataset occlusion ground truth labels. This information is used to train a classifier able to identify target occlusions, from the tracker responses. To this end, a baseline tracker is employed in order to obtain its responses on the annotated ground truth frame ROIs. In fact, target scaling may result in ROIs of different dimensionality for each frame in the same video, e.g., $\boldsymbol{r}_i \in \mathcal{R}^{\tilde{N}}$, but they can be aligned offline by cropping or zero padding.

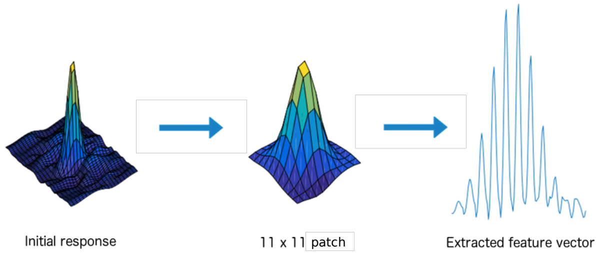


Fig. 1: Feature vector extraction for the Occlusion detection module of the proposed algorithm.

The differences in tracker response distributions, under or without target occlusion, can be learned using the standard two-class Support Vector Machines (SVM) classifier [13], [14], due to its lightweight model computational and memory requirements. In its dual form, SVM learns a hyperplane $\boldsymbol{a} \in \mathbb{R}^F$ in a feature space associated with a kernel function $\kappa(\cdot,\cdot)$, e.g., the RBF kernel function. Representations of the tracker responses in $\mathcal{F}$ are obtained by a function that maps the tracker responses to the feature space $\mathcal{H}$ ($\phi(\cdot) : \mathbb{R}^F \mapsto \mathcal{H}$). Then, the optimization problem for learning $\boldsymbol{a}$ is of the following form:

$$\begin{aligned} \min_{\boldsymbol{a},\boldsymbol{\xi},\rho} \quad & \frac{1}{2}\|\boldsymbol{a}\|^2 + c\sum_{i=1}^{F} \xi_i - \rho, \\ \text{s.t.} \quad & y_i\left(\boldsymbol{a}^T\phi(\boldsymbol{r}_i) - \rho\right) \leq 1 - \xi_i, i = 1, \ldots, F, \\ & \xi_i \geq 0, \end{aligned} \tag{7}$$

where $\phi(\boldsymbol{r}_i)$ are the representations of the tracker responses in $\mathcal{F}$, ξ_i are the slack variables, $c > 0$ is the parameter controlling the amount of allowed error and ρ is the SVM bias term, representing the offset of the SVM hyperplane from the origin. After obtaining $\boldsymbol{a}$, the response of the classifier for a given tracker response $\boldsymbol{r}$, is of the following form:

$$o = \sum_{i=1}^{F} y_i e_i \kappa(\boldsymbol{r}_i, \boldsymbol{r}) - \rho + \beta, \qquad (8)$$

where e_i are the Lagrange Multipliers of the support vectors. Occlusions are detected if $o \geq 0$. An additional bias term β has been introduced to the standard SVM bias term, that can be manually tuned, in order to adjust the precision of the trained model, allowing possibly minor partial occlusions and misclassified cases to be classified as non-occlusions.

B. Handling detected occlusions

Tracker responses are evaluated by the occlusion detection module and are handled by the validation and re-detection functions.

The validation function reads the tracker response $\boldsymbol{R}$ and employs the occlusion detection module in order to validate the obtained response using equation (8). If no occlusion is detected, i.e., $o_i < 0$, the tracker model is updated. As a result, for video sequences where no occlusion have been detected, the proposed tracking algorithm degenerates to a standard Correlation-filter based 2D tracker. The Staple [18] tracker is employed to this end. For the cases where an occlusion has been detected (i.e., $o_i > 0$), the re-detection function is triggered.

The re-detection function checks whether the target appears in the neighborhood area of the last tracked object ROI. The re-detection search area is broken into 8 nearest overlapping window size areas, from top left to bottom right of the standard window size area. The trained filter $\boldsymbol{w}$ is employed to obtain all neighborhood area responses. Then, the 9 responses are compared (including the last detected object ROI) and the target is allocated to the position where the tracking response is maximum. Unlike related methods, the tracking model is not trained or re-initialized during the re-detection process. As shown in Figure 2 the proposed framework detects an occlusion, searches for the target at the re-detection area and manages to re-detect the target using the same baseline tracking model $\boldsymbol{w}$ and the overall algorithm is presented in Algorithm 1.

The introduced properties of the proposed tracking algorithm allow the tracking model to stop being updated when an occlusion has been detected, without polluting the trained target model and hence to allow target re-detection, once the object re-appears in the neighborhood area. This is valuable in UAV cinematography applications, since manual tracking model re-initialization (due to model pollution) is no longer required, as it can be potentially costly and frustrating for the drone pilot/cameraman. Moreover, by considering that the tracking model does not require re-initialization, it can potentially be more accurate than a re-initialized model that was trained only in the few previous video frames.

Algorithm 1 Proposed algorithm

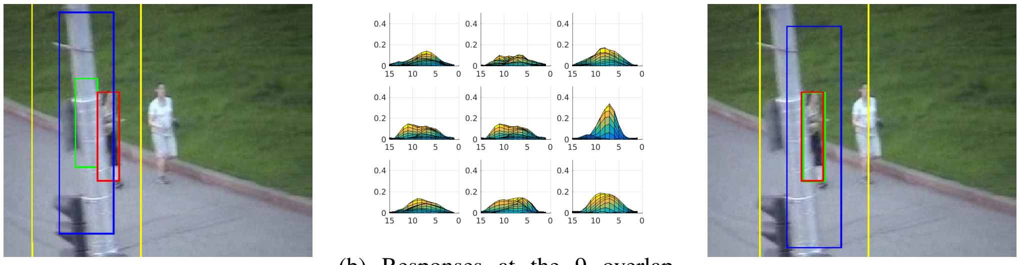
```
procedure MAIN                                              ▷ Main loop
    $f \leftarrow obtainNextFrame()$
    $[\boldsymbol{w}, \boldsymbol{X}, pos] \leftarrow initModel(p, f, pos)$
    $f \leftarrow obtainNextFrame()$
    while $(f\ != NULL)$ do
        $\boldsymbol{Z} \leftarrow trackingWindow(f, pos)$
        $[\boldsymbol{r}, pos] \leftarrow track(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z})$
        $o \leftarrow validation(\boldsymbol{r})$
        if $o < 0$ then
            $[\boldsymbol{w}, \boldsymbol{X}] \leftarrow updateModel(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z})$
        else
            $pos \leftarrow Redetection(f, \boldsymbol{w}, \boldsymbol{X})$
        $f = obtainNextFrame()$
                                                            ▷ End while loop
    return
procedure REDETECTION($f, \boldsymbol{w}, \boldsymbol{X}$)
    for $k = 1 : 8$ do
        $\boldsymbol{Z} \leftarrow trackingWindow(f, area(k))$
        $[\boldsymbol{r}_k, tempPos(k)] \leftarrow track(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z}_k)$
    $pos \leftarrow tempPos(\text{argmax}(\boldsymbol{r}_k))$
    return $pos$
```

IV. Experimental Results

For our evaluation, we utilized the videos from a publicly available video tracking benchmark, namely **UAV123** [19] which is a UAV specific video tracking benchmark dataset. It contains 123 fully annotated sequences captured using UAVs in various scenes with challenging attributes for visual 2D target tracking such as occlusions, fast target and camera motion, background clutter etc. In addition to this dataset we utilized the **BikeUAV** dataset which contains footage from famous bike races. This dataset consists of 21 sequences that most of them have been used for live broadcasting. The targets in BikeUAV are usually partially or full occluded, have fast motion or other challenging attributes for the 2D visual tracker.

The performance of the proposed DHO tracker is compared against state-of-the art Correlation filter based tracking algorithms such as Staple [18], BACF [20] and SRDCF [17]. In addition we compare the performance of DHO against algorithms that employ occlusion detection/long-term tracking mechanisms, i.e., ROT [8] and LCT [9] and against framework enhanced tracking algorithms e.g. Staple-CA [21].

In all the benchmark datasets, we exploit the one-pass evaluation (OPE) method. In this approach, the tracking algorithm is initialized in the first frame with the ground truth position and size of the object and tries to track the object for the rest of the video frames. This evaluation type is more appropriate for evaluating methods in terms of long-term tracking, which is more related to the proposed 2D tracking algorithm application domain. Our evaluation is based on two widely used metrics, Center Location Error (CLE) and Overlap Score (OS). The first one is computed by measuring the Euclidean distance in pixels, between the center locations of the tracker output



(a) Initial tracker response at frame #79. Occlusion is detected.

(b) Responses at the 9 overlapping window sizes inside the re-detection area.

(c) Output of the DHO tracker. The target is re-detected.

Fig. 2: Re-detection process during a sequence. Tracker outputs (green), window size area (blue), re-detection area (yellow), target position (red). The proposed DHO tracker detects an occlusion at frame #79, searches for the target at the re-detection area and manages to re-detect the target.

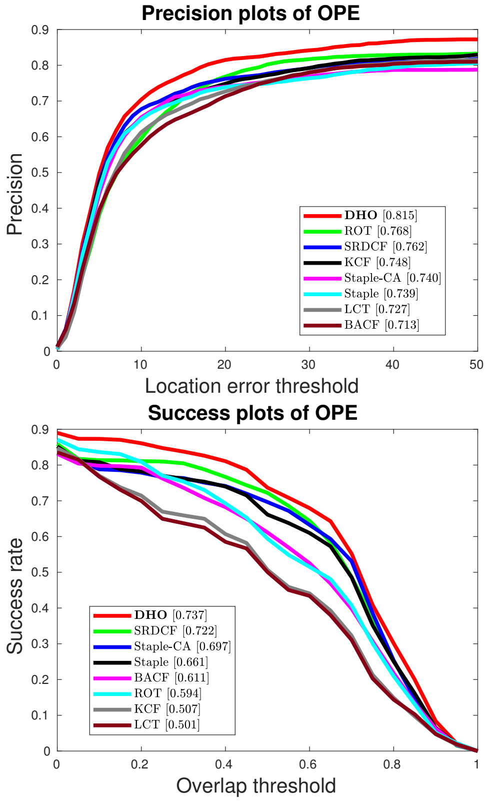


Fig. 3: Precision and success plots for the real case scenario dataset BikeUAV.

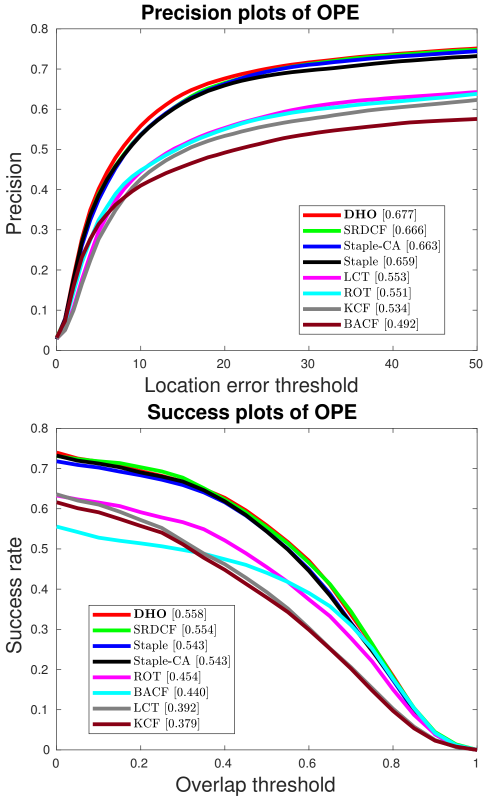


Fig. 4: Precision and success plots for the challenging benchmark UAV123.

ROIs and the ground truth ones. Then the CLE measure is computed by calculating the percentage of the frame where this distance is less than a certain threshold, in our case 20 pixels. This metric, although cannot evaluate the performance of the tracker in terms of target scale variations, is useful for UAV tracking applications since independent of correct scale estimation, if the center position of the tracker is close to the ground truth one, the UAV will manage to follow the desired object.

We also use the Overlap Score (OS) defined as $S = \frac{|r_t \cap r_0|}{|r_t \cup r_0|}$,

TABLE I: Average precision and frames per second for the evaluation datasets. Bold font indicates the proposed tracking algorithm and the top performance is annotated with red color.

Tracker	Avg. Precision	Avg. FPS
DHO	**0.746**	**60.1**
SRDCF	0.714	12.9
Staple-CA	0.702	33.8
Staple	0.699	80.3
ROT	0.660	52.7
KCF	0.641	375
LCT	0.640	28.0
BACF	0.603	25.9

where r_t and r_0 is the tracked and ground truth bounding boxes respectively, $\cap$ and $\cup$ denote the intersection and union operators and $|\cdot|$ denotes the number of pixels inside the specified area. OS is calculated in a per frame basis. When the value of S is larger than a certain threshold (0.5 in our case), it is assumed that the tracker, successfully tracks the desired object. In contrast with the CLE, this evaluation metric is affected when the 2D tracking algorithm fails to adjust to target ROI scale and size variations.

In Table I, the average results in terms of presicion and frames per second (FPS) are presented. The proposed algorithm outperforms SRDCF in average precision by more than 3% while managing to be almost 6 times faster in terms of FPS, on a Linux PC equipped with an Intel i7 processor. By examining the FPS results, only KCF, which achieves a remarkable top performance of 375 fps and Staple have better performance than the proposed tracker. Although, the proposed tracker manages a 10% gap in terms of precision when compared to KCF and almost 5% against Staple.

Figures 3 and 4 depict the precision and success plots in BikeUAV and UAV123 datasets, respectively. By examining the results, it should be noted that the proposed DHO tracker manages to outperform much more demanding trackers in terms of computational burden, such as SRDCF or BACF. In BikeUAV, DHO tracker manages to achieve the top performance in precision (0.815) and success rate (0.737) outperforming the competition by 5% in terms of precision. In UAV123, DHO tracker has the best performance overall both in precision and success rate metrics. From the rest of the competing tracking algorithms, only SRDCF has a performance close to the proposed DHO tracking method, however it is much slower.

V. Conclusion

A 2D tracking algorithm, namely DHO, that detects and handles target occlusions in 2D tracking has been presented. DHO tracker decreases drifts that may occur due to target occlusions, thus the tracker performance is enhanced. In addition, our algorithm prevents the tracker model to be updated during occlusions resulting in a more robust model that can be exploited for object re-detection when it reappears. Future work could include adapting the proposed tracker to deep-learning based trackers, since embedded systems in the future will probably posses improved computational power.

References

[1] Matej Kristan, Aleš Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin Zajc, Tomas Vojir, Gustav Häger, Alan Lukežič, Abdelrahman Eldesokey, and Gustavo Fernandez, "The visual object tracking vot2017 challenge results," 2017.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking," *European conference on computer vision (ECCV)*, pp. 850–865, 2016.

[3] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[4] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui, "Visual object tracking using adaptive correlation filters," *Computer Vision and Pattern Recognition (CVPR)*, pp. 2544–2550, 2010.

[5] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[6] Erhan Gundogdu, Huseyin Ozkan, and A Aydın Alatan, "Extending correlation filter-based visual tracking by tree-structured ensemble and spatial windowing," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5270–5283, 2017.

[7] Rui Li, Minjian Pang, Cong Zhao, Guyue Zhou, and Lu Fang, "Monocular long-term target following on uavs," *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29–37, 2016.

[8] Xingping Dong, Jianbing Shen, Dajiang Yu, Wenguan Wang, Jianhong Liu, and Hua Huang, "Occlusion-aware real-time object tracking," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 763–771, 2017.

[9] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang, "Long-term correlation tracking," *Computer Vision and Pattern Recognition (CVPR)*, pp. 5388–5396, 2015.

[10] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang, "Adaptive correlation filters with long-term and short-term memory for object tracking," *CoRR*, vol. abs/1707.02309, 2017.

[11] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[12] Tianzhu Zhang, Bernard Ghanem, Changsheng Xu, and Narendra Ahuja, "Object tracking by occlusion detection via structured sparse learning," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1033–1040, 2013.

[13] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola, "A generalized representer theorem," *International Conference on Computational Learning Theory*, pp. 416–426, 2001.

[14] Alex J Smola and Bernhard Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[15] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, Nov 2016.

[16] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," *European conference on computer vision (ECCV)*, pp. 702–715, 2012.

[17] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg, "Learning spatially regularized correlation filters for visual tracking," *International Conference on Computer Vision (ICCV)*, pp. 4310–4318, 2015.

[18] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr, "Staple: Complementary learners for real-time tracking," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1401–1409, 2016.

[19] Matthias Mueller, Neil Smith, and Bernard Ghanem, "A benchmark and simulator for uav tracking," *European Conference on Computer Vision (ECCV)*, pp. 445–461, 2016.

[20] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey, "Learning background-aware correlation filters for visual tracking," *Computer Vision and Pattern Recognition (CVPR)*, pp. 21–26, 2017.

[21] Matthias Mueller, Neil Smith, and Bernard Ghanem, "Context-aware correlation filter tracking," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1396–1404, 2017.