Occlusion detection and drift-avoidance framework for 2D visual object tracking

Iason Karakostas[a,*], Vasileios Mygdalis[a], Anastasios Tefas[a] and Ioannis Pitas[a]

[a]*Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece*

Author pre-print version, submitted to Elsevier Signal Processing: Image Communication.
Please cite the publisher maintained version in your work.
DOI : https://doi.org/10.1016/j.image.2020.116011

ARTICLE INFO

Keywords:
2D visual object tracking
Occlusion-detection
Fast motion change
ODDA framework.

ABSTRACT

This paper presents a long-term 2D tracking framework for the coverage of live outdoor (e.g., sports) events that is suitable for embedded system application (e.g. Unmanned Aerial Vehicles). This application scenario requires 2D target (e.g., athlete, ball, bicycle, boat) tracking for visually assisting the UAV pilot (or cameraman) to maintain proper target framing, or even for actual 3D target following/localization when the drone flies autonomously. In these cases, it should be expected that the target to be tracked/followed, may disappear from the UAV camera field of view, due to fast 3D target motion, illumination changes, or due to visual target occlusions by obstacles, even if the actual UAV continues following it (either autonomously, by exploiting alternative target localization sensors, or by pilot maneuvering). Therefore, the 2D tracker should be able to recover from such situations. The proposed framework solves exactly this problem. Target occlusions are detected from the 2D tracker responses. Depending on the occlusion immensity, the proposed framework decides whether to not update the tracking model, or to employ target re-detection in a broader window. As a result, the proposed framework allows continued target tracking once the target re-appears in the video stream, without tracker re-initialization.

1. Introduction

2D visual object tracking is one of the most widely studied computer vision problems, having applications in a wide range of domains, including robotics, media production, industrial semantic annotation, visual surveillance, human-computer interfaces and augmented reality. It involves localizing and tracking an object Region of Interest (ROI) in a video input. Tracking can be initialized by object detection or by manual object localization, and continues for a number of successive video frames in the stream, thus maintaining the identity label of the tracked object. Most tracking algorithms assume that the tracked ROI always re-appears in the successive frames, perhaps slightly translated, rotated or scaled. However, in realistic application scenarios, it should also be expected that at times the target may be partially or fully visually occluded for some video frames. According to Visual Object Tracking Challange 2017 (VOT2017) report [21], occlusions are the most common causes of tracking failure. Depending on the application, different tracking specifications are required regarding tracking speed and robustness to target ROI transformations and occlusions. This paper focuses on the application of 2D visual object tracking in Unmanned Aerial Vehicles (UAV) for the coverage of live outdoor (e.g., sports) events, by filming moving targets (e.g., athletes, boats, cars etc.).

In UAV cinematography applications, the main purpose of employing a 2D visual object tracking algorithm is for visually assisting the control operations of the drone pilot (or cameraman), in order to maintain proper target framing according to cinematographic rules (e.g., the center of the video frame). In the absence of 3D target localization sensors e.g., target GPS, 2D tracking may be also be employed for target following (e.g. by visual survoing) or for 3D target localization (e.g. by stereo). The employed 2D tracking algorithm must operate in a timely manner, fast enough in order to provide the necessary output to a control module, that estimates the required change of speed/direction of the UAV itself or its camera gimbal. Furthermore, since the mo-

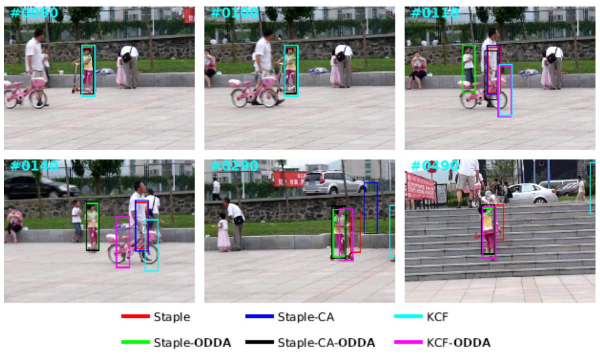


Figure 1: Traditional 2D tracking methods continue training their model during target occlusions, causing unrecoverable tracking failure. The proposed framework, implemented on top of traditional tracking methods (noted by ODDA), manages to detect the occlusion and enables tracking continuation at target reappearance.

*Corresponding author
iasonekv@csd.auth.gr (I. Karakostas); mygdalisv@csd.auth.gr (V. Mygdalis); tefas@csd.auth.gr (A. Tefas); pitas@csd.auth.gr (I. Pitas)
ORCID(s):

tion of both the target and the UAV are unconstrained, the tracker should be robust to 2D target rotations, i.e., learn the 2D target from varying viewing angles. Moreover, it should be expected that the tracked target ROI may disappear from the UAV camera field of view, or may be occluded by obstacles, while the UAV (either autonomously, or by pilot control) continues following the target. Therefore, we argue that a potentially successful UAV tracking application should handle target occlusions in a precise manner.

In this paper, a generic 2D tracking framework that detects and handles target occlusions is presented, consisting of: a) a baseline 2D visual target tracker, b) a lightweight occlusion detector and c) an occlusion handling algorithm. We employ baseline correlation filter 2D trackers, due to computational capacity limitations on embedded/mobile platforms, such as the ones used for on-drone computing. The occlusion detector is built using a-priory information about the distributions of the 2D tracker responses when no, full or partial target occlusions occur. It is trained offline (using Support Vector Machines [38, 41]), on previously annotated videos. Its responses are employed during 2D tracking as a measure of tracking quality and are managed by the occlusion handling algorithm, that decides whether to train the tracking model or attempt to re-detect the target in a bigger search region, by employing the tracker as object detector. As a result, the tracker model is not updated/polluted during occlusions, and could still be employed to detect the target once the target re-appears in the video stream. Our experimental results denote that our proposed framework increases the tracking precision of baseline trackers in well known 2D tracking benchmarks, as well as in realistic UAV cinematography applications, with a only a small fraction in 2D tracking runtime.

Our contributions can be summarized as follows:

- The proposed framework is generic, easy to implement, and can be employed on top of any correlation filter-based tracker, or other tracking frameworks, e.g., Context-Aware tracking [33]. Extensions of the proposed framework will allow it to work on top of deep-learning based trackers, as well.
- The proposed solution targets on potential UAV implementation, thus it was evaluated/implemented with consideration to computational complexity burdens.
- Our proposed framework detects occlusions from the whole tracker responses maps, not only specific points/features of the tracker response distribution (e.g., max, mean, etc.). The tracker stops being trained when occlusions occur.
- Instead of training an additional object detector [31, 30] to be employed for object re-detection, the proposed framework employs the tracker itself for object detection [26], in a sub-frame search region, in order to maintain real-time tracking performance. Hence, no tracker re-initialization is needed.

The remainder of the paper is structured as follows. An extensive review of available state-of-the art trackers is provided in Section 2. The proposed framework that handles occlusions is described in Section 3. Experimental results are given in Section 4. Finally, conclusions are drawn in Section 5.

2. 2D tracking for UAV cinematography overview

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. A very important factor for tracking precision and computational complexity, are the employed feature representations of the object template. Such can be gray scale pixel luminance, fHOG features [18], color histograms [39] combined multi-channel feature descriptor types [7, 1, 8] or multi-resolution feature maps extracted using deep learning architectures [9, 6, 43, 16, 2, 34]. Deep learning representations are typically extracted by well-established architectures such as the VGG [9, 40], with some fine tuning on the problem by hand [42] or even combinations of pre-trained and newly introduced architectures, specified for tracking [30, 6, 43, 12, 13, 48, 44, 35, 36]. Tracking methods can also focus on a specific problem such as face tracking [5, 37, 24, 23, 19] or pre-trained for specific object tracking [25]. Up to date, advanced Deep Learning-based object template representations provide the best 2D tracking precision. However, their implementation suggests specific hardware requirements to be met especially in terms of GPU, which are not widely available in mobile/embedded hardware platforms such as UAVs, with some notable exceptions (e.g., custom made UAV platforms based on NVIDIA Xavier/TX2, etc.).

Perhaps the most prominent algorithm family for potent real-time UAV implementation, whose performance have been well-proven over the past few years, are 2D tracking algorithms based on correlation filters [3, 18, 14]. A correlation filter works by regressing the representations of all possible object template translations to a Gaussian distribution. The original (untranslated) ROI object template is regressed to its peak. Due to the circulant structure of the template representations, the regression problem can be solved fast in the Fourier domain, thus accelerating the learning and testing processes of the tracker. Correlation filter-based tracking approaches seem to be more robust to target rotations than alternatives based on classification [15]. The most advanced correlation filter-based variants employ alternative/-modified optimization problems, combining single-channel with color information [7], estimate spatial reliability of the features [8], or even complement the standard correlation filter-based optimization problem with another filter, that could be based on color statistics [1], particle filters [47] or even template matching [29] methods. In addition, other methods employ information from the exterior of the search area. For example they use patches from the background [33, 11] in the optimization problem, and force patches around the target ROI to be regressed to zero, alleviating tracking drifting to background areas. Such advanced correlation filter trackers, closely match the precision that can be obtained by deep learning methods, only at a fraction of computational complexity. Thus today they are more suitable for embedded system implementation. However, although they are more robust to occlusions than their simpler alternatives [18], none of the above mentioned methods have a innate mechanism to handle tracking failures, which is important in UAV cinematography applications, as it requires long-term and robust target tracking.

Methods that consider target occlusions during the tracker optimization process [26, 10], or long-term tracking methods [31, 30, 20, 46], have been introduced in the recent past as well. In order to detect target occlusions, they typically exploit heuristic methods, i.e., they empirically set thresholds between successive frame tracking metrics e.g., distance metrics between consecutive window patches [10], or metrics based on statistical features of the tracker response distribution, such as the maximum value [31, 30] or the Peak-to Sidelobe Ratio (PSR) [26]. When an occlusion is detected, an additional target detector may be employed, trained separately from the target tracker, such as a random fern detector [31, 30]. Alternatively, the tracker itself may be applied in video frame regions selected by an object proposal algorithm [26]. Another approach is to maintain a filter pool from the previously successful tracked image patches and employ the one producing maximum response [10, 49]. However, such tracking methods have some disadvantages. Some of them have not been specifically designed for UAV implementation and moreover, computational complexity is not always considered as limitation. Furthermore, occlusion detection based on heuristic methods is not optimal, since it does not exploit any prior information about the tracker responses.

3. Occlusion detection and drift-avoidance framework tracking framework

In this section, we detail the proposed Occlusion Detection and Drift-Avoidance (ODDA) 2D tracking framework. In Subsection 3.1, we describe the baseline correlation filter-based tracking module. In Subsection 3.2, we describe the occlusion detection module. Finally, the overall algorithm that combines the occlusion detection and tracking modules is described in Subsection 3.3.

3.1. Correlation filter-based tracking

The first component of the proposed framework is a baseline correlation filter-based tracking module. Tracking methods expect an input template image ROI containing the target to be tracked, that has been obtained by manual annotation or object detection. Let $\boldsymbol{x}$ be the vectorial representation of size $N = H \times W$, where H, W are the window height and width of a single-channel description of the initial tracking region extracted by using some descriptor type, e.g., fHOG. For each descriptor channel, the goal is to use the input template ROI representation and several negative/distorted examples in order to train a tracking model, in the form of a correlation filter $\boldsymbol{w}$. In correlation-filter based methods, the negative/distorted examples are generated implicitly using translated image templates derived by performing cyclic shifts of the elements of $\boldsymbol{x}$ [18], corresponding to all possible 2D image translations of the image template in the succeeding tracking search region. To this end, these samples are generated using a cyclic shift operator, i.e., a permutation matrix $\boldsymbol{P}$ that shifts $\boldsymbol{x}$ by one element at a time:

$$\boldsymbol{x}_i = \boldsymbol{P}^i \boldsymbol{x}, i = 0 \dots n-1, \tag{1}$$

forming a datamatrix $\boldsymbol{X} \in \mathbb{R}^{N \times N}$ that contains the input target template representations and all its cyclic shifts. The i−th row of $\boldsymbol{X}$ contains the vectorial representation of i−th pre-specified target ROI cyclic shift $\boldsymbol{x}_i$. All horizontal and vertical 2D translations are encoded by 1D row cyclic shifts on matrix $\boldsymbol{X}$ [18].

The filter $\boldsymbol{w}$ is trained in order to regress the data matrix $\boldsymbol{X}$ to a Gaussian distribution $\boldsymbol{y} \in \mathbb{R}^N$, corresponding to all possible circular ROI translations, where the original target template $\boldsymbol{x}_1$ (not translated) is mapped to the peak y_i value. In order to exploit multiple descriptor channels D (e.g., fHog or deep CNN representations), a set $\mathcal{X} = \{\boldsymbol{X}^1, \dots, \boldsymbol{X}^D\}$ can be employed, by replicating the procedure below for each descriptor channel. Filter $\boldsymbol{w}$ is optimized by solving a Ridge Regression problem [18]:

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

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

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{3}$$

where $\boldsymbol{I}$ is an identity matrix of appropriate dimensions. Since $\boldsymbol{X}$ contains all the possible circular ROI translations of the first sample $\boldsymbol{x}_1$, it is circulant [17], having the following property:

$$\boldsymbol{X} = \boldsymbol{F}^H diag(\boldsymbol{F}\boldsymbol{x}_1)\boldsymbol{F}, \tag{4}$$

where $\boldsymbol{F}$ is the so-called Discrete Fourier Transform (DFT) matrix and $\boldsymbol{F}^H$ is its Hermitian transpose. By replacing (4) in (3), $\boldsymbol{w}$ can be obtained in the Fourier domain:

$$\hat{\boldsymbol{w}}^* = \frac{\hat{\boldsymbol{x}}^* \odot \hat{\boldsymbol{y}}}{\hat{\boldsymbol{x}}^* \odot \hat{\boldsymbol{x}} + \lambda}, \tag{5}$$

where $\odot$ denotes element-wise operations, the division is element-wise, $\hat{\cdot}$ and $\hat{\cdot}^*$ denote the DFT transform and its complex-conjugate, respectively.

Let $\boldsymbol{Z}$ be a data matrix that contains the test template representations defined in a similar manner to $\boldsymbol{X}$, i.e., its first row $\boldsymbol{z}_1$ is the untranslated image representation of the tracker search region. The target to be tracked is expected to appear somewhere in this region, without being centered. The 2D translation of the target in the test image can be found by exploiting the response map $\boldsymbol{R}^{H\times W}$ in the spatial domain for a test image ROI feature matrix $\boldsymbol{Z}$ at frame i:

$$\boldsymbol{R} = \mathcal{F}^{-1}\left(\hat{\boldsymbol{w}} \odot \hat{\boldsymbol{z}}\right), \tag{6}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier Transform. An example of a response map is given in Fig. 3, labeled as initial response. The target center translation can be obtained by the offset of the maximal response of the filter compared to the expected one (centered target). When employing multiple descriptor channels, the obtained response maps can be averaged, or linearly combined using a descriptor reliability metric [8].

After successful tracking, at the new ROI position, the training feature image ROI and all its translations $\boldsymbol{X}_n$ are employed to update the correlation filter, in the following manner:

$$\begin{aligned}\hat{\boldsymbol{w}}_i &= \frac{A_i}{B_i}, \\ A_i &= (1-h)A_{i-1} + h\left(\hat{\boldsymbol{x}}_n^* \odot \hat{\boldsymbol{y}}\right) \\ B_i &= (1-h)B_{i-1} + h\left(\hat{\boldsymbol{x}}_n^* \odot \hat{\boldsymbol{x}}_n + \lambda\right),\end{aligned} \tag{7}$$

where $0 < h \leq 1$ is the so-called learning rate, which adapts the filter to increase its peak response at frame i. This baseline tracking algorithm does not introduce any metric to define successful tracking, to be discussed in the following subsection.

Besides the above described baseline tracker, any tracker that provides its response in a similar manner can be employed in our proposed tracking framework.

3.2. Learning tracker response distribution for occluded targets

The second component of the proposed framework is a classifier that estimates tracking quality, by evaluating the output tracker response maps. This classifier is trained offline using previously acquired knowledge about target occlusions, in publicly released tracking datasets. Let us assume that there is a dataset consisting of target templates of size $N = W \times H$, in K video frames. Also let us assume that along with the target ROI annotation, there is some annotation $o_i = \{-1, 1\}, i = 1, \dots, F$ 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 ($o_i = 1$). Recent 2D tracking benchmarks (e.g., VOT 2017 dataset [21, 22]), provide such information. To train the occlusion detection module, we created a dataset of tracker response maps and occlusion labels $\mathcal{R}_i = \{\boldsymbol{r}_i, o_i\}, i = 1, \dots, F$, using vectorized response maps of the baseline tracker module for each video frame $\boldsymbol{r}_i \in \mathbb{R}^N$, on window locations around the annotated ground truth ROIs. 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.

The differences in tracker response distributions, under or without target occlusion, can be learned using the standard two-class Support Vector Machines (SVM) classifier [38, 41] by employing LibSVM [4]. In its dual form, SVM learns a coefficient vector $\boldsymbol{a} \in \mathbb{R}^F$ for reconstructing the SVM hyperplane on a feature space that can be of arbitrary dimensionality and is associated with a kernel function. A mapping function is employed to map the tracker responses to the feature space $\mathcal{H}$ ($\phi(\cdot) : \mathbb{R}^F \mapsto \mathcal{H}$). The similarities in that space that must be known, in order to train the SVM classifier, are associated with the kernel function $\kappa(\boldsymbol{r}_i, \boldsymbol{r}_j) = \phi(\boldsymbol{r}_i)^T\phi(\boldsymbol{r}_j)$. To this end, we employed the RBF kernel function. The optimization problem for learning $\boldsymbol{a}$ is of the following form:

$$\begin{aligned}\min_{\boldsymbol{a},\xi,\rho} \quad & \frac{1}{2}\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a} + c\sum_{i=1}^{F}\xi_i - \rho, \\ \text{s.t.} \quad & y_i\left(\boldsymbol{a}^T\boldsymbol{k}_i - \rho\right) \leq 1 - \xi_i, i = 1, \dots, F, \\ & \xi_i \geq 0,\end{aligned} \tag{8}$$

where $\boldsymbol{K}$ is the so-called SVM kernel matrix, $\boldsymbol{k}_i$ corresponds to its i-th row, 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:

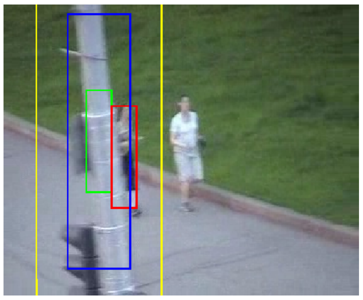
$$o = \sum_{i=1}^{F} y_i\alpha_i\kappa(\boldsymbol{r}_i, \boldsymbol{r}) - \rho + \beta. \tag{9}$$

Values of f $o \geq 0$ indicate an occlusion. The detailed procedure for training the Occlusion detection module is discussed in Subsection 4.1. In order to adjust the precision of the trained model, allowing possibly minor partial occlusions and misclassified cases to be classified as non-occlusions, we introduce an additional bias term β to the standard SVM bias term, that can be manually tuned based on the application by hand, whose effect is discussed thoroughly in Subsection 4.1.1.

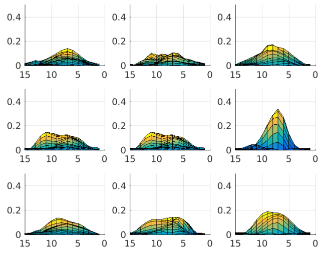
3.3. Overall tracking framework

The overall tracking framework consists of three modules: a) the baseline tracking module, b) the tracking failure detection module and c) the failure handling mechanism. The proposed occlusion detection method can be implemented on top of any correlation filter-based tracking module, as the ones described in Subsection 3.1.

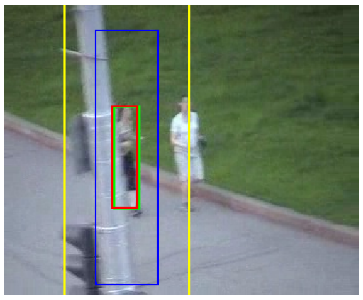
- The track function takes as input the tracking model $\boldsymbol{w}$, the input target template representations $\boldsymbol{X}$ and the representations of the test image ROI feature matrix



(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 ODDA framework. The target is re-detected.

Figure 2: Redetection process during the Jogging sequence, using KCF as baseline tracker. Tracker outputs (green), window size area (blue), re-detection area (yellow), target position (red). The proposed ODDA framework detects an occlusion at frame #79, searches for the target at the re-detection area and manages to re-detect the target using the same baseline tracking model $\boldsymbol{w}$.

Algorithm 1 The ODDA tracking framework

```
1:  procedure MAIN                                          ▷ Main loop
2:      $f \leftarrow obtainNextFrame()$
3:      $[\boldsymbol{w}, \boldsymbol{X}, pos] \leftarrow initModel(p, f, pos)$
4:      $f \leftarrow obtainNextFrame()$
5:      while $(f\ != NULL)$ do
6:          $\boldsymbol{Z} \leftarrow trackingWindow(f, pos)$
7:          $[\boldsymbol{r}, pos, \boldsymbol{X}_n] \leftarrow track(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z})$
8:          $o \leftarrow validation(\boldsymbol{r})$
9:          if $o < 0$ then
10:             $[\boldsymbol{w}, \boldsymbol{X}] \leftarrow updateModel(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{X}_n)$
11:         else
12:             $pos \leftarrow Redetection(f, \boldsymbol{w}, \boldsymbol{X})$
13:         $f = obtainNextFrame()$
                                                            ▷ End while loop
14:     return
15: procedure REDETECTION$(f, \boldsymbol{w}, \boldsymbol{X})$
16:     for $k = 1 : 8$ do
17:         $\boldsymbol{Z} \leftarrow trackingWindow(f, area(k))$
18:         $[\boldsymbol{r}_k, tempPos(k)] \leftarrow track(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z}_k)$
19:     $pos \leftarrow tempPos(\text{argmax}(\boldsymbol{r}_k))$
20:     return $pos$
```

$\boldsymbol{Z}$. Equation 6 is employed in order to calculate the response $\boldsymbol{R}$ and from $\boldsymbol{R}$ the updated ROI position. This function also returns the $\boldsymbol{X}_n$ matrix of the updated position.

- The validation function reads the tracker response $\boldsymbol{R}$ and employs the occlusion detection module in order to validate the obtained response using equation (9). If no occlusion is detected, i.e., $o_i < 0$, the tracker model is updated using (7). As a result, for video sequences where no occlusion have been detected, the proposed framework degenerates to the employed baseline tracker. For the cases where an occlusion has been detected, it employs the re-detection function ($o_i > 0$).

- The update model function, employs the tracking model $\boldsymbol{w}$, the input target template representations $\boldsymbol{X}$ and the representations of the test image ROI feature matrix at the updated position $\boldsymbol{X}_n$. The tracking model is updated using (7).

- 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 $\boldsymbol{w}$.

The overall algorithm is presented in Algorithm 1.

The introduced properties of the proposed framework 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.

3.4. Computational complexity

The computational complexity introduced by incorporating the proposed framework to a baseline tracker is related to the complexity of the introduced occlusion detection

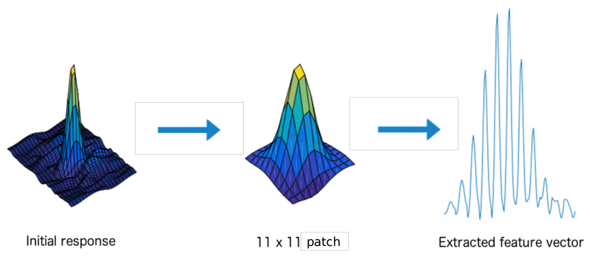


Figure 3: Feature vector extraction for the proposed Occlusion detection module.

and re-detection functions. Let O be the required computational complexity of the baseline tracker, in order to obtain its response $\boldsymbol{R}$. The proposed framework introduces two additional processes, one related to tracker response evaluation using the SVM classifier, and one related to the re-detection process. In that sense, evaluation of $\boldsymbol{R}$ with the occlusion detection module, requires estimating the kernel $\kappa(\boldsymbol{r}_i, \boldsymbol{r}), i = 1, \dots, K$ of (9). For the case where the adopted kernel functions calculates dot products, such as the RBF kernel, the complexity is linear to the number of support vectors and the descriptor dimensionality, which of $\mathcal{O}(NF)$. In addition to evaluating the tracker response, when occlusions are detected, the re-detection function is activated, obtaining 8 additional responses of complexity O in the corresponding frame. Therefore, the total complexity of the proposed framework for each frame is equal to $O + \mathcal{O}(NF)$, plus $8 \times O$ for every re-detection process. It should be noted, that the tracker model is not updated, if the re-detection process has been activated.

4. Framework evaluation and experimental results

4.1. Precision of the Occlusion Detection module

In order to obtain $\boldsymbol{r}_i$ of Eq. 9, a tracking module is employed to extract its responses at the annotated video sequences. To this end, KCF [18], LDES [27] without the rotation module thus indicated as LDES-no, Staple [1] and Staple-CA [33] were employed as baseline tracking modules. For each tracker, we utilized the video sequences of the VOT2017 dataset [21, 22] which provides occlusion labels on a per frame basis. In order to evaluate the proposed framework to other datasets as well, we only employed 29 videos that have occlusions and are unique in the VOT2017 dataset (i.e., they do not exist in datasets used for the proposed framework experimental evaluation) and we calculated the response of each baseline tracker at the given ground truth position. From its $2D$ response $\boldsymbol{R}_i$, we extract 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 3. 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. It should be noted that in VOT2017 both partial and full occlusions are denoted as occlusions, thus, the same applies to the labels of the extracted feature vectors.

Table 1
Evaluation of our proposed Occlusion detector in terms of mAP.

Method	**Proposed**	$max(\boldsymbol{r})$ [31]	PSR-metric [26]
mAP	**92.1 %**	72.0 %	73.7 %

The idea of detecting target occlusions from the tracker responses is not new [31, 26]. One method to detect target occlusions from the tracker responses is to set a threshold T for the maximum tracker response, i.e., $o = 1$ if $T < max(\boldsymbol{r})$ [31], since it is expected that during occlusions, the tracker responses are lower than during normal/no occlusion circumstances. Another interesting approach is to employ the PSR metric [26], i.e., $PSR = \frac{max(\boldsymbol{r}) - mean(\boldsymbol{r})}{std(\boldsymbol{r})}$, by setting a threshold in a similar fashion $o = 1$ if $T < PSR$.

The proposed method was evaluated against these two approaches on the VOT 2017 dataset, in a $5-$fold cross validation procedure, i.e., using 5 data splits where $4/5$ of the dataset were employed for training the classifier and determining the optimal parameters for the competing methods and $1/5$ was used for testing purposes. The optimal threshold T and the SVM parameter c of the proposed method were determined using grid search, while the introduced parameter β was set to $\beta = 0$ for this task. In Table 1 we report the obtained mean average precision between the 5 folds for all methods in terms of detecting the full and partial target occlusions that were annotated in the dataset. As can be seen, the proposed method greatly outperformed the heuristic methods, by a margin of 20%, hence being much more accurate for determining partial and full occlusions.

4.1.1. *Effect of adjusting the bias term β*

Detecting occlusions and successful target tracking are two different problems. The pre-trained occlusion detection model does not discriminate between full and partial occlusions. Moreover, it cannot be perfect, some cases perhaps due to extreme camera motion or heavy illumination changes may be misclassified. Some of these cases are necessary for training the tracking model. Therefore, the introduced bias term β should be adjusted in such manner, in order to focus on detecting medium/heavy occlusions, that is optimal for successful tracking.

In order to demonstrate the effect of different values of the bias term, we employed a subset of the TC128 [28] dataset, that incorporates the 78 uniquely appearing videos, which cannot be found in other datasets. KCF [18], Staple [1] and Staple-CA [33] were employed as baseline tracking modules. Table 2 demonstrates the obtained tracking precision results for different values of the bias term. As can be seen, neighboring values of the bias term have a small impact on tracking performance.

Table 2
Sensitivity analysis of different values of the bias term β on the subset of TC128, in term of 20 px tracking precision (TP), i.e., the percentage of the frames where the center of the tracked ROI is closer than 20 pixels to the ground truth ROI center position.

KCF	β	-0.15	-0.2	-0.25	-0.3	-0.35
	20 px TP	0.490	0.485	**0.497**	0.478	0.464
Staple	β	-1.6	-1.65	-1.7	-1.75	-1.8
	20 px TP	0.660	0.659	**0.661**	0.657	0.647
Staple-CA	β	-0.45	-0.5	-0.55	-0.6	-0.65
	20 px TP	0.620	0.626	**0.637**	0.629	0.614
LDES-no	β	-0.4	-0.45	-0.5	-0.55	-0.6
	20 px TP	0.718	0.722	**0.724**	0.723	0.723

4.2. Framework evaluation

In order to evaluate the proposed framework, we have employed three challenging, publicly available 2D tracking benchmark datasets. In addition, in order to evaluate its performance on UAV cinematography, we have employed the realistic video database for a UAV application, namely the BikeUAV, by collecting footage from bicycle races. Detailed description of the datasets employed in our experiments is given in Subsection 4.3.

KCF [18], LDES [27] without the rotation calculation module indicated as LDES-no, Staple [1] and Staple-CA [33] were employed as baseline trackers of our proposed framework. The tracking performance was evaluated against the same baseline trackers, without incorporating our framework, as well as with other state-of-the-art correlation-filter based methods, such as BACF [11] and SRDCF [8], as well as with algorithms that have occlusion detection/long-term tracking mechanisms, i.e., ROT [10] and LCT [31]. We did not employ any 2D tracking methods based on deep learning to this end, since the proposed framework is evaluated for potential UAV implementation in the near future. Embedded/UAV systems are not equipped with the necessary hardware options required to employ such deep learning methods for real time implementation, yet.

All experiments were conducted on an Intel NUC which has a small footprint, low weight and power consumption, making it ideal for UAV applications. It is equipped with an Intel Core i7 processor running at 3.5 GHz and 8 GB of RAM.

The competing algorithms were implemented in MATLAB, using the code provided by their respective authors and with their default parameter settings. The same parameter settings were kept in all our experiments. Performance is assessed in both quantitative and qualitative terms, as described in Subsections 4.4 and 4.5, respectively.

4.3. Datasets

For the evaluation of the proposed framework we utilized the videos from three publicly available video tracking benchmarks, namely the OTB100 [45], Temple-Color128 (TC128) [28] and UAV123 [32].

The **OTB100** is a standard 2D tracking benchmark dataset that consists of 100 video sequences with challenging attributes such as illumination and scale variation, in and out-of-plane target rotations, fast target and camera motion, target deformations, background clutter and finally, target occlusions as well. It was specially designed to determine tracking performance in several circumstances, some of them being relevant to the UAV cinematography.

UAV123 is a more UAV specific video tracking benchmark dataset, since it contains sequences captured from a low-altitude aerial perspective. This dataset provides 123 fully annotated sequences containing a wide variety of scenes (e.g., urban landscape, roads, buildings, fields, beaches and a harbor/marina). In addition to the OTB100 attributes, this dataset includes videos, where the viewpoint affects target appearance significantly (Viewpoint Change). In some videos there are objects of similar shape or of same type near the target (Similar Object).

TC128 is a video tracking benchmark dataset which consists of 128 color videos. The challenging attributes appearing in the sequences of this dataset are the same as in the UAV123.

BikeUAV is a dataset collected by the authors that showcases a realistic UAV cinematography case scenario. It contains aerial footage from famous bike races, e.g., Giro d'Italia, consisting of 21 video sequences captured from helicopter/UAV cameras. Most of the sequences have been employed for live broadcasting. The dataset is very challenging, since the targets suffer from long-term partial or full occlusions. There are visually similar objects around the target, fast moving targets and other challenging attributes. Example frames of this dataset are shown in the first row of Figure 8.

4.4. Quantitative tracking performance evaluation

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 framework application domain.

In order to examine the performance of the proposed framework we employ the Center Location Error (CLE) which is a widely used evaluation metric for object tracking. It is computed by measuring the Euclidean distance in pixels, between the center locations of the tracker output ROIs and the ground truth ROIs in all frames. Tracking precision measure is given by calculating the percentage of frames, in which the estimated locations are within a given threshold, e.g., 20 pixels. It should be noted that this metric indicates if the position of the tracker's output is close to the target center, but cannot evaluate tracker's performance on size and scale variations of the tracked object. This metric is valuable for UAV tracking applications, since independent of correct scale estimation, if the center position of the tracker output is close to the ground truth, the UAV can still follow the desired object.

Table 3
Tracking evaluation in OTB-100. Bold values indicate the proposed ODDA framework implementation results and red values indicate the top performance for each evaluation metric.

Tracker	Precision (20 px)	Success rate (0.5)	FPS
KCF	0.695	0.477	335.8
KCF-ODDA	**0.720**	**0.497**	**152.9**
LDES-no	0.786	0.612	69.1
LDES-ODDA	**0.795**	**0.616**	**40.6**
Staple	0.784	0.581	72.9
Staple-ODDA	**0.799**	**0.593**	**62.1**
Staple-CA	0.810	0.600	50
Staple-CA-ODDA	**0.813**	**0.601**	**41.5**
BACF	0.797	0.603	34.1
ROT	0.699	0.519	49.8
LCT	0.762	0.562	24.2
SRDCF	0.789	0.598	12.7

We also use the Overlap Score (OS) defined as $S = \frac{|r_t \cap r_0|}{|r_t \cup r_0|}$, 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. Its average value resulting on all frames can be used as the tracking success metric. When the value of S is larger than a certain threshold, it is assumed that the tracker, successfully tracks the desired object. In the quantitative evaluation section, we have selected an overlap score threshold of 0.5, which is the standard value reported in related benchmarks. 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. Thus it does not favor algorithms with no target scale adaption mechanisms, such as KCF.

4.4.1. Center Location Error and Overlap Score results

Table 3 shows the evaluation results of the proposed framework at the OTB100 dataset, in terms of precision, success rate and tracking speed (in FPS) when applied to the three baseline tracking algorithms against the baseline methods, as well as other real-time 2D tracking methods. As can be seen, the top precision is achieved by the proposed LDES-ODDA which performs slightly better than the respective baseline version. In addition, all of the baseline tracker performances were enhanced by applying the proposed ODDA framework. More specifically, the performance of the rest baseline trackers were improved by 2.5% for KCF, Staple-CA's by 0.3% and 1.5% for Staple, respectively. The success rate performance of KCF was improved by 2%, LDES's by 0.4%, Staple's by 1.2% and Staple-CA's by 0.1%. It should be noted that Staple-ODDA manages to outperform in terms of precision, much more demanding trackers in terms of computational burden, such as SRDCF. For instance, SRDCF manages to achieve an average tracking speed of $14fps$ while Staple-ODDA has an average speed of $41fps$. Baseline KCF achieves the top speed ($384fps$), while KCF-ODDA achieves the second best ($179fps$). The impact of applying the proposed method in KCF is high, since the initial complexity in order to obtain $\boldsymbol{r}$ of KCF is very low, comparable to the required complexity for detecting occlusions. For LDES-ODDA, Staple-ODDA and Staple-CA-ODDA, the impact of the proposed framework is mostly related to re-detection occurrences, since the complexity required to evaluate the correlation filter response is only a small fraction of the total baseline tracking complexity.

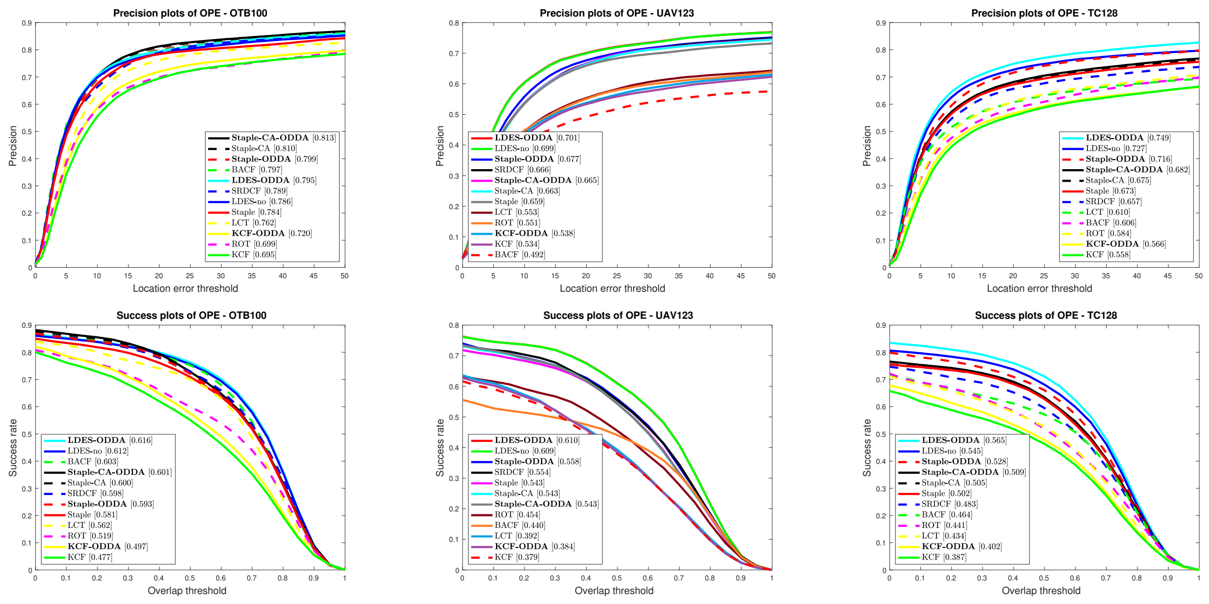
Figure 4 presents the success plots of OS and precision plots of CLE metric for the competing tracking algorithms on the three challenging datasets OTB100, UAV123 and TC128. It should be noted that the ODDA versions of all the baseline trackers achieve better performance for both metrics in all of the datasets.

In UAV123, LDES-ODDA has the best performance overall and Staple-ODDA improves the baseline version by +1.4% for precision and +1.4% for the success rate metrics. Again, a slightly better performance of Staple-CA-ODDA over baseline Staple-CA is observed and KCF-ODDA improves the baseline KCF performance as well. From the rest of the competing tracking algorithms, only SRDCF has a performance close to Staple-ODDA and Staple-CA-ODDA methods, however it is much slower.

In TC128, LDES-ODDA achieves top performance outperforming the baseline version by 2.2% in terms of CLE precision and 2% in success rate score. Staple-ODDA outperformed the baseline version as well, having a performance margin of 4.3% in precision and 2.6% in success rate, when compared to the Staple tracking algorithm. Staple-CA-ODDA achieves a good performance as well and all of these tracking algorithms perform significantly better than other state-of-the art tracking algorithms such as SRDCF, as well as LCT and BACF.

Figure 5 demonstrates the precision and success plots of the comparing tracking algorithms in BikeUAV dataset. Staple-ODDA manages to achieve the top performance in precision (0.815) and success rate (0.737) having improved the baseline tracker Staple by +7.6% for both metrics. KCF-ODDA has the second best performance in precision (0.792) with an improvement over the KCF of +4.4%. As this tracker does not have an object scale adaptation mechanism, the performance in the overlap error metric is lower than the rest of the trackers. However our framework improved the success rate by 4%. The ODDA framework also improved the results for both LDES-no and Staple-CA.

Table 4 summarizes the results in terms of precision for all datasets used for evaluation. Staple-ODDA manages to achieve the top performance in most of the benchmarks and the top performance in average precision. Staple-CA-ODDA is the second best performing tracker with an average precision of 0.729. KCF-ODDA managed to outperform methods that have been specifically designed for handling occlusions, such as ROT and LCT, even though it does not in-



(a) (b) (c)

Figure 4: Precision and success plots plots for the three challenging benchmarks: a) OTB-100, b) UAV123 and c) TC128.

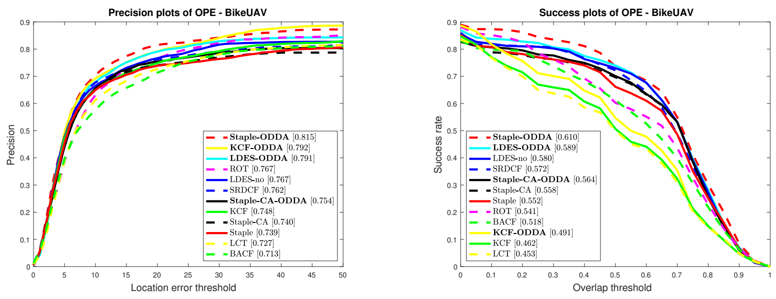


Figure 5: Precision and success plots for the real case scenario dataset BikeUAV.

clude any mechanism to handle object scale changes. This is even more obvious when comparing ROT and LCT with the proposed Staple-ODDA and Staple-CA-ODDA, which do include a mechanism for handling target scale.

4.4.2. *Attribute evaluation on UAV123*

Here we present quantitative evaluation on videos with specific attributes from the UAV123 dataset. Precision and success rate plots for center location error and overlap score respectively are shown in Figure 6.

Attribute evaluation includes four group of videos with special attributes. The first one consists of sequences where the camera motion can result to extreme 2D motion of the tracked object in consecutive video frames. Furthermore, the target can appear blurry, which can result to a mis-trained correlation filter. By examining the precision and success plots for this attribute it appears that our proposed framework aids the baseline trackers to improve their performance. For instance, LDES-ODDA bypasses all other tracking methods and Staple-ODDA improves its precision score by 2.1%. Staple-CA-ODDA and KCF-ODDA have also an improved performance compared to their baseline versions.

In the second group of videos, the target is fully occluded in some of the video frames. Here SRDCF marginally outperforms Staple-ODDA in both metrics. Although, the proposed framework enhanced trackers do not achieve the best performance, it must be noted that SRDCF is a much slower method. In the third group of videos with illumination variation of the target, Staple-ODDA has the top precision performance. Finally, the last group consist of videos where some portion of the target is out-of-view. Here, once more the proposed framework improves the performance of the baseline tracking methods.

4.5. Qualitative evaluation

In this section, we describe the qualitative evaluation of our proposed framework when implemented on top of the baseline methods. To this end, we showcase the differences in qualitative terms when compared with the baseline methods, as well as with related methods that handle target occlusions.

Figure 7 depicts the occlusion handling capabilities of

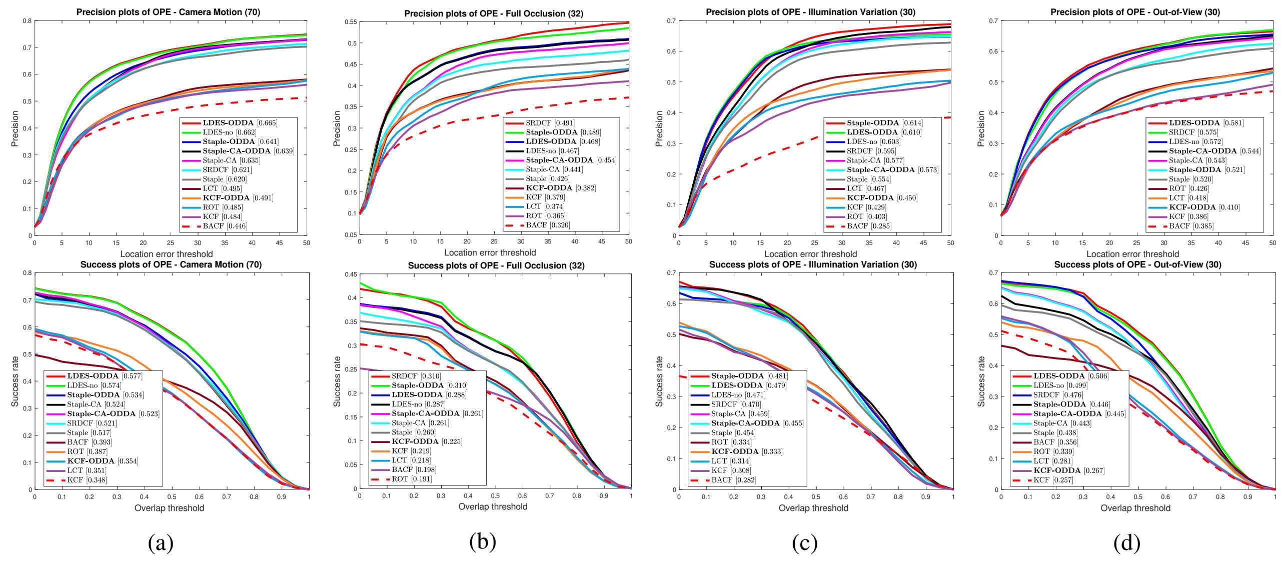


Figure 6: Precision and success plots in videos having: a) Camera Motion, b) Full Occlusion, c) Low Resolution and d) Out-of-View attributes in the UAV123 dataset.

Table 4
Precision results of each tracker evaluated on four video datasets and average precision per tracker. LDES-ODDA achieves the top performance in average precision while Staple-ODDA the second one. KCF-ODDA improves the performance of the baseline KCF achieving similar average precision with other tracking algorithms. Bold results indicate ODDA framework implementation and red results indicate the top performance.

	OTB100	TC128	UAV123	BikeUAV	Avg. Precision
KCF	0.695	0.558	0.534	0.748	0.634
KCF-ODDA	**0.720**	**0.566**	**0.538**	**0.792**	**0.654**
LDES-no	0.786	0.727	0.699	0.767	0.745
LDES-ODDA	**0.795**	**0.749**	**0.701**	**0.791**	**0.759**
Staple	0.784	0.673	0.659	0.739	0.714
Staple-ODDA	**0.799**	**0.716**	**0.677**	**0.815**	**0.752**
Staple-CA	0.810	0.675	0.663	0.740	0.722
Staple-CA-ODDA	**0.813**	**0.682**	**0.665**	**0.754**	**0.729**
BACF	0.797	0.606	0.492	0.713	0.652
ROT	0.699	0.584	0.551	0.768	0.651
LCT	0.762	0.610	0.553	0.727	0.663
SRDCF	0.789	0.657	0.666	0.762	0.719

our proposed framework. To this end, we employ the 'Jogging', 'Lemming' and 'Girl2' OTB100 video sequences, and demonstrate the results of KCF, Staple and Staple-CA and their ODDA versions on the same video sequences. As can be seen, these sequences contain severe occlusions that the baseline trackers fail to handle. The proposed framework however, allows tracking continuation by timely detecting the occlusions and re-detecting the target once it re-appears in a subsequent video frame.

In Figure 8, we demonstrate the obtained results of the framework implemented on KCF tracker, in comparison with state-of-the-art trackers having mechanisms for occlusion handling. The top video sequence shows a bike race from the BikeUAV database, where the target (bicycle) is occluded by the yellow hot-air balloon. All trackers fail to detect the occlusion and to continue tracking the target correctly, except the proposed KCF-ODDA. The middle examples are from the 'BlurOwl' video sequence of OTB100, where there is very fast motion, which causes the center location of target bounding box to change position significantly in consecutive frames. In addition, due to the fast camera motion the target is blurred in many video frames. As can be seen, the proposed KCF-ODDA is the only tracker that successfully tracks the target in this video. Although there are no occlusions in this video, KCF-ODDA handles the rapid camera motion as occlusion, as it drives the target outside the default search window. Therefore, it overcomes rapid camera motion by employing the re-detection function, that increases the search window. Finally, the last sequence that is used to demonstrate our framework performance is 'Box' from the OTB100 video tracking benchmark dataset. In this sequence a small box is moving/rotating inside the video frame, so

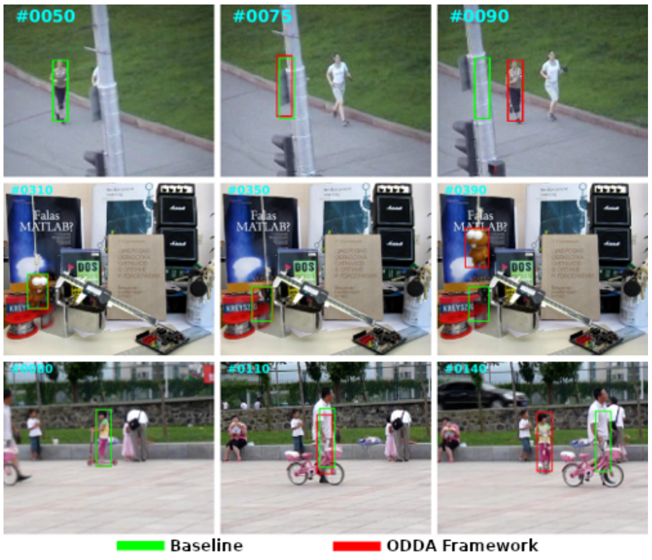


Figure 7: From top to bottom. KCF and KCF-ODDA implementation on 'Jogging'. Staple and Staple-ODDA on 'Lemming'. Staple-CA and Staple-CA-ODDA on 'Girl2'. Our framework allows baseline trackers to overcome the occlusions.

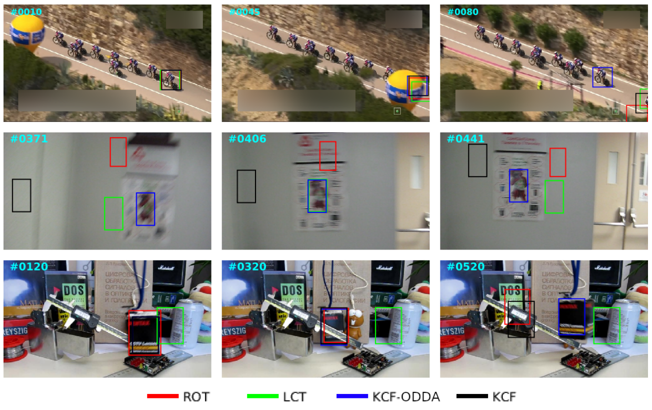


Figure 8: Evaluation of the proposed framework using KCF as baseline tracking algorithm, against related methods that handle occlusions. First row depicts the obtained results in a sequence from BikeUAV, second and third rows depict the obtained results on 'BlurOwl' and 'Box' videos of OTB100.

that it is partially and full occluded by other objects. In this video, the baseline KCF is affected by the partial/full object occlusions or rotations and fails to continue tracking the object. LCT and ROT, do not detect all occlusions successfully. Therefore, their tracking model is polluted and the tracker fails after some video frames. On the contrary, the occlusion handling mechanism of KCF-ODDA results in successful target tracking.

5. Conclusion

This paper presents a generic, easy to implement framework, employed on top of correlation filter based trackers, that decreases tracker drifts and tracking failures caused mainly by object occlusions. This enhances the performance of the baseline tracking method, by reducing the cases where the tracker model was updated (when it should have not been), especially when occlusions occur. Moreover, when the framework detects tracking failure, object re-detection is allowed if the object appears inside the re-detection area after some frames, by employing the already trained tracking filter, without the need of a separate detector. This feature of the framework makes it more lightweight and as the evaluation of the framework indicates suitable for UAV implementation.

Future work includes implementing and evaluating the proposed framework on a UAV hardware platforms. Since UAV hardware platforms are evolving and may contain GPU modules in the near future, the proposed method may be applied on top of the most advanced deep learning-based methods. In addition to developing a classification model that detects occlusions, models that detect abrupt motion changes or target scale changes from the tracker responses, would also be an interesting future work direction.

Acknowledgement

This work has received funding from two European Union's Horizon 2020 research and innovation programmes under grant agreement No 731667 (MULTIDRONE) and 871479 (AERIAL-CORE). 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.

References

[1] Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H., 2016a. Staple: Complementary learners for real-time tracking. Computer Vision and Pattern Recognition (CVPR) , 1401–1409.

[2] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H., 2016b. Fully-convolutional siamese networks for object tracking. European conference on computer vision (ECCV) , 850–865.

[3] Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M., 2010. Visual object tracking using adaptive correlation filters. Computer Vision and Pattern Recognition (CVPR) , 2544–2550.

[4] Chang, C.C., Lin, C.J., 2011. Libsvm: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) 2, 27.

[5] Chrysos, G.G., Antonakos, E., Zafeiriou, S., Snape, P., 2015. Offline deformable face tracking in arbitrary videos, in: Proceedings of the IEEE international conference on computer vision workshops, pp. 1–9.

[6] Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M., 2017a. Eco: efficient convolution operators for tracking. Computer Vision and Pattern Recognition (CVPR) , 21–26.

[7] Danelljan, M., Häger, G., Khan, F.S., Felsberg, M., 2017b. Discriminative scale space tracking. IEEE transactions on pattern analysis and machine intelligence 39, 1561–1575.

[8] Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M., 2015. Learning spatially regularized correlation filters for visual tracking. International Conference on Computer Vision (ICCV) , 4310–4318.

[9] Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M., 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking. European Conference on Computer Vision (ECCV) , 472–488.

[10] Dong, X., Shen, J., Yu, D., Wang, W., Liu, J., Huang, H., 2017.

Occlusion-aware real-time object tracking. IEEE Transactions on Multimedia 19, 763–771.

[11] Galoogahi, H.K., Fagg, A., Lucey, S., 2017. Learning background-aware correlation filters for visual tracking. Computer Vision and Pattern Recognition (CVPR) , 21–26.

[12] Gladh, S., Danelljan, M., Khan, F.S., Felsberg, M., 2016. Deep motion features for visual tracking. International Conference on Pattern Recognition (ICPR) , 1243–1248.

[13] Gundogdu, E., Alatan, A.A., 2018. Good features to correlate for visual tracking. IEEE Transactions on Image Processing 27, 2526–2540.

[14] Gundogdu, E., Ozkan, H., Alatan, A.A., 2017. Extending correlation filter-based visual tracking by tree-structured ensemble and spatial windowing. IEEE Transactions on Image Processing 26, 5270–5283.

[15] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M.M., Hicks, S.L., Torr, P.H., 2016. Struck: Structured output tracking with kernels. IEEE transactions on pattern analysis and machine intelligence 38, 2096–2109.

[16] Held, D., Thrun, S., Savarese, S., 2016. Learning to track at 100 fps with deep regression networks. European Conference on Computer Vision (ECCV) , 749–765.

[17] Henriques, J.F., Caseiro, R., Martins, P., Batista, J., 2012. Exploiting the circulant structure of tracking-by-detection with kernels. European conference on computer vision (ECCV) , 702–715.

[18] Henriques, J.F., Caseiro, R., Martins, P., Batista, J., 2015. High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 37, 583–596.

[19] Kalal, Z., Mikolajczyk, K., Matas, J., 2010. Face-tld: Tracking-learning-detection applied to faces, in: 2010 IEEE International Conference on Image Processing, IEEE. pp. 3789–3792.

[20] Kalal, Z., Mikolajczyk, K., Matas, J., 2012. Tracking-learning-detection. IEEE transactions on pattern analysis and machine intelligence 34, 1409–1422.

[21] Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Čehovin Zajc, L., Vojir, T., Häger, G., Lukežič, A., Eldesokey, A., Fernandez, G., 2017. The visual object tracking vot2017 challenge results.

[22] Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L., 2016. A novel performance evaluation methodology for single-target trackers. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 2137–2155. doi:10.1109/TPAMI.2016.2516982.

[23] Lee, H.S., Kim, D., Lee, S.Y., 2003. Robust face-tracking using skin color and facial shape, in: International Conference on Audio-and Video-Based Biometric Person Authentication, Springer. pp. 302–309.

[24] Lee, K.C., Ho, J., Yang, M.H., Kriegman, D., 2005. Visual tracking and recognition using probabilistic appearance manifolds. Computer Vision and Image Understanding 99, 303–331.

[25] Lee, K.C., Kriegman, D., 2005. Online learning of probabilistic appearance manifolds for video-based recognition and tracking, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE. pp. 852–859.

[26] Li, R., Pang, M., Zhao, C., Zhou, G., Fang, L., 2016. Monocular long-term target following on uavs. Conference on Computer Vision and Pattern Recognition (CVPR) , 29–37.

[27] Li, Y., Zhu, J., Hoi, S.C., Song, W., Wang, Z., Liu, H., 2019. Robust estimation of similarity transformation for visual object tracking, in: The Conference on Association for the Advancement of Artificial Intelligence (AAAI).

[28] Liang, P., Blasch, E., Ling, H., 2015. Encoding color information for visual tracking: Algorithms and benchmark. IEEE Transactions on Image Processing 24, 5630–5644.

[29] Liu, F., Gong, C., Huang, X., Zhou, T., Yang, J., Tao, D., 2018. Robust visual tracking revisited: From correlation filter to template matching. IEEE Transactions on Image Processing 27, 2777–2790.

[30] Ma, C., Huang, J., Yang, X., Yang, M., 2017. Adaptive correlation filters with long-term and short-term memory for object tracking. CoRR abs/1707.02309. URL: http://arxiv.org/abs/1707.02309, arXiv:1707.02309.

[31] Ma, C., Yang, X., Zhang, C., Yang, M.H., 2015. Long-term correlation tracking. Computer Vision and Pattern Recognition (CVPR) , 5388–5396.

[32] Mueller, M., Smith, N., Ghanem, B., 2016. A benchmark and simulator for uav tracking. European Conference on Computer Vision (ECCV) , 445–461.

[33] Mueller, M., Smith, N., Ghanem, B., 2017. Context-aware correlation filter tracking. Computer Vision and Pattern Recognition (CVPR) , 1396–1404.

[34] Nam, H., Han, B., 2016. Learning multi-domain convolutional neural networks for visual tracking. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) .

[35] Qi, Y., Zhang, S., Qin, L., Huang, Q., Yao, H., Lim, J., Yang, M.H., 2018. Hedging deep features for visual tracking. IEEE transactions on pattern analysis and machine intelligence 41, 1116–1130.

[36] Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J., Yang, M.H., 2016. Hedged deep tracking, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4303–4311.

[37] Sánchez-Lozano, E., Martinez, B., Tzimiropoulos, G., Valstar, M., 2016. Cascaded continuous regression for real-time incremental face tracking, in: European Conference on Computer Vision, Springer. pp. 645–661.

[38] Schölkopf, B., Herbrich, R., Smola, A.J., 2001. A generalized representer theorem. International Conference on Computational Learning Theory , 416–426.

[39] Schwerdt, K., Crowley, J.L., 2000. Robust face tracking using color, in: Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), IEEE. pp. 90–95.

[40] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556.

[41] Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Statistics and computing 14, 199–222.

[42] Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R., Yang, M.H., 2017. Crest: Convolutional residual learning for visual tracking. International Conference on Computer Vision (ICCV) .

[43] Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., Torr, P.H., 2017. End-to-end representation learning for correlation filter based tracking. Computer Vision and Pattern Recognition (CVPR) , 5000–5008.

[44] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H., 2019. Fast online object tracking and segmentation: A unifying approach, in: Proceedings of the IEEE conference on computer vision and pattern recognition.

[45] Wu, Y., Lim, J., Yang, M.H., 2015. Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence 37, 1834–1848.

[46] Zhang, T., Ghanem, B., Xu, C., Ahuja, N., 2013. Object tracking by occlusion detection via structured sparse learning. Computer Vision and Pattern Recognition (CVPR) , 1033–1040.

[47] Zhang, T., Liu, S., Xu, C., Liu, B., Yang, M.H., 2018. Correlation particle filter for visual tracking. IEEE Transactions on Image Processing 27, 2676–2687.

[48] Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W., 2018. Distractor-aware siamese networks for visual object tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 101–117.

[49] Zoidi, O., Tefas, A., Pitas, I., 2013. Visual object tracking based on local steering kernels and color histograms. IEEE Transactions on Circuits and Systems for Video Technology 23, 870–882.

Iason Karakostas received the Diploma of Electrical and Computer Engineering in 2017 and is currently a PhD Student at the Artificial Intelligence and Information Analysis Laboratory (AIIA) in the Department of Informatics of AUTH. He has co-authored 9 papers in academic journals and international conferences and has participated in two European Union-funded R&D projects. His current research interests include machine learning, computer vision, autonomous robotics and intelligent cinematography.

Vasileios Mygdalis Vasileios Mygdalis received the B.Sc. degree in Biomedical Informatics from the University of Central Greece in 2010. He was awarded the M.Sc. degree in biomedical Informatics in 2014 and the Ph.D. degree in Informatics in 2019, both from the Aristotle University of Thessaloniki. During his Ph.D. studies, he served as a researcher and teaching assistant in the fields of machine learning, image processing, computer vision and pattern recognition. Vasileios is currently a research associate at the Artificial Intelligence and Information Analysis laboratory, in the Department of Informatics, Aristotle University of Thessaloniki, Greece. He has (co-)authored more than 25 peer-reviewed papers in academic journals and international conferences. His current research interests include the areas of machine learning, computer/robotic vision and autonomous systems.

Anastasios Tefas received the B.Sc. and Ph.D. degrees in Informatics from the Aristotle University of Thessaloniki, Greece, in 1997 and 2002, respectively. From 1997 to 2002, he was a Researcher and Teaching Assistant with the Department of Informatics, University of Thessaloniki. From 2003 to 2004, he was a Temporary Lecturer with the Department of Informatics, University of Thessaloniki. From 2006 to 2008, he was an Assistant Professor with the Department of Information Management, Technological Institute of Kavala. From 2008 to 2012, he was a Lecturer with the Aristotle University of Thessaloniki. Since 2013, he has been an Assistant Professor with the Department of Informatics, Aristotle University of Thessaloniki. He has participated in 15 research projects financed by national and European funds. He has coauthored more than 70 journal papers and 180 papers in international conferences, and contributed eight chapters to edited books in his area of expertise. Over 5000 citations have been recorded to his publications and his H-index is 37 according to Google Scholar. His current research interests include computational intelligence, pattern recognition, statistical machine learning, digital signal and image processing, computer vision, and biometrics and security.

Prof. **Ioannis Pitas** (IEEE Fellow, IEEE Distinguished Lecturer, EURASIP Fellow) received the Diploma and PhD degree in Electrical Engineering, both from the Aristotle University of Thessaloniki (AUTH), Greece. Since 1994, he has been a Professor at the Department of Informatics of AUTH and Director of the Artificial Intelligence and Information Analysis (AIIA) lab. He served as a Visiting Professor at several Universities. His current interests are in the areas of computer vision, machine learning, autonomous systems, intelligent digital media, image/video processing, human-centred interfaces, affective computing, 3D imaging and biomedical imaging. He has published over 906 papers, contributed in 47 books in his areas of interest and edited or (co-)authored another 11 books. He has also been member of the program committee of many scientific conferences and workshops. In the past he served as Associate Editor or co-Editor of 9 international journals and General or Technical Chair of 4 international conferences. He participated in 70 R&D projects, primarily funded by the European Union and is/was principal investigator/researcher in 42 such projects. He has 30600+ citations to his work and h-index 82+ (Google Scholar).