# An Approach to Overcome Occlusions in Visual Tracking: By Occlusion Estimating Agency and Self-Adapting Learning Rate for Filter's Training

Kaiwen Jiang, Feng Qian, Ce Song, and Bao Zhang

***Abstract*—Visual tracking methods have been successful in recent years. Correlation filter (CF) based methods significantly advanced state-of-the-art tracking. The advancement in CF tracking performance is predominantly attributed to powerful features and sophisticated online learning formulations. However, there would be trouble if the tracker indiscriminately learned samples. Particularly, when the target is severely occluded or out-of-view, the tracker will continuously learn the wrong information, resulting target loss in the following frames. In this study, aiming to avoid incorrect training when occlusions occur, we propose a regional color histogram-based occlusion estimating agency (RCHBOEA), which estimates the occlusion level and then instructs, based on the result, the tracker to work in one of two modes: normal or lost. In the normal mode, an occlusion level-based self-adopting learning rate is used for tracker training. In the lost mode, the tracker pauses its training and conducts a search and recapture strategy on a wider searching area. Our method can easily complement CF-based trackers. In our experiments, we employed four CF-based trackers as a baseline: discriminative CFs (DCF), kernelized CFs (KCF), background-aware CFs (BACF), and efficient convolution operators for tracking: hand-crafted feature version (ECO_HC). We performed extensive experiments on the standard benchmarks: VIVID, OTB50, and OTB100. The results demonstrated that combined with RCHBOEA, the trackers achieved a remarkable improvement.**

***Index Terms*—Overcome occlusion, regional color histogram (RCH), self-adopting learning rate, visual tracking.**

## I. Introduction

VISUAL object tracking is a fundamental task of computer vision, and it plays a crucial role in numerous real-time vision applications (e.g., robotics, visual surveillance, and human–computer interaction). In recent decades, visual object tracking has achieved great success. Based on tracking benchmarks, correlation filter (CF) based approaches [1]–[5] have shown continuous performance improvements in terms of accuracy and robustness. The advancement in CF tracking performance is predominantly attributed to sophisticated online learning formulations and powerful features, such as the histogram of gradients [2], [6], [7], color name (CN) [8], and convolutional neural networks (CNN) feature [2], [3]. Tracker online learning is important for improving robustness. However, there will be problems if the tracker collects the training sample indiscriminately. The reason is that the tracker will continuously learn the wrong samples when the target is severely occluded or missing. In this situation, the tracker will probably fail to capture the target again when it reappears in consecutive frames.

Typical methods for dealing with occlusions incorporate a detector into the tracker, such as Kalal's [10] TLD framework or Ma's [7] LCT framework. Recently, Fan *et al*. [11] proposed a parallel tracking and verifying framework, in which a sophisticated CNN-based tracker verified the high efficient tracker's results. However, all those methods were unsatisfactory regarding long-term and total occlusion. Alternatively, James *et al*. [12] proposed an online decision-making tracker, but it required offline reinforcement learning.

In this study, we propose a regional color histogram-based occlusion estimating agency (RCHBOEA), which estimates the occlusion level by the regional color histogram (RCH) of the target area. The tracker adopts different learning and searching strategies based on the occlusion level. This method showed satisfactory performance in experiments.

In addition, our RCHBOEA could be combined with any CF-based online training tracker. In our experiments, we employ four CF-based trackers as the baseline. These include two recent state-of-the-art trackers: background-aware CFs (BACF) [13] and efficient convolution operators for tracking: hand-crafted feature version (ECO_HC) [2]. We also include two prior trackers: discriminative CFs (DCF) [4] and kernelized CFs (KCF) [4]. Those baseline trackers all achieve impressive performance both in efficiency and robustness. However, when it comes to severe occlusion, even state-of-the-art trackers are powerless [see Fig. 1(a)]. Fortunately, once improved by our method, trackers can successfully recapture the target postocclusion [see Fig. 1(b)].

## II. Our Approach

As previously discussed, one approach of coping with severe occlusion is estimating the occlusion level. We employ RCH for its two notable benefits. First, as a statistical feature, RCH

Manuscript received May 20, 2018; revised June 27, 2018; accepted June 30, 2018. Date of publication July 13, 2018; date of current version November 9, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61705225. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wei Li. *(Corresponding author: Bao Zhang.)*

K. Jiang is with the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: jiangkaiwen16@mails.ucas.ac.cn).

F. Qian, C. Song, and B. Zhang are with the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China (e-mail: tsienfung@126.com; songce528@163.com; zhangb@ciomp.ac.cn).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2018.2856102

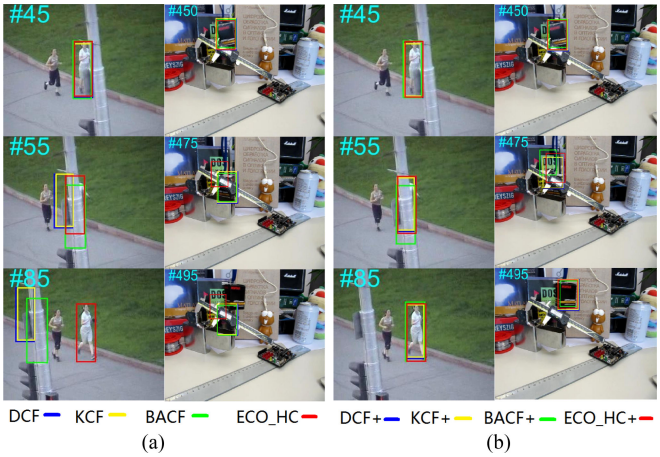


Fig. 1. Some of results on OTB100. (a) Original DCF, KCF, BACF, ECO_HC tracker fail to track when total occlusions happen. (b) Improved trackers could succeed in capturing the target after occluding.

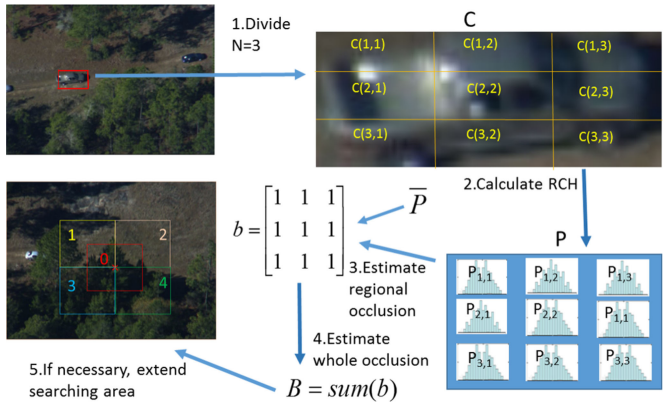


Fig. 2. We divide target area $C$ into $N \times N$ blocks and calculate its color histogram $P_{m,n}(u)$, respectively. Here, $P$ is the RCH in current frame and $\overline{P}$ is target's RCH model obtained from previous frames. And then we calculate the distance between $P$ and $\overline{P}$ to obtain the occlusion level B. Next, the tracker chooses its work modes according to occlusion level. If necessary (when the tracker works in the lost mode), the tracker extends searching area. The way is using the same scale and searching several different locations (label 0, 1, 2, 3, 4) around the target's last location.

is robust to scale changes and noise jamming. Second, the object area is divided into several blocks, thus we can gather the color histogram for each block. This makes it easy to discriminate which block is occlusive and is convenient to estimate the occlusion level of the total object. Our estimation process has two steps. In step 1, we divide the object region into several blocks and obtain the RCH. In step 2, we collect occlusion information from the RCH. These two steps are discussed in Sections II-A and II-B, respectively. In Section II-C, we discuss how the tracker uses the results of the RCHBOEA to determine its work mode (i.e., normal or lost) and introduce the different learning and searching strategies of each work mode.

### A. Obtain RCH

First, the tracker obtains the position and scale of the target in the image $I$ of the current frame. Then, the target area $C$, in this image, will be obtained. We divide $C$ into $N \times N$ blocks (see Fig. 2). In each block, $C(m,n)$ $m,n \in 1,2...,N$, we calculate its color histogram $P_{m,n}(u)$. If the image is a single channel gray image, it is $u \in 1,2,...,D$. If it is an RGB 3-channel color image, it is $u \in 1,2,...,3*D$. Here, $u$ is the gray-level label, and $D$ is a fixed parameter, which is the number of gray levels in one channel. In our algorithm, two groups of RCH are saved. $P = \{P_{m,n}\}$ $m,n \in 1,2...,N$ is the RCH of the object in the current image. $\overline{P} = \{\overline{P}_{m,n}\}$ $m,n \in 1,2...,N$ is the average RCH of the object in the history. $\overline{P}_{m,n}$ and $P_{m,n}$ are normalized. Thus, $\sum_u P_{m,n}(u) = 1$ and $\sum_u \overline{P}_{m,n}(u) = 1$.

The average RCH of the object is updated by a self-adapting learning rate, $\alpha = \{\alpha_{m,n}\}$

$$\overline{P}^t_{m,n} = (1 - \alpha_{m,n}) \cdot \overline{P}^{t-1}_{m,n} + \alpha_{m,n} \cdot P^t_{m,n}. \tag{1}$$

### B. Estimate Occlusion Level by RCH

We obtain the distance $d_{m,n}$ between $P_{m,n}$ and $\overline{P}_{m,n}$

$$d_{m,n} = \sum_u \sqrt{\left(P_{m,n}(u) - \overline{P}_{m,n}(u)\right)^2} \tag{2}$$

where $d_{m,n} \in [0,2]$ and its value corresponds to the occlusion level in this region $C(m,n)$. $u$ is the gray-level label, like the one mentioned in Section II-A.

The self-adapting learning rate $\alpha_{m,n}$ in (1) corresponds to $d_{m,n}$

$$\alpha_{m,n} = \alpha_0 \exp\left(-k_0 \cdot d_{m,n}\right) \tag{3}$$

where $\alpha_0 \in (0,1)$ and $k_0 > 0$ both are fixed parameters. $\alpha_{m,n}$ can also reflect the similarity between the current image and the stored model in this region $C(m,n)$. Equation (3) plays a crucial role in ensuring the accuracy of the object's RCH model $\overline{P}$. If a regional occlusion occurs in $C(m,n)$, most likely, the new state only lasts a short duration. In this situation, $\overline{P}_{m,n}$ will show nearly no change during the occlusion. However, if the change of $P_{m,n}$ is caused by an object's intrinsic change, such as rotation and scale change, the new state may last a long time. Thus, $\overline{P}_{m,n}$ will accelerate toward the new value.

We use a binary factor $b(m,n)$ to represent the region $C(m,n)$ as occlusive (mark 0) or not occlusive (mark 1)

$$b(m,n) = \begin{cases} 1, & \alpha_{m,n} \geq \beta_0 \\ 0, & \alpha_{m,n} < \beta_0 \end{cases} \tag{4}$$

where $\beta_0$ is an empirical threshold.

Finally, we obtain the final occlusion level $B$ of the total object region

$$B = \sum_{m,n} b(m,n) \tag{5}$$

where $B \in [0, N^2]$. The smaller $B$ indicates a higher level of occlusion. A larger $B$ indicates a higher quality of the training sample.

### C. Tracker's Two Work Modes: Normal and Lost

To avoid filter updating in the wrong direction, the tracker, according to the occlusion level, should choose from the two work modes of different learning and searching strategies.

**Algorithm 1:**

**INPUT**:
Image $I(t)$ in current frame $t$;
position $s(t-1)$ and scale $z(t-1)$ of target in frame $t-1$;
**OUTPUT**:
position $s(t)$ and scale $z(t)$ of target in frame $t$;

```
if flag==0
  extend searching area based on s(t-1) and z(t-1);
    do normal search operation on each area;
    if G_M = max(G_i) > g_1 · Ḡ
      calculate B in area M;
      if B > b_1 · B̄
        obtain the position s(t) and scale z(t) of
         target in current frame t;
        flag=1;
      end if
    else
      s(t) = s(t-1), z(t) = z(t-1);
    end if
  else
    the tracker searching in normal model based on
    s(t-1) and z(t-1), obtain s(t) and z(t);
    calculate B;
    if B^t < b_0 · B̄^{t-1} and G^t < g_0 · Ḡ^{t-1}
      flag=0;
    else
      update α, P̄, Ḡ, B̄, ξ;
      update tracker by ξ;
    end if
  end if
```

Factor $B$ in (5) represents the occlusion level of the object. In our formulation, the tracker learns new samples at a self-adopting learning rate $\xi$ instead of a fixed rate.

$$\xi^t = \xi_0 \exp\left(\theta \cdot \min\left((B^t/\overline{B}^{t-1} - 1), 0\right)\right) \tag{6}$$

$$\overline{B}^t = (1-\lambda_1)\overline{B}^{t-1} + \lambda_1 B^t. \tag{7}$$

Here $\xi_0$ is the basis learning rate, and $\overline{B}$ is average of $B$ from the previous number of frames.

We calculate the CF's peak response $G$ in the current image, and we obtain its previous average value $\overline{G}$

$$\overline{G}^t = (1-\lambda_2)\,\overline{G}^{t-1} + \lambda_2 G^t. \tag{8}$$

Here,$\lambda_2$ is the learning rate of the average peak response.

Once $B^t < b_0 \cdot \overline{B}$ and $G^t < g_0 \cdot \overline{G}$, we consider the target in the current frame as severely occluded or missing, and the tracker switches to *lost* mode (see Fig. 3). Otherwise, the tracker maintains its original (baseline) searching method and updates the filter with the self-adopting learning rate, $\xi$. It is a *normal* mode. Here, $b_0 \in (0,1)$ and $g_0 \in (0,1)$ are threshold factors.

In the *lost* mode, the tracker does an operation of searching and recapturing at a wider area. First, it extends the searching area. For CF-based trackers, the scale of the searching area is directly proportional to the scale of the target area. Thus, we cannot independently change the searching scale of a single searching operation. Thus, our method uses the same scale and searches several different locations around the target's last location. Some overlap of the searching area is necessary. As shown in Fig. 2, $i = 0, 1, 2, 3, 4$ is the label of the respective searching area. Next, we estimate whether the target appears in those areas. We perform the original search operation and obtain the filter's peak response $G_i$ for each area. When $G_M = \max\{G_i | i = 0, 1, 2, 3, 4\} > g_1 \cdot \overline{G}$, we calculate $B$ in the area $M$. Then, if $B > b_1 \cdot \overline{B}$, the tracker will consider the target "reappeared" and then recapture it. Here, $g_1 \in (0,1)$ and $b_1 \in (0,1)$ are threshold factors.

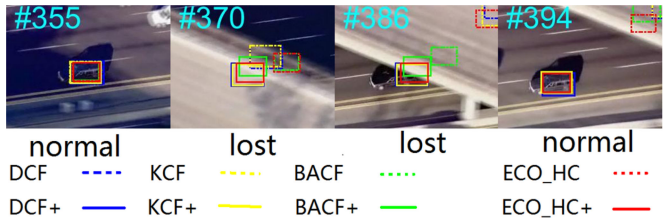


Fig. 3. Lost mode and normal model (for our updated trackers, solid line box). When the target is severe occlusive or out of view, the tracker will work in the lost mode and pause collecting samples. Otherwise, the tracker will work in the normal mode, and update itself by the self-adapting learning rate $\xi$.

The factors, $\xi$, $\overline{B}$, $\overline{P}$, $\overline{G}$, $\alpha$, and the filter are only updated in the *normal* model. The total process is shown in Algorithm 1.

## III. Experiments

We employed four CF-based trackers as our baseline and extensively evaluated our method on three standard datasets: VIVID [14], OTB50 [15], and OTB100 [9]. The updated trackers were also compared with two long-term tracking methods (i.e., TLD [10], LCT [7]) and some state-of-art trackers mentioned in Section I (i.e., CN [8], DeepSRDCF [2], and LMCF [5]).

*Evaluation Methodology:* For the OTB50 and OTB100, we used a metric [15] to evaluate all trackers. Success measured the intersection over union (IoU) of the predicted and ground truth bounding boxes. The success plot shows the percentage of bounding boxes whose IoU score is larger than a given threshold. For the VIVID dataset, some sequences included abundant severe occlusions, which were attractive challenges. However, its ground truth was given on every tenth frame, which could not use the success metric as we did in the OTB50 and OTB100. Thus, we directly provided result bounding boxes in some frames to show the improvements.

*Implementation Details:* The number of divided blocks ($N \times N$) was set to $5 \times 5$. The gray levels of RCH, D, was set to 8. The filter's fixed learning rate of the original baseline trackers were all 0.012–0.018. Thus, we wanted our trackers' self-adopting learning rate to be similar to it in most normal frames and to be a little larger in the highest quality frames. We also sought for it to decline fast in the bad frames. With this goal, after some testing, we performed the following initialization. $\alpha_0$ and $k_0$ in (3) were set to 0.1 and 3.7, respectively. $\xi_0$ and $\theta$ in (6) were set to 0.02 and 3.5, respectively. The threshold $\beta_0$, in (4), was set to 0.015. The factors $\lambda_1$ and $\lambda_2$, in (7) and (8), were set to 0.005 and 0.012, respectively. Thresholds $b_0$ and $b_1$ were set to 0.35 and 0.36, respectively. Thresholds of peak response $g_0$ and $g_1$ were set to 0.49 and 0.50, respectively. We performed our experiments


Fig. 4. Some of results on VIVID. The dotted line boxes are the results of original baseline trackers. The baseline trackers failed to track when total occlusion occurred. The solid line boxes are the results of our improved trackers. The improved trackers succeeded in capturing the target after total occlusion.

using MATLAB on an Intel Core(TM) i5-4430 @3.00-GHz CPU with 4-GB RAM. The average operating speed of our updated trackers was DCF+ 140 fps, KCF+ 105 fps, BACF+ 28 fps, and ECO_HC+ 32 fps, because their baseline trackers were DCF 380 fps, KCF 215 fps, BACF 30 fps, and ECO_HC 44 fps.

### A. Comparison on VIVID

The severe occlusions in the videos from the VIVID benchmark [14] were rigorous challenges for trackers. Besides severe occlusions, the object in *egtest05* included a rotation, and the object in *pktest01* and *egtest04* lacked color and texture information.

In our experiments, the baseline trackers, DFC, KCF, BACF, and ECO_HC, all failed to track the target when the severe occlusion occurred (see the dotted line box in Fig. 4). After combining with RCHBOEA, the updated trackers, ECO_HC+, BACF+, DCF+, and KCF+, all achieved remarkable improvements. DCF+ and KCF+, however, for their original baseline trackers' weaker ability, failed with the video, *pktest01*. All others succeeded in capturing the target again after total occlusion in these videos (see the solid line box in Fig. 4).

### B. Evaluation on OTB50 and OTB100

We compared our improved trackers with their baseline trackers and state-of-the-art trackers, including CN [8], LMCF [5], DeepSRDCF [2], TLD [10], and LCT [7], on OTB50 and OTB100.

Fig. 5 shows the success plots on OTB50 and OTB100. The results show that all the updated trackers achieved an improvement of success (IoU) when compared with their respective baseline trackers. The BACF+ tracker's improvement is the most remarkable, compared with its baseline.

Fig. 6 shows the results with out-of-view and occlusion attributes on OTB50 and OTB100. Obviously, our method markedly improves all baselines tracker performances, which corresponds with our goal: overcoming occlusion. Additionally, the results also show that the final performance of the updated tracker was directly related to the original performance of the baseline tracker.

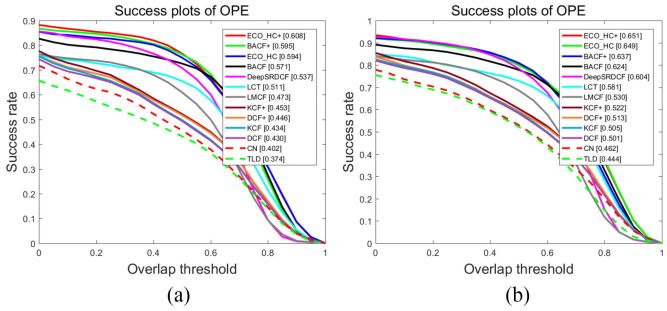


Fig. 5. Success plots comparing our updated trackers, DCF+, KCF+, BACF+, and ECH_HC+, with their baseline trackers and some state-of-the-art trackers on (a) OTB50 and (b) OTB100.

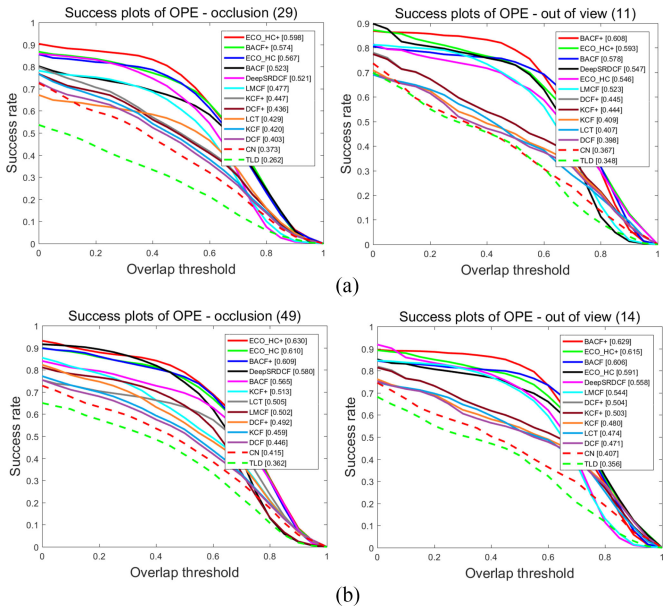


Fig. 6. Success plots with occlusion and out-of-view attributes on (a) OTB50 and (b) OTB 100. Our updated trackers achieved a remarkable performance.

## IV. Conclusion

We proposed an effective algorithm for overcoming severe occlusions in visual tracking. Occlusion estimating agency and self-adapting learning rate are two novel and key points of our approach. Our method included two steps: establish RCHBOEA to estimate the occlusion level of the target; and according to the estimating result, choose normal or lost mode to search and learn. Those operations effectively avoided filter degeneration and helped trackers capture the target again after the severe occlusions or disappearance. Meanwhile, our method is universal and can be easily combined with CF-based trackers. In our experiments, the updated DCF+, KCF+, BACF+, and ECO_HC+ trackers, compared with their original trackers and some state-of-the-art trackers, included two typical long-term tracking methods, all achieved remarkable performance.

## REFERENCES


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

[2] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6931–6939.

[3] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2015, pp. 621–629.

[4] J. F. Henriques, C. Rui, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 702–715.

[5] M. Wang, Y. Liu, and Z. Huang, "Large margin object tracking with circulant feature maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4800–4808.

[6] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 1, 2015.

[7] C. Ma, X. Yang, C. Zhang, and M. H. Yang, "Long-term correlation tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2015, pp. 5388–5396.

[8] M. Danelljan, F. S. Khan, M. Felsberg, and J. v. d. Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1090–1097.

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

[10] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.

[11] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5487–5495.

[12] J. Supancic and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 322–331.

[13] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1144–1152.

[14] R. Collins, X. Zhou, and K. T. Seng, "An open source tracking testbed and evaluation web site," in *Proc. IEEE Int. Workshop Perform. Eval. Tracking Surv.*, 2005.

[15] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2411–2418.