DRIFT DETECTION AND CORRECTION POST-TRACKING

Tarek Ghoniemy and Maria A. Amer

Department of Electrical and Computer Engineering, Concordia University, Montréal, Québec, Canada

ABSTRACT

Accurate object tracking is a challenging problem due to numerous factors, that may cause the tracker to drift away from the target object. Typically, the output of a tracker is a bounding box (BB); such BB may not well discriminate the object from its background and may not be centered correctly around the object. This paper proposes a method that first detects, *at each frame*, if a tracker tends to drift by analyzing saliency features of the output BB of a tracker, and then applies automatic seeded object segmentation on the BB to correct the drift once detected. Such segmentation is meant to relocate (recenter) the BB adaptive to the object segmented. As seeds, we propose to use SIFT and salient points conditioned they are non-background pixels. Different than related work, our approach thus models drift external to a base tracker by examining its output BB at each and corrects drift, as needed, by updating that BB adaptive to segmentation. We show the ability of the proposed method to significantly improve the tracking quality of base trackers. We also show that the proposed method outperforms by far segmentation-based trackers.

Index Terms— Object tracking, drift detection, drift correction, saliency, SIFT points, object segmentation, video.

1. INTRODUCTION

Object tracking is a demanding application: given the initial location of a target in a frame, it estimates the states of the target in subsequent frames under challenging factors that can be object-related (appearance and scale change, deformation, fast motion, motion blur, or occlusion), environment-related (non-stationary scenes, cluttered background, or illumination changes), system-related (real-time and automation constraints), or combinations thereof. These factors may cause the tracker to drift away (to show inaccuracies) from the target object [1, 2, 3, 4]). Drift detection is important, as it allows the tracking algorithm to start tracker reinitialization (drift correction) in order to maximize the tracking accuracy.

Drift detection can be modeled *within* a tracker [5, 6, 7] (e.g., by monitoring tracker's variables) or *post-tracking* by examining the tracker's output *at each frame*. Our contribution in this paper is in the latter category, where we first detect drift using saliency models in the tracker's output BB without prior knowledge and then correct drift by applying automatic seeded object segmentation on the output BB in order to reinitialize the base tracker with a revised BB. To our knowledge no such post-tracking solution exists in the literature.

In [8], a fine Random-Walker (R-Walk) segmentation of an object at any frame is used to initialize the tracking for the next frame. In [9], a closed loop interaction between EM-like color-histogram tracking and R-Walk segmentation has enhanced the accuracy of object localization. The spatial properties and appearance of segmented objects are exploited to initialize the tracking algorithm in the next step. In [10], G-Cut segmentation [11] is applied to mean-shift tracking in a closed loop. In [12], Wen et al. presented a joint tracking and segmentation (JOTS) algorithm which integrates multi-part tracking and segmentation into a unified energy optimization framework. JOTS uses the SLIC super pixel to initialize the tracking at each frame. Running segmentation at each frame has two main drawbacks: first, it is computationally intensive; second, segmentation may become inaccurate which may mislead the tracking. In the above methods, seeds through interactive user input are required.

Different than prior work, our drift detection and correction method 1) is tracker independent and applies to any tracking algorithm; 2) requires no prior information about the target object; 3) uses automatic segmentation with robust seed selection through SIFT, saliency, and intensity features to filter out seeds in background regions; 4) outputs a BB adaptive to object boundaries; 5) applies object segmentation only when a drift is detected and not at each frame.

2. PROPOSED METHOD

The proposed method comprises two main components: drift detection using saliency features and drift correction using seeded segmentation. At current frame F_t, given the estimated tracking output BB from previous frame F_{t-1}, a tracker estimates the BB B_t around the target. If we detect a drift in F_t, our drift correction relocates B_t based on the object mask from a seeded segmentation.

2.1. Saliency-Based Drift Detection

When a tracker tends to drift, the region inside its output BB will start to deviate from *being an object*. The use of objectness measures [13] or saliency models [14] might be thus

useful for drift detection. For saliency of a region, a high contrast to its surrounding regions is usually stronger evidence than that of far-away regions. Generally, an object is more likely to be salient than a region on the background, since image background is usually more structured and homogeneous (thus less salient) than objects.To this end, we propose to use visual saliency to detect if a drift is about to occur.

While most of saliency models [14] employ local contrast, we calculate the saliency map more robustly [15] using global contrast differences and spatial coherence. However, directly introducing the spatial relation among individual pixels is computationally expensive and thus, we partition the BB B_t into K regions and calculate the saliency s_k of each region r_k, $k = 1, \cdots, K$, as a weighted sum of corresponding regions' contrast according to the spatial distance among them. For this, we first find the histogram of each region r_k and then calculate the saliency s_k of r_k as

$$s_k = \sum_{(i \neq k)} n_i \cdot D_{lab}(r_k, r_i) \cdot e^{-E(r_k, r_i)/\sigma_s^2}, \qquad (1)$$

where n_i is the number of pixels inside r_i and $D_{lab}(r_k, r_i)$ is the color distance between regions r_k and r_i in LAB color space, $E(r_k, r_i)$ is the Euclidean spatial distance between centers of r_k and r_i, and σ_s controls the strength of spatial weighting. $\sigma_s = \frac{\sum_{j=1}^{N} dp_j}{N}$ is the average of differences between pixel pairs of frame F_t, where dp_j is the average of absolute intensity differences between pixel p_j and its four neighbors, and N is the number of pixels in F_t. We calculate the number of regions K using the super-pixel segmentation, which groups pixels of B_t into regions with similar values. With $\{s_k\}$, each pixel p_l of B_t has a saliency value. To reduce complexity, we apply a saliency thresholding of s_k to get the binary mask b_l for B_t as

$$b_l = \begin{cases} 1 & : s_k(p_l) > t_s, \\ 0 & : otherwise \end{cases} \quad t_s = \frac{\sum_k (n_k \cdot s_k)}{N_B}, \qquad (2)$$

where n_k is the number of pixels in r_k and $N_B = \sum_k n_k$ is the number of pixels in the BB B_t.

We then determine whether the target object inside B_t has drifted from its expected position depending on $0 < \alpha_s \leq 1$, the ratio of the binary salient pixels inside $\{b_l\}$, defined as $\alpha_s = \frac{N_s}{N_B}$, where N_s is the number of binary salient pixels (i.e., pixels p_l with $b_l = 1$) in B_t and N_B is the total number of pixels in all K regions (or the number of pixels in B_t). If α_s is smaller than c_1 or within the range (c_2, c_3), the target saliency is low and hence a drift starts to occur in frame F_t,

$$(\alpha_s < c_1) \vee ((\alpha_s > c_2) \wedge (\alpha_s < c_3)). \qquad (3)$$

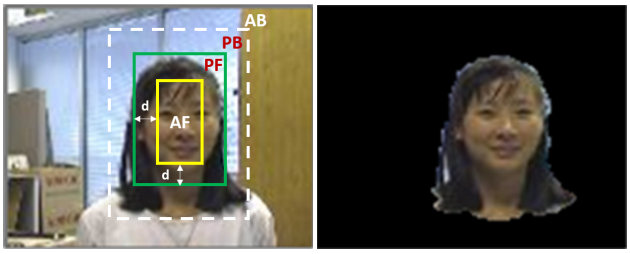
c_1, c_2, and c_3 are constants. A $\alpha_s > c_3$ indicates the mask inside B_t is highly salient (thus no drift assumed); a $\alpha_s < c_1$ indicates the mask is not salient (thus drift assumed); the uncertainty is when α_s is in the ranges (c_1, c_2) and (c_2, c_3). We have observed that once a BB *starts* to drift its saliency is not high but not low, e.g., $\alpha_s \in (c_1, c_2)$; but once a BB is in the process to significantly drift a good portion of it contains the background and α_s is medium high, e.g., $\alpha_s \in (c_2, c_3)$.

2.2. Drift Correction Using Seeded Segmentation

To correct drift, we relocate B_t, the output of the base tracker, based on the boundaries of the object segmented inside B_t; for this, we require an object segmentation method. Recent region proposal methods, such as [16], are powerful but output thousands of BBs as object candidates and the major challenge is to select a single BB representing the target for the sake of drift correction. G-Cut segmentation [11] is more suitable because it outputs a mask (not a BB) and has appealing results due to its ability to achieve a global solution [17]. It however, requires manual seed selection. The seeds in interactive segmentation [11, 17] should represent both object and background as hard constraints through user interaction. Interactive seed selection has been shown to improve the tracking quality [18]. However, interactivity is impractical for automated tracking applications. We propose to automatically select seeds for G-Cut using a two-layer filter: SIFT interest points and non-background pixels inside B_t.

SIFT finds distinctive interest points that are invariant to location, scale and rotation, and robust to affine transformations and illumination changes [19]. Among the SIFT points, there exist interest points that are more likely to belong to the background (not the target) which may mislead the segmentation. Accordingly, we propose a two-layered filter to reduce false alarms in SIFT interest points so that they are more likely to belong to the object. The first layer uses the already generated binary saliency map $\{b_l\}$ to filter out all points outside the saliency map of the target object. We select only those SIFT interest points that intersect with the most salient pixels of the binarized saliency map b_l. Meaning those pixels $p_l \in B_t$ with $b_l = 1$. The second layer filters out the interest points that belong to the background. We define the background as illustrated in Figure 1. Given the base tracker BB (green), we divide the frame into four regions: absolutely foreground (AF), probably foreground (PF), probably background (PB), and absolutely background (AB). The AF region belongs to the object as a hard constraint and is centered inside B_t with a distance d, defined in (4), from the boundaries of B_t. The PB is a margin to handle segmentation of irregular object parts outside the BB. Seeded segmentation expands from AF through PF (and possibly PB) regions until it reaches BB boundaries. We consider PB and AB that are outside the BB as background and thus exclude SIFT interest points located in PB and AB.

Seeded segmentation is sensitive to seed quantity and placement. It is important to select seeds that have a low probability of false alarm. As a consequence, our seeds selection in region AF avoids boundaries of BB and places the



(a) Seed masks. (b) Segmentation.

Fig. 1: Seed masks for object segmentation.

seeds starting from the center of the output BB of a tracker. Low-intensity (or dark) objects surrounded with a dark background are a challenge for segmentation and we thus select less seeds for such object to decrease false alarms. To this end, we use the average intensities μ_o of AF region and μ_b of its immediate neighbor pixels (e.g., in a radius of 10 pixels). The idea is to determine the appropriate AF region inside the BB centered at a distance d from its boundaries

$$d = \begin{cases} (1 - 0.1\mu_o) \cdot \upsilon & : (\mu_o < c_o) \wedge (|\mu_o - \mu_b| < c_b) \\ (1 - 2\mu_o + \mu_o^2) \cdot \upsilon & : otherwise, \end{cases} \tag{4}$$

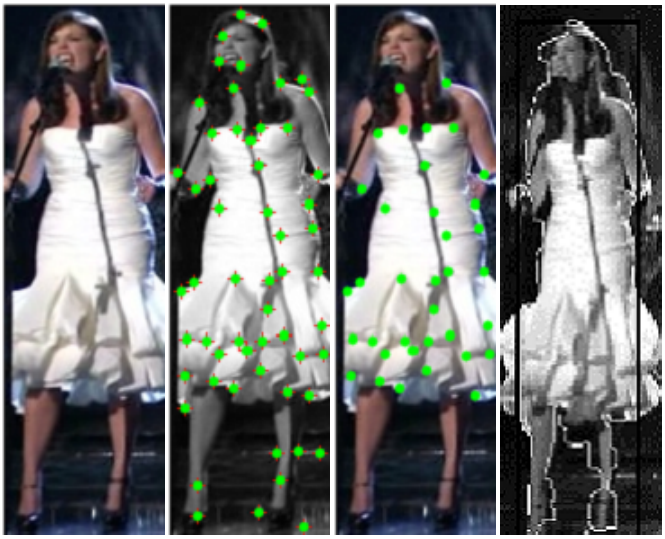
where $\upsilon = min(W, H)/2$ with W and H as the width and height of the BB. Then d is upper bounded to $\upsilon - 1$, which is the maximum distance to move from any BB boundary to reach the center of B_t. We select $c_o = 0.35$ to represent dark objects and $c_b = 0.03$ for the contrast between the object and its surrounded background. We assume that dark objects on low intensity background are to assign a smaller AF region (i.e., less seeds) for more accurate segmentation. This is because a smaller AF allows the segmentation, not the seeds, to decide what are the object pixels inside the BB. In (4), for high μ_o (bright BB), small d (i.e., more seeds) is estimated because the object boundaries are more distinguishable. For stability, d is determined one time per video sequence based on B_1, and not every B_t.

The output of our two-layer filter is thus seeds S (pixels p_l in B_t) that are SIFT interest points filtered by the binary saliency map and at the same time fall inside the AF region,

$$S = \{p_l \in (SIFT \wedge AF) \wedge (b_l = 1)\}. \tag{5}$$

Figure 2 illustrates this filtering and the segmentation result.

Object segmentation initialized with seeds S produces the object mask M that represents the target object. Due to various tracking and segmentation challenges, the object mask M from G-Cut is likely to include noisy blobs. Thus we process M as follows: we use the flood filling to fill small holes in M through connected component algorithm; we calculate the contour length of each region inside M; we select the region with the largest contour as the final mask C and disregard all other small regions. The BB B_t is then updated by relocating it around the center of C.



(a) Input BB. (b) SIFT points. (c) Filtered SIFT. (d) Segmentation.

Fig. 2: Interest points filtering and segmentation output.

3. RESULTS AND ANALYSIS

3.1. Drift Detection and Correction to Base Trackers

For object segmentation, we use the seeded G-Cut method; we experimented with recent region proposal methods [16] but those output thousands of BBs as object candidates to represent the target. For G-Cut, we set the parameters $\lambda = 218$ (weighting to control both over and under segmentation) and $\sigma = 10$ (image noise). For (3) to measure saliency of B_t, we experimentally selected $c_1 = 0.375$, $c_2 = 0.625$, and $c_3 = 0.925$ for scale-adaptive trackers (such as DSST [20], SAMF [21], and STAPLE [22]) and $c_1 = 0.2$, $c_2 = 0.6$, and $c_3 = 0.9$ for not scale-adaptive trackers (such as KCF [23] and STRUCK [24]). Scale-adaptive trackers adapt the estimated BB to the target size across the video sequence, while not scale-adaptive ones output a fixed size BB. For testing we use the publicly available OTB dataset of 100 sequences [1] that covers 11 different tracking challenges. We run each experiment three times on each sequence so to obtain fair statistics. For objective evaluation, we use overlap ratio AOR, center location error CLE, and failure rate FR.

We first tested our approach on five trackers, STRUCK[24], KCF[23], SAMF[21], DSST[20], and STAPLE[22], that show different performance in terms of tracking accuracy and speed [1, 2, 3]. Table 1 summarizes the average improvement in AOR, CLE, and FR, using the proposed method (labeled -SegTrack)), for the five trackers relative to their corresponding original (base) tracking quality over all test sequences. As can be seen, our approach improves the quality of all base trackers. Specifically, considering FR measure as an indicator for drift of a tracker, we can see major improvement (e.g.,

34% to SAMF tracker). We have examined the FR over all frames of each test video sequences and the proposed method either gives better or similar quality; not lower.

Table 1: Average improvement of the proposed drift detection and correction method. Better quality shown in bold.

Trackers	Overlap Ratio (AOR)	Center Location Error (CLE)	Failure Rate (FR)
STAPLE	0.586	31.3	0.148
STAPLE-SegTrack	**0.613**	**25.4**	**0.108**
Improvement	4.51%	18.72%	26.98%
SAMF	0.559	35.04	0.15
SAMF-SegTrack	**0.579**	**22.7**	**0.098**
Improvement	3.49%	35.03%	34.47%
DSST	0.528	48.3	0.205
DSST-SegTrack	**0.543**	**40.3**	**0.175**
Improvement	2.97%	16.52%	14.61%
KCF	0.481	44.3	0.199
KCF-SegTrack	**0.500**	**35.6**	**0.146**
Improvement	3.96%	19.66%	26.92%
STRUCK	0.465	48.9	0.206
STRUCK-SegTrack	**0.477**	**41.9**	**0.164**
Improvement	2.44%	14.27%	20.32%

With our multi-layer adaptive seeds selection, a modified tracker using our approach generates BBs that are better-placed (in terms of objective measure AOR and FR) than the BBs of the base tracker. This is because we restrict object seeds to be inside the original output BB and adapt the seeds to the BB itself. Experiments with high-performing deep-learning trackers such as MD-Net [25] or those with much lower quality such as LOT [26] show that in all cases, our approach increased the tracking quality of the base tracker even if little (e.g., 2% in FR); not FR, CLE, nor AOR were lower.

Subjectively, Figure 3 shows how the proposed method (continuous boxes) applied to all trackers reduces the tracking drift and successfully tracks the target object while the base trackers (dashed boxes) drift away from the target object.

3.2. Comparison to Segmentation-Based Trackers

We compared our method to the segmentation-based tracker (JOTS) [12], which requires the user to provide seeds and segmentation mask in the first frame. For fair comparison with our automated segmentation-based tracking on the publicly available OTB benchmark [1], we used our seeded segmentation to replace the required user interaction in JOTS, i.e., seeds and segmentation mask in F_1. Table 2 shows that the proposed method applied to KCF tracker (called KCF-SegTrack) well outperforms JOTS method for videos of the OTB benchmark [1]. Comparing Tables 1 and 2, we see that all our post-tracking modified trackers well outperform JOTS. Different than segmentation-based trackers (see review in section 1), we do not apply segmentation at each frame but only when drift is detected; this reduces both computations and dependency on segmentation.

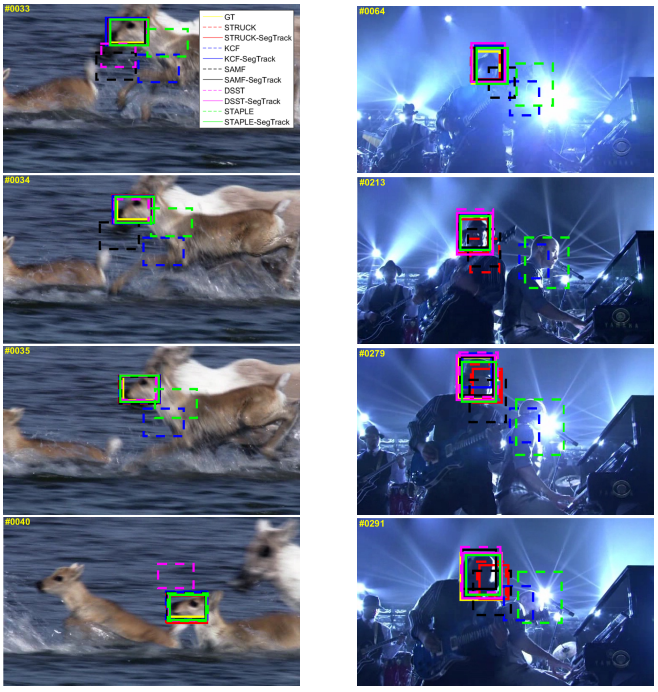


Fig. 3: Subjective improvement by the proposed method (solid boxes; best viewed in color).

Method	AOR	CLE	FR
KCF-SegTrack	**0.500**	**35.6**	**0.146**
JOTS	0.2943	54.06	0.2478

Table 2: Proposed SegTrack (applied to KCF, cf. table 1) versus JOTS segmentation-based tracker.

4. CONCLUSION

A tracker may drift away from an object due to different challenges. In this paper, we modeled drift detection by examining the output BB of a tracker, at each frame, aiming to correct its drift as needed. We proposed an approach for drift detection, using saliency features of the target contained in the output BB of a base tracker, integrated with a drift correction mechanism through seeded segmentation of the output BB. Our automated seeds selection used SIFT, saliency, and non-background pixels. Our approach is tracker-independent, i.e., can be applied to any tracking algorithm so to update its output BB at each frame once drift is detected. We tested our approach on different-performing base trackers. Results showed how our approach well reduces the tracking drifts in all tested trackers; for example, the failure rate of STAPLE tracker dropped by 26% and that of SAMF tracker by 34%. We also showed that the proposed approach significantly outperforms the segmentation-based tracker JOTS; our method did not apply segmentation at each frame but only at drift; this made it more robust to segmentation inaccuracies.

5. REFERENCES

[1] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Portland, OR, USA, June 2013, pp. 2411–2418.

[2] M. Kristan et al., "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 38, pp. 2137–2155, 2016.

[3] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *Proc. IEEE Int. Conf. Image Processing (ICIP)*, vol. 25, pp. 1261–1274, 2016.

[4] L. Čehovin, A. Lukežič, A. Leonardis, and M. Kristan, "Beyond standard benchmarks: Parameterizing performance evaluation in visual object tracking," in *Proc. IEEE Int. Conf. Computer Vision*, Oct. 2017.

[5] H. Xiaowei, L. Zheng, D. Yingkui, and G. Yuan, "A new template tracking algorithms based on active drift correction," in *5th Int. Conf. on Instrumentation and Measurement, Computer, Communication and Control*, Qinhuangdao, China, Sept. 2015, pp. 1579–1584, IEEE.

[6] X. Chen, K. Ren, and Y. Hou, "Multi-stage target tracking with drift correction and position prediction," *J. of Physics: Conf. Ser.*, vol. 1004, pp. 1–10, 2018.

[7] K. Ratnayake and M.A. Amer, "Drift detection using SVM in structured object tracking," in *16th Int. Conf. on Image Analysis and Recognition (ICIAR)*, Waterloo, Canada, Aug. 2019, pp. 67–76.

[8] K.E. Papoutsakis and A. Argyros, "Integrating tracking with fine object segmentation," *Image Vis. Comput.*, vol. 31, pp. 771–785, 2013.

[9] K.E. Papoutsakis and A. Argyros, "Object tracking and segmentation in a closed loop," in *Int. Conf. Adv. Visual Comput. (ISCV)*, Las Vegas, NV, USA, June 2010, pp. 405–416, Springer-Verlag.

[10] Y. Wang, G. Jiang, and C. Jiang, "Mean shift tracking with graph cuts based image segmentation," in *Int. Congress Image Signal Processing (CISP)*, Chongqing, Sichuan, China, Oct. 2012, pp. 675–679.

[11] Y. Boykov and M-P Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in ND images," in *Proc. IEEE Int. Conf. Computer Vision*, July 2001, pp. 105–112.

[12] L. Wen et al., "JOTS: Joint online tracking and segmentation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June. 2015, pp. 2226–2234.

[13] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. European Conf. Computer Vision*, 2014, pp. 391–405.

[14] C. Li, C. Xu, C. Gui, and M. D. Fox, "A saliency map based on sampling an image into random rectangular regions of interest," *J. Pattern Recognition*, vol. 45, pp. 3114–3124, 2012.

[15] M. Cheng, G. Zhang, N. J. Mitra, X. Huang, and S. Hu, "Global contrast based salient region detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2011, pp. 409–416.

[16] J. Wang, K. Chen, S. Yang, C.C. Loy, and D. Lin, "Region proposal by guided anchoring," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Long Beach, CA, June 2019.

[17] M. Schmidt and K. Alahari, "Generalized fast approximate energy minimization via graph cuts: Alpha-expansion beta-shrink moves," *27th Conf. Uncertainty Artificial Intelligence (UAI)*, 2011.

[18] B. Vasileios, F. Schubert, N. Navab, and S. Ilic, "Segmentation based particle filtering for real-time 2D object tracking," in *Proc. European Conf. Computer Vision*, Florence, Italy, 2012, pp. 842–855, Springer-Verlag.

[19] H.R. Kher and V.K. Thakar, "Scale invariant feature transform based image matching and registration," in *Proc. IEEE Int. Conf. Signal Image Processing (ICSIP)*, 2014, pp. 50–55.

[20] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conf.*, Nottingham, Sept. 1-5 2014, pp. 1838–1845, BMVA Press.

[21] L. Yang and Z. Jianke, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proc. European Conf. Computer Vision*, Cham, Sept. 1-5 2014, pp. 254–265, Springer.

[22] L. Bertinetto et al., "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Oct. 2016, pp. 1401–1409.

[23] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, pp. 583–596, 2015.

[24] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Computer Vision*, Barcelona, Nov. 2011, pp. 263–270.

[25] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 4293–4302.

[26] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *Int. J. Computer Vision*, vol. 111, pp. 213–228, 2015.