A Compressive Tracking Method Based on Gaussian Differential Graph and Weighted Cosine Similarity Metric

Wang Minmin, Sun Shengli, and Li Yejin

***Abstract*—This letter presents an extended compressive tracking algorithm to increase the stabilization and robustness in scale variation, rotation, and illumination variation. The features are extracted from the Gaussian differential graphs and taken as input signals of compressive sensing. In order to reduce the computational cost, the operating area is narrowed down to the region of interest. The algorithm utilizes a naïve Bayes classifier to get *N* candidate target locations with *N* highest classifying scores. Their weighted multiframe cosine similarities with the ground truth object from the initial frame and the most similar object in some adjacent frame are calculated to find the target location in current frame. Experimental results demonstrate the superiority of the proposed algorithm over some state-of-the-art tracking algorithms. It also can efficiently meet the needs of real-time tracking.**

***Index Terms*—Compressive sensing (CS), cosine similarity, Gaussian differential graph, visual tracking.**

I. Introduction

COMPRESSIVE sensing (CS)[1], [2] states that a high-dimensional signal which is sparse on a given basis can be randomly sampled at a frequency which is much less than the Nyquist frequency. And the original signal can be reconstructed from these samples by a nonlinear reconstruction algorithm. Therefore, if the feature space is sufficiently sparse, CS can be imported to target tracking as well. As a first step toward this, Mei and Ling [3] apply the sparse representation method into target detection. The sparse representation of a candidate object is achieved via object projection to a template space consisting of target templates and trivial templates. Mei *et al.* [4] improve the tracking method in [3] by using an error bound derived from the least squares computation. Zhang *et al.* [5] propose a tracking algorithm with features from the compressed domain. The high-dimension Haar-like feature space is projected to a low-dimension compressed space by a sparse Gaussian measurement matrix. A naïve Bayes classifier is adopted to obtain the tracking result in current frame. Zeng *et al.* [6] train the classifier with sparse Harr-like features weighted by a spatial kernel function. Zhang *et al.* then extend the work of [5] and propose a fast compressive tracking (FCT) algorithm [7]. By combining the compressed feature classification with a coarse-to-fine detection strategy, the computational cost of the FCT is reduced dramatically. However, working only with data from the previous frame results in an error accumulation problem in the FCT. Song [8] develops the image feature by combing the multiscale appearance information with spatial layout information. With an online feature selection technique based on entropy energy, the algorithm can achieve good performance. Kong *et al.* [9] propose a compressive tracking algorithm based on a Gaussian differential graph. Features are extracted from the Gaussian differential graphs and regarded as input signals of the tracking process. Experimental results show the robustness of the algorithm in scaling, texture change, and illumination variance. But this algorithm is hindered by its huge computational cost as it calculates the whole image's Gaussian differential graphs. And error accumulation is still an issue in [8] and [9]. Jenkins *et al.* [10], [11] improve the FCT by ultilizing a weighted cosine similarity metric with the initial target and the most similar object in some adjacent frame. The candidate object with the maximal cosine similarity is taken as the current tracking result. Hence, error accumulation can be alleviated to some extent.

Inspired by the work in [9] and [10], an extended compressive tracking algorithm is proposed in this letter. In view of the robustness of the Gaussian differential graph to illumination and texture change, features are extracted from the Gaussian differential graphs. And the object with the highest weighted cosine similarity is selected as the new target location, which is effective to alleviate the error accumulation.

II. Proposed Algorithm

Our proposed algorithm is an extension of the FCT and termed as compressive tracker based on the Gaussian differential graph and cosine similarity (GDCS). The main operation of the GDCS is shown in Fig. 1. And a more detailed description of the GDCS is as follows.

A. Fast Compressive Tracking

FCT algorithm [7] extracts features from the compressed domain rather than directly from the original image. And the tracking process is cast as a binary classification problem.

Manuscript received July 5, 2016; accepted August 21, 2016. Date of publication August 25, 2016; date of current version February 28, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ba-Tuong Vo.

The authors are with the Key Laboratory of Infrared System Detection and Imaging Technology, Chinese Academy of Sciences, Shanghai 200083, China, and also with Shanghai Institute of Technical Physics of the Chinese Academy of Science, Shanghai 200083, China (e-mail: wangminmindata@163.com; palm_sun@163.com; biangjin@hotmail.com).

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.2016.2603170

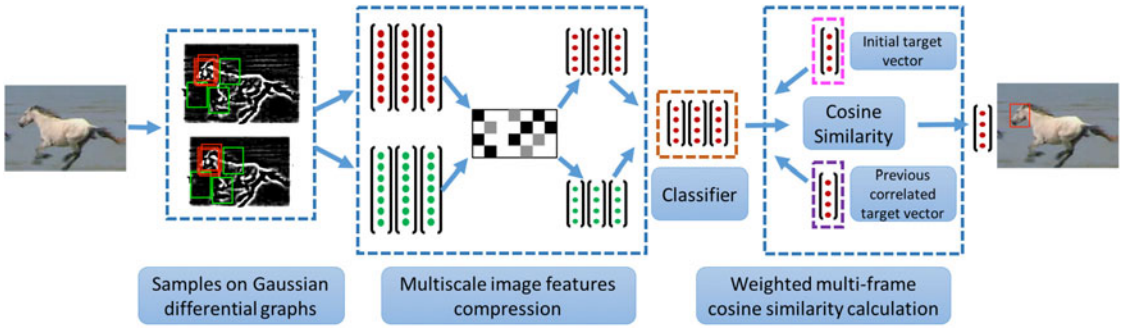

Fig. 1. Main operation of the proposed algorithm.

1) Feature Extraction: The FCT defines the multiscale representation of each sample $z \in \mathbb{R}^{w\times h}$ as convolutions of z with a set of multiscale rectangle filters $\{F_{1,1}, \ldots F_{w,h}\}$

$$F_{w_f,h_f}(x,y) = \frac{1}{w_f h_f} \times \begin{cases} 1 & 1 \le x \le w_f, 1 \le y \le h_f \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where w and h define the width and height of the samples; w_f and h_f are the width and height of the rectangle filter, respectively. After converted to column vectors in $\mathbb{R}^{wh}$, all filtered samples are concatenated as a high-dimensional feature vector $x \in \mathbb{R}^n$, where $n = (wh)^2$.

Then, the feature $x \in \mathbb{R}^n$ is projected to a lower dimensional feature $v \in \mathbb{R}^m$ by a sparse measurement matrix $R \in \mathbb{R}^{m\times n} (m \ll n)$ as

$$v = Rx. \tag{2}$$

R in (2) satisfies the Johnson–Lindenstrauss lemma[12]. And the elements of R are defined as

$$r_{\text{ij}} = \sqrt{s} \times \begin{cases} 1 & \text{with probability } \dfrac{1}{2s} \\ 0 & \text{with probability } 1 - \dfrac{1}{s} \\ -1 & \text{with probability } \dfrac{1}{2s}. \end{cases} \tag{3}$$

Since $n = 10^6 - 10^{10}$, s can be set as $s = O(n) = n/(a\log_{10}^n) = n/(10a) \sim n/(6a)$, where $a = 0.4$. The number of nonzero elements for each row of R is 4 at most.

2) Detection: The FCT assumes that all features are independently distributed. A naïve Bayes classifier [13] $H(v)$ is utilized to realize the classification

$$\begin{aligned} H(v) &= \log\left(\frac{\prod_{i=1}^m p(v_i \mid y=1)p(y=1)}{\prod_{i=1}^m p(v_i \mid y=0)p(y=0)}\right) \\ &= \sum_{i=1}^m \log\left(\frac{p(v_i \mid y=1)}{p(v_i \mid y=0)}\right) \end{aligned} \tag{4}$$

where $p(y=1) = p(y=0)$ is assumed, and $y \in \{0,1\}$ is a binary sample label. And the conditional distributions $p(v_i \mid y=1)$ and $p(v_i \mid y=0)$ in the classifier $H(v)$ are assumed to be Gaussian distributed as

$$p(v_i \mid y=1) \sim N(\mu_i^1, \sigma_i^1) \tag{5}$$

$$p(v_i \mid y=0) \sim N(\mu_i^0, \sigma_i^0) \tag{6}$$

Fig. 2. Coarse-to-fine search strategy. Left: object center location in the $(t-1)$th frame denoted by the yellow point. Middle: coarsely sample in a large region with a radius γ_c. Right: finely sample in a small region with a radius γ_f. The white point indicates the object center location in the tth frame.

where $\mu_i^1(\mu_i^0)$ and $\sigma_i^1(\sigma_i^0)$ are mean and standard deviation of the positive (negative) samples, respectively.

Then the candidate object with the maximal classifier score is chosen as the current tracking result. And the parameters in (5) can be updated as follows:

$$\mu_i^1 \leftarrow \lambda\mu_i^1 + (1-\lambda)\mu^1 \tag{7}$$

$$\sigma_i^1 \leftarrow \sqrt{\lambda(\sigma_i^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2} \tag{8}$$

where $\lambda(\lambda > 0)$ is a learning parameter, and a smaller λ indicates faster updating speed. The parameters μ_i^0 and σ_i^0 in (6) can be updated in a similar way.

The FCT algorithm adopts a coarse-to-fine detection strategy (see Fig. 2) to further reduce the computational cost. First, the FCT coarsely samples in a large region and narrows down the probable location of the target to a smaller region. Then, it finely samples in this smaller region to determine the exact target location.

B. Gaussian Differential Graph

Before tracking, two Gaussian differential graphs should be constructed. The main process is similar to the constructor of the DoG scale-space [14]. For a given image, the difference of two Gaussians can be defined as

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) \otimes I(x,y) \tag{9}$$

where $I(x,y)$ represents the original image and $G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ is a two-dimensional Gaussian kernel with a scale factor σ. The parameter k represents the multiple relationship between two scale factors of the Gaussians. In this letter, $O_{\max}$ octaves are constructed first. And each octave is divided into S sublevels. The scale factor $\sigma(k,i)$ of the ith sublevel in the kth

TABLE I
SCALE FACTOR

Sublevel	1	2	3	4	5
Octave 1	0.98	1.13	1.29	1.49	1.71
Octave 2	1.96	2.25	2.59	2.97	3.41

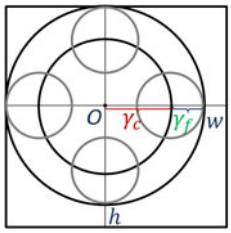

Fig. 3. Illustration of the operating region. O is the top left point of the tracker bounding box in the previous frame. γ_c is the large radius for coarsely sampling. γ_f is the small radius for finely sampling. w and h are the width and height of the sample, respectively.

octave can be written as

$$\sigma(k,i) = \sigma_0 2^{(k-1)+(i-1)/S}, k \in [1,...,O_{\max}], i \in [1,...,S] \tag{10}$$

where σ_0 is the initial scale factor. In this letter, $O_{\max} = 2$, $S = 5$, and $\sigma_0 = 0.98$. Table I presents the scale factors for all sublevels. Our octaves are constructed by Gaussian low-pass filter with standard deviations (scale factors) from Table I. And the size of the filter template $W \times W$ is set as (11) for reasons of real time and robustness

$$W \times W = \begin{cases} 9 \times 9 & \sigma < 1.6 \\ 11 \times 11 & \text{otherwise.} \end{cases} \tag{11}$$

For each octave, the $(i+1)$th sublevel is constructed by Gaussian filtering on the ith sublevel with the $(i+1)$th template. And the first sublevel in the first octave is obtained by filtering on the input image, while the one in the second octave is constructed by filtering on the third from the bottom in the first octave. By subtracting the last sublevel from the first sublevel in each octave, two Gaussian differential graphs can be constructed respectively.

Then features are extracted from the above Gaussian differential graphs. For a particular location, a combination of its features from these two Gaussian differential graphs are taken as its final feature before projection. In this letter, simple addition is executed.

To reduce the computational cost of the Gaussian differential graph, a window-based method that restricts the operating region to a particular region rather than the whole image is proposed. But this region should still be bigger than the sampling area. Thus, the operating region is determined by the coarse-sampling radius γ_c, fine-sampling radius γ_f, as well as the width w and height h of the sample. As shown in Fig. 3, the width of the operating window winw should satisfy the inequality (12) as

$$\text{winw} \geq (\gamma_f + \gamma_c) + (\gamma_c + \gamma_f + w). \tag{12}$$

And its height winh should satisfy that

$$\text{winh} \geq (\gamma_f + \gamma_c) + (\gamma_c + \gamma_f + h) \tag{13}$$

where O is the top left point of the tracker bounding box in the previous frame.

C. Weighted Cosine Similarity Metric

Tracking algorithm in [10] also is an extension of the FCT. It selects N candidate locations with the N highest classifying scores, and calculates their weighted multiframe cosine similarities afterward. Then the location with the maximal weighted cosine similarity is taken as the tracking result in current frame.

The cosine similarity between the ground truth object from the initial frame G and some candidate object in the current frame T is defined as

$$S_{g,t} = \frac{\sum_{i=1}^{n} g_i \times t_i}{\sqrt{\sum_{i=1}^{n} g_i^2}\sqrt{\sum_{i=1}^{n} t_i^2}} \tag{14}$$

where $g = \text{vec}(G)$ and $t = \text{vec}(T)$.

Analogously, the cosine similarity between the most similar object from some adjacent frame P and some candidate object in the current frame T is written as

$$S_{p,t} = \frac{\sum_{i=1}^{n} p_i \times t_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} t_i^2}} \tag{15}$$

where $p = \text{vec}(P)$ and $t = \text{vec}(T)$. When $S_{p,t}$ is bigger than a given threshold τ, the most similar object from the adjacent frame should be updated.

Thus, the final weighted multiframe cosine similarity can be defined as

$$S = \alpha S_{g,t} + (1-\alpha) S_{p,t} \tag{16}$$

where α is a manually selected weight and $0 < \alpha < 1$.

III. EXPERIMENTS

A. Experimental Setup

The proposed algorithm is evaluated with 7 state-of-the-art tracking methods on 11 challenging sequences which are publicly available in [15]. All algorithms are implemented in MATLAB, which runs on an Intel(R) Core(TM) i7-4770 3.40 GHz CPU with 4G RAM. The number of candidate locations is set as $N = 20$. The weight of the cosine similarity metric is set as $\alpha = 0.8$ and the threshold τ for updating the most similar object from the adjacent frame is set as $\tau = 0.85$.

B. Evaluation Criteria

Tracking performance is evaluated by the precision and success rate which are used in the benchmark paper [15]. Center location error (CLE) is a widely used evaluation metric of precision, defined as the Euclidean distance between center locations of the manually labeled ground truth bounding box and the tracker bounding box. To be more objective, the precision plot [16] is also adopted. It shows the percentage of frames whose CLE is smaller than the given threshold t_0. In this letter, t_0 varies from 0 to 50. Another evolution metric is the success rate which is obtained by calculating the overlap score (OS) of the bounding boxes. It is defined as

$$\text{OS} = \frac{|G_{\text{rec}} \cap T_{\text{rec}}|}{|G_{\text{rec}} \cup T_{\text{rec}}|} \tag{17}$$

TABLE II
AVERAGE CENTER LOCATION ERROR AND AVERAGE FRAME PER SECOND (FPS)

	GDCS	CT [5]	FCT [7]	CSK [17]	CXT [18]	CPF [19]	Struck [20]	CNT [21]
Fish	*6*	11	12	41	*6*	39	3	35
Mhyang	3	15	15	4	4	13	3	2
Man	2	45	*4*	2	2	62	2	2
Boy	3	40	7	20	2	5	4	45
Singer2	59	32	22	185	205	53	174	177
Diving	*47*	63	61	82	66	18	53	82
David3	81	63	105	56	222	20	106	10
Walking	5	218	5	7	211	4	5	2
Biker	96	58	34	73	50	49	26	75
Freeman4	19	93	76	79	123	66	*49*	66
Soccer	75	101	136	71	45	61	71	124
AveCLE	*36*	67	43	56	85	35	45	56
AveFPS	50.8	61.2	*145.2*	568.0	18.8	118.4	10.5	0.5

Underline fonts indicate the best results while the *italic* fonts indicate the second best ones.

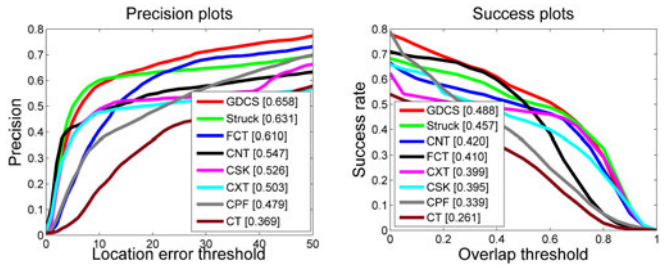

Fig. 4. Precision plots and success plots of all trackers.

where G_{rec} is the ground truth bounding box; T_{rec} is the tracker bounding box. Tracking result with OS larger than the threshold t_1 is considered as a success. Similarly, the success plot is adopted. It shows the percentage of frames whose OS is larger than t_1. t_1 varies from 0 to 1. And the area under curve (AUC) is used to rank the tracking methods [15].

C. Experimental Results

Table II presents the average CLE and frames per second of each tracking method. For most of the sequences, the GDCS outperforms the CT and FCT. It is predictable that CSK has the highest running speed among all trackers. Although the average frame rate of the GDCS is slower than the CT, FCT, CSK, and CPF, it still reaches 50.8 frames per second, which can meet the needs of real-time tracking.

However, once the tracker loses the target, the tracker location can be unpredictable. Therefore, average CLE cannot accurately describe the overall tracking performance. Fig. 4 shows the precision plots and success plots of all trackers. For precision plots, the ranking list is produced according to the tracking results at a threshold of 20. And for success plots, AUC scores are used to rank the trackers. It is obvious that the GDCS achieves the best tracking performance throughout these image sequences.

To further analyze the algorithmic characteristics, the sequences are divided into several subsets according to their own attributes. The main attributes include illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution. Fig. 5 shows the success plots for some attribute subsets. For image sequences with in-plane rotation, scale variation, out-of-plane rotation, illumination variation, and background clutter, the GDCS ranks first. Although the GDCS ranks second for the attributes of motion blur, deformation and fast motion, it still can achieve excellent performance.

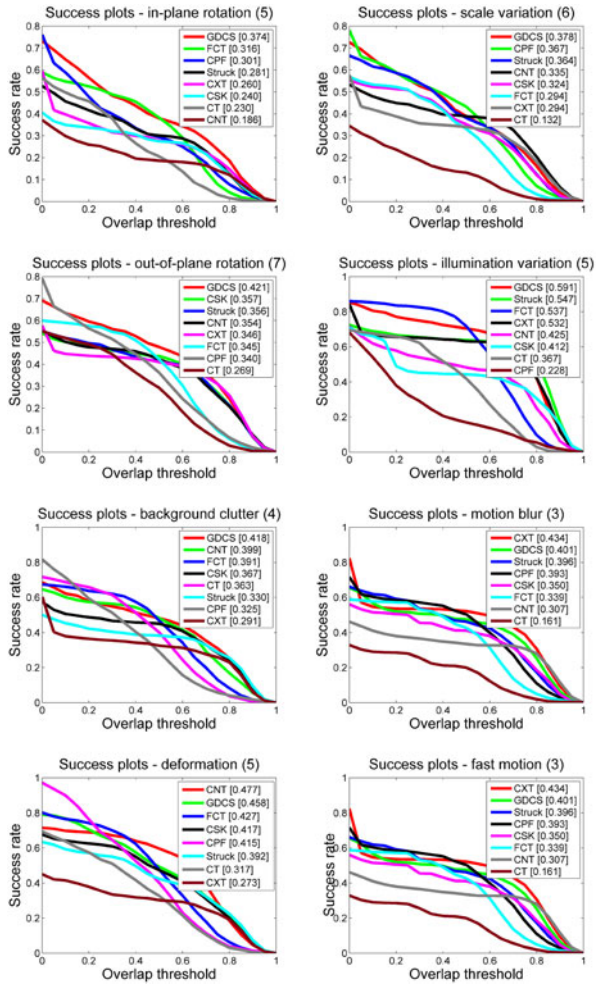

Fig. 5. Success plots for several attribute subsets.

IV. CONCLUSION

In this letter, an extended compressive tracking algorithm with increased robustness in scale variation, rotation, deformation, and illumination variation is proposed. The algorithm extracts features from Gaussian differential graphs of the region-of-interest and utilizes a weighted multiframe cosine similarity metric to alleviate error accumulation. As the weight α and updating threshold τ for weighted cosine similarity metric are manually set in this letter and the optimal values of these parameters are different from sequence to sequence, there is still work to be carried out to make the tracker automatically choose the parameters α and τ for a particular image sequence.

REFERENCES

[1] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[2] E. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Trans. Inf. Theory,* vol. 52, no. 12, pp. 5406–5425, Dec. 2006.

[3] X. Mei and H. Ling, "Robust visual tracking using l1 minimization," in *Proc. 2009 IEEE 12th Int. Conf. Comput. Vision*, Sep. 2009, pp. 1436–1443.

[4] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, "Efficient minimum error bounded particle resampling l1 tracker with occlusion detection," *IEEE Trans. Image Process.*, vol. 22, no. 7, pp. 2661–2675, Jul. 2013.

[5] K. Zhang, L. Zhang, and M.-H. Yang, *Real-Time Compressive Tracking*. Berlin, Germany: Springer, 2012, pp. 864–877.

[6] F. Zeng, X. Liu, Z. Huang, and Y. Ji, "Kernel based multiple cue adaptive appearance model for robust real-time visual tracking," *IEEE Signal Process. Lett.*, vol. 20, no. 11, pp. 1094–109, Nov. 2013.

[7] K. Zhang, L. Zhang, and M.-H. Yang, "Fast compressive tracking," *IEEE Trans., Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 2002–2015, Oct. 2014.

[8] H. Song, "Robust visual tracking via online informative feature selection," *Electron. Lett.*, vol. 50, no. 25, pp. 1931–1933, 2014.

[9] J. Kong, M. Jiang, X.-W. Tang, Y.-N. Sun, K. Jiang, and G.-R. Wen, "Target tracking by compressive sensing based on gaussian differential graph," *J. Infrared Millimeter Waves*, vol. 34, no. 1, pp. 100–105, Feb. 2015.

[10] M. D. Jenkins, P. Barrie, T. Buggy, and G. Morison, "An extended real-time compressive tracking method using weighted multi-frame cosine similarity metric," in *Proc. 2014 6th Eur. Embedded Des. Educ. Res. Conf.*, Sep. 2014, pp. 147–151.

[11] M. D. Jenkins, P. Barrie, T. Buggy, and G. Morison, "Extended fast compressive tracking with weighted multi-frame template matching for fast motion tracking," *Pattern Recognit. Lett.*, vol. 69, pp. 82–87, 2016.

[12] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[13] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, pp. 841–848, 2002.

[14] S. Nilufar, N. Ray, and H. Zhang, "Object detection with dog scale-space: A multiple kernel learning approach," *IEEE Trans. Image Process.*, vol. 21, no. 8, pp. 3744–3756, Aug. 2012.

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

[16] B. Babenko, M. H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, *Exploiting the Circulant Structure of Tracking-by-Detection With Kernels*. Berlin, Germany: Springer, 2012, pp. 702–715.

[18] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *Proc. 2011 IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2011, pp. 1177–1184.

[19] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, *Color-Based Probabilistic Tracking*. Berlin, Germany: Springer, 2002, pp. 661–675.

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

[21] K. Zhang, Q. Liu, Y. Wu, and M. H. Yang, "Robust visual tracking via convolutional networks without training," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1779–1792, Apr. 2016.