Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1574054, 17 pages
https://doi.org/10.1155/2020/1574054

Research Article

Object Tracking with Multi-Classifier Fusion Based on Compressive Sensing and Multiple Instance Learning

Si Chen, Xiaoshun Lu, Xiaosen Chen, Min Chen, Jianghu Chen, Dahan Wang, and Shunzhi Zhu

School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China

Correspondence should be addressed to Shunzhi Zhu; szzhu@xmut.edu.cn

Received 14 September 2019; Revised 25 November 2019; Accepted 5 December 2019; Published 10 March 2020

Academic Editor: Mohammad D. Aliyu

Object tracking is a critical research in computer vision and has attracted significant attention over the past few years. However, the traditional object tracking algorithms often suffer from the object drifting problem due to various challenging factors in complex environments such as object occlusion and background clutter. This paper proposes a robust and effective object tracking algorithm, called MCM, which combines compressive sensing and online multiple instance learning in a multi-classifier fusion framework. In this framework, we integrate the different discriminative classifiers by learning the varied and compressed feature vectors based on different random projection matrices. And then an improved online multiple instance learning mechanism SMILE is adopted, which introduces the relative similarity to select and weight the instances in the positive bag. The experiments show that the proposed algorithm can improve the performance of object tracking on the challenging video sequences.

1. Introduction

Object tracking [1] is an important research direction in the field of computer vision and has a wide range of applications. For example, video surveillance [2], mobile robots [3, 4], and defense reconnaissance [5]. The object tracking methods continuously predict the location and the other information of the object for the subsequent frames in a video sequence, given the initial state in the first frame. At the same time, there are many challenges in object tracking, such as object deformation, out-of-plane rotation, and illumination variation. In the above scenes, the traditional tracking algorithms are easy to lead to tracking failure [6]. In this paper, we mainly focus on the online object tracking problem when the object location in the first frame of a video sequence is only given without both the other training samples and prior knowledge.

During the past several years, many scholars have proposed many object tracking algorithms, for example, CT (Compressive Tracking) [7], MIL (Multiple Instance Learning) tracker [6], MOSSE (Minimum Output Sum of Squared Error) filter [8], TLD (Tracking-Learning-Detection) tracker [9], Struck (Structured Output Tracking with Kernels) [10], CSK (Circulant Structure of Tracking-by-detection with Kernels) method [11], MD-Net (Multi-Domain Convolutional Neural Networks) tracker [12], and DLC (Discriminative Local Collaborative Representation) tracker [13]. These algorithms also have their own advantages and disadvantages. For example, the compressive sensing technology is introduced in the CT algorithm [7] for feature extraction. However, only a fixed random projection matrix is used to feature extraction, which easily loses discriminative features when the appearance of the object changes drastically. Although the MIL paradigm has been successfully used in the object tracking [6], there are still many problems to be solved. The traditional MIL-based object tracking algorithm [6] may be inaccurate in the complex scenes such as fast movements and occlusions because the model easily introduces the error information due to the unsuitable choice of the positive bag.

In this paper, we propose a novel object tracking algorithm with multi-classifier fusion based on compressive sensing and multiple instance learning, termed as MCM. In the multi-classifier fusion framework, the MCM algorithm

firstly generates the different feature vectors of instances with the different random projection matrices based on the compressive sensing technology. Then, an improved online multiple instance learning mechanism, i.e., SMILE (Relative Similarity Based Online Multiple Instance Learning), is adopted, which further selects and weights the instances in the positive bag by introducing the concept of relative similarity. Finally, by fusing the tracking results of multiple strong classifiers, the MCM algorithm can obtain the more discriminative classifier to effectively deal with the dramatic appearance changes in the complex environments.

The main contributions of our paper are summarized as follows:

(1) We introduce the multi-classifier fusion framework to object tracking, by utilizing multiple single classifiers to ensemble learning, so as to obtain better tracking results, benefited from the diversity of the multiple discriminative classifiers

(2) We can extract the different low-dimensional feature vectors by using multiple different random projection matrices based on the compressive sensing technology, in order to obtain the more discriminative classifiers in the multi-classifier fusion framework

(3) The MCM algorithm employs an improved SMILE mechanism to filter and weight the instances in the positive bag, so as to avoid the mistake information to be introduced in the tracking process

The rest of this paper is organized as follows. We first review the related work about object tracking in Section 2 and then describe the overview of the proposed algorithm in Section 3. Section 4 illustrates the experimental results tested on the challenging video sequences. The conclusion is made in Section 5.

2. Related Work

The object tracking methods [6–49] can be divided into offline methods [14, 15] and online methods [6–13, 16–49]. The offline object tracking methods [14, 15] use the object model built before tracking to track the object, and the learned model is not further updated during the tracking process. The shortcoming of the offline methods is that it is difficult to adapt to the drastic changes of the object appearance in the complex environment. On the other hand, the online tracking methods [16–21] can keep updating the object appearance model during the tracking process, so that the dynamic changes of the object appearance can be more robustly dealt with, and the object drifting problem can be avoided to some extent. Therefore, this paper mainly focuses on the online tracking methods. According to the different modelling patterns, the online tracking algorithms can be further divided into generative learning [8, 12, 17, 18, 22–25, 33–38, 41, 42, 44, 45] and discriminative learning [6, 7, 9–11, 16, 19–21, 26–32, 39, 40, 43, 46–49].

2.1. Generative Tracking. The online tracking algorithm based on generative learning usually establishes an object appearance model through the object representation technology and then searches for the region most similar to the object appearance model in a new image frame. For example, Havangi [18] performed the MCMC (Markov chain Monte Carlo) motion processing after resampling. However, the required number of probability transfers is large, and the convergence is difficult to achieve. In order to avoid the noise interference and the inaccurate observation value of the Jacobian matrix, Zhou et al. [44] proposed a robust Kalman filter (KF) algorithm with long short-term memory (LSTM) for an image-based visual servo control system. Mei and Ling [23] applied the sparse representation to the object tracking, which uses the reconstruction error of the sparse representation as the weight of the candidate object and selects the candidate object of the maximum weight as the tracking result. An adaptive image processing method [34] is suggested to determine the 3D position and the velocity of moving objects by using a distributed camera array. Marata et al. [35] proposed the Separate Monte Carlo Mean (SMC-MEAN), which is applied to an autonomous object tracking problem in both Gaussian and non-Gaussian scenarios. Nam et al. [32] proposed the convolutional neural networks (CNNs) algorithm to represent the appearance of the object, which constructs a tree structure based on multiple CNNs and maintains the reliability of the model by smoothly updating along the tree path. Lu et al. [33] presented a transform-aware attentive tracking framework, which uses a deep attentive network to directly predict the target states via spatial transform parameters. Pan et al. [37] proposed a robust object tracking framework, based on a canonical CF tracker. Specifically, an adaptive model update strategy is proposed to evaluate this model with a siamese network. Liu et al. [41] used the spatial feature and the temporal feature which are fused into the deep residual network model with the multi-scale feature vector, so that they can obtain the deep multi-scale spatiotemporal feature model. Zhou et al. [45] proposed a novel object tracking method with the fusion of the extended Kalman particle filter (EKPF) and the least squares support vector regression (LSSVR) to effectively improve the accuracy. Zhang et al. [42] put forward a multi-feature integration framework, including the gray features, histogram of gradient (HOG), color-naming (CN), and illumination invariant features (IIF), in order to overcome the problem of poor representation of single feature in a complex image sequence.

Although many online generative tracking methods have been proposed, there are still several issues to be solved. Firstly, the online appearance model is built which is largely dependent on a number of training samples, for example, deep learning-based trackers [32, 33, 37, 41]. If there are only a few training samples at the beginning of tracking, the appearance model is hard to be robust when the appearance of the object changes significantly in the process of tracking. Secondly, when many samples around the current object location are used to build an object model, the error background information is potentially introduced. Thirdly, the generative algorithms usually only use the object information, without considering the helpful background.

2.2. Discriminative Tracking. The online tracking algorithm based on discriminative learning learns a classifier to distinguish the object sample from the background. Collins et al. [26] embedded an online multi-feature selection mechanism in the mean shift tracking method, and the features with high discrimination are selected. This method easily suffers from the object drifting, when the object is similar to the background. Huang et al. [43] proposed the MD-JITS (multiple detection joint integrated track splitting) tracker, which is used to the multiple-detection multi-target tracking in cluttered environments. Grabner et al. [28] proposed a well-known online boosting framework for real-time object tracking, which uses only one positive sample in the object region and several negative samples from the background to update the classifier. However, the tracker may not keep very accurately tracking the object, and thus, some possibly mispositioned samples are easily introduced to update the appearance model.

To guard against selecting the incorrect samples to update the classifier, Grabner et al. [20] proposed an online semi-supervised boosting method, where the samples at the first frame are labelled and the samples from the other frames are unlabelled. Chen et al. [46] proposed a novel visual tracking algorithm via online semi-supervised co-boosting, and the algorithm has a good ability to recover from drifting. Babenko et al. [6] introduced the multiple instance learning (MIL) paradigm to online object tracking by using the positive and negative bags, where some samples approaching the object location are added to a positive bag. A large number of samples far from the object location are taken to form a negative bag. In order to solve the unselective treatment of instances in the positive bag during the MIL tracking process and the lack of prior knowledge of the object in instance modelling, Chen et al. [47] proposed an online multiple instance boosting algorithm with instance-level semi-supervised learning, termed SemiMILBoost, to achieve more robust object tracking. Zhang and Song [16] proposed an improved online weighted MIL tracker (WMIL), where the instances are weighted by the distance from the object location. Zhou et al. [48] improved the MIL tracking algorithm by optimizing a bag Fisher information function and integrating the co-training criterion. And, Zhou et al. [49] also proposed a multiple instance learning (MIL) tracking method based on a semi-supervised learning model with Fisher linear discriminant. However, in the above methods [6, 16, 47, 48, 49], the positive bag may contain some negative samples because the radius of the positive bag is difficult to be very accurately selected. Therefore, when all the instances in the positive bag are used to update the classifiers, the above algorithms might suffer from the drifting problem in the complex environments.

3. The MCM Algorithm

3.1. Overview. The MCM algorithm first uses the different random projection matrices to reduce the dimensions of instances in the positive and negative bags, respectively, thus obtaining the different compressed feature vectors. In the MCM algorithm, the SMILE mechanism is presented to improve the traditional online multiple instance learning by introducing the relative similarity metric. After the SMILE mechanism, the final strong classifier is obtained by the multi-classifier fusion framework. The whole flow of the MCM algorithm is shown in Figure 1.

During the tracking process, the original multi-scale image feature vectors (i.e., Haar-like features are used here) of the instances in the positive and negative bags are firstly obtained at the i-th frame. As shown in Figure 1, the feature vector in the green row represents an instance in the positive bag, and the feature vector in the red row represents the sample in the negative bag. To reduce the time complexity and improve the robustness of object tracking, we use the different random projection matrices to compress the high-dimensional multi-scale image feature vectors into the different low-dimensional feature vectors. Then, by exploiting the improved SMILE mechanism, the compressed Haar-like feature vectors are, respectively, used to update and select the best weak classifiers to obtain the G different strong classifiers $\{H_1, H_2, \ldots, H_G\}$. Then, the G different strong classifiers are combined into a final strong classifier by a multi-classifier fusion framework. Thereby, the object location with the largest response value is found by the combined final strong classifier in the search area of the $(i+1)$-th frame.

In the next sections, we will further elaborate compressive sensing, the SMILE mechanism, and the multi-classifier fusion framework.

3.2. Compressive Sensing. To avoid the high computational complexity and large storage consumption, compressive sensing is employed to object tracking [7, 17]. The compressive sensing technology [50–52] satisfies the condition of RIP (Restricted Isometry Property) [53] and uses the random projection matrix to reduce the high-dimensional multi-scale image features to the low-dimensional image domain in real time. The random projection matrix satisfying the Johnson–Lindenstrauss (J-L) lemma is proved to hold true for the RIP condition in compressive sensing [54].

Given that $\mathbf{R} \in \mathbb{R}^{n \times m}$ $(n \ll m)$ denotes a random projection matrix, where the n and m, respectively, denote the row number and column number of $\mathbf{R}$. Then, the high-dimensional spatial image feature vector $\mathbf{x} \in \mathbb{R}^m$ can be compressed into the low-dimensional image feature vector $\mathbf{v} \in \mathbb{R}^n$, expressed as $\mathbf{v} = \mathbf{R}\mathbf{x}$. As long as the matrix $\mathbf{R}$ satisfies the J-L lemma, $\mathbf{x}$ can be reconstructed from $\mathbf{v}$ with a low error probability. Similar to topology reconstruction, the compressed feature $\mathbf{v}$ still retains most of the original information [7].

At present, there are the four main types of random projection matrices suitable for the J-L lemma as follows, and the reduced low-dimensional image feature vector can preserve basically the information of the original image feature vector [7, 17, 50–52].

(1) Plus-Minus-One (PMO) or Bernoulli random projection [50]: $\mathbf{R} = 1/\sqrt{m}(r_{ij})$ is represented by $n \times m$

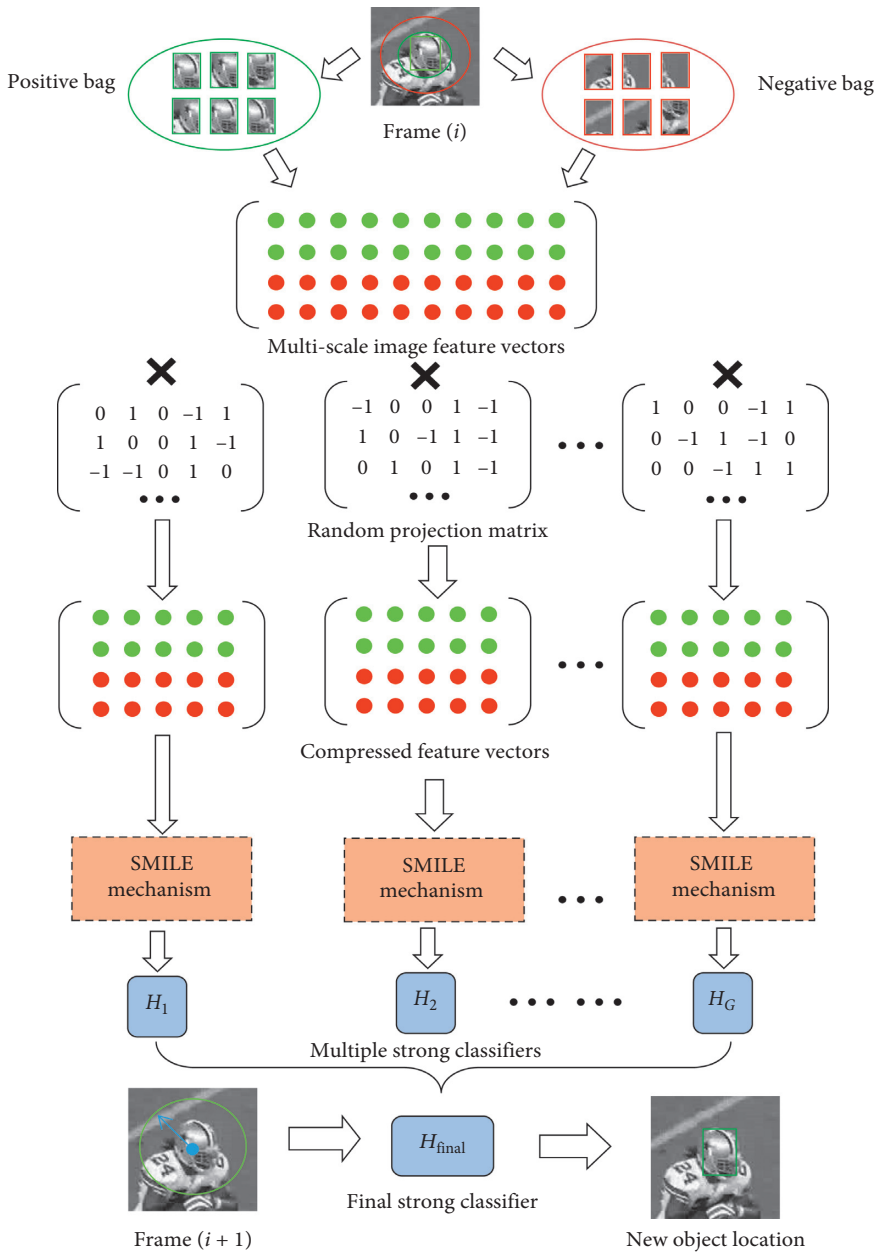


Figure 1: The algorithm flow.

matrices, where r_{ij} belongs to the set $\{-1, 1\}$, with the probability $\text{Prob}(r_{ij} = 1) = \text{Prob}(r_{ij} = -1) = 1/2$ (r_{ij} is Bernoulli's random variable).

(2) Achlioptas random projection [51]: $\mathbf{R} = 1/\sqrt{m}(r_{ij})$ is represented by $n \times m$ matrices, where r_{ij} belongs to the set $\{-\sqrt{3}, 0, \sqrt{3}\}$, and the probability is $\text{Prob}(r_{ij} = 0) = 2/3$, and $\text{Prob}(r_{ij} = \sqrt{3}) = \text{Prob}(r_{ij} = -\sqrt{3}) = 1/6$.

(3) Normal distribution random projection is also called Gaussian distribution random projection [17, 52]. According to J-L lemma, the random projection is represented by a $n \times m$ matrix $\mathbf{R} = 1/\sqrt{m}(r_{ij})$, where r_{ij} is the Gaussian distribution with $N(0, 1)$.

(4) The CT algorithm [7] uses the following random projection matrix:

$$r_{ij} = \sqrt{F} \times \begin{cases} 1 \text{ with probability } \dfrac{1}{2F}, \\ 0 \text{ with probability } 1 - \dfrac{1}{F}, \\ -1 \text{ with probability } \dfrac{1}{2F}. \end{cases} \quad (1)$$

The verification result in [55] shows that when F is 2 or 3 in equation (1), $\mathbf{R}$ satisfies J-L lemma.

The traditional CT algorithm [7] only uses a fixed random projection matrix in the whole process of tracking, and thus, the helpful information might be omitted,

especially in the complex environments. To avoid the above problem, we randomly select a sparse random projection matrix $\mathbf{R}$ from the above four main types of random projection matrices and thus obtain the different compressed feature vectors for each instance, which is inputted to the following online SMILE mechanism.

3.3. Online SMILE Mechanism. The workflow of the developed online SMILE mechanism is shown in Figure 2. In order to avoid the object drifting problem caused by the accumulated errors, we can filter some samples more likely to be negative in the positive bag by calculating the relative similarity of all instances in the positive bag. Then, the filtered positive bag and the negative bag are used to update all the weak classifiers. At the same time, the relative similarity is also employed to weight the objective function and then the best weak classifiers are selected to integrate a strong classifier from the updated weak classifier pool.

In the multiple instance learning paradigm, the training data are $\{(X_0, y_0), \ldots, (X_{N-1}, y_{N-1})\}$, where the bag $X_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{ij}, \ldots, \mathbf{x}_{iN_i}\}$; y_i is the bag label; $\mathbf{x}_{ij}$ is the j-th instance in the i-th bag; N_i is the number of instances in the i-th bag; and N is the total number of bags. The bag label is defined as $y_i = \max(y_{ij})$, where y_{ij} is the label of the j-th instance in the i-th bag, but the instance label is unknown during MIL training. In the MIL, it is known that a positive bag contains at least one positive instance, and all the instances in a negative bag are negative. In this paper, we select a set of image patches $X_1 = \{\mathbf{x}: \|d(\mathbf{x}) - d_t^*\| < \gamma\}, \gamma < s$ as a positive bag, where $\|\cdot\|$ represents the Euclidean distance; γ is the radius from the center point (in pixels); $\mathbf{x}$ is an image patch; $d(\mathbf{x})$ indicates the location of the image patch $\mathbf{x}$; d_t^* represents the location of the object at time t; and s is the search radius of the tracker. For the negative bag, we select the image patches from the annular region $X_0 = \{\mathbf{x}: \gamma < \|d(\mathbf{x}) - d_t^*\| < \beta\}$. Because a large set of potential samples are generated, we select a random subset of these samples as a negative bag as in [1].

In most cases, there are some negative samples in the positive bag. However, the traditional online multiple instance learning algorithms [1] treat all the instances in the positive bag as positive in the process of updating all the weak classifiers, which will degrade the performance of the tracker. In order to improve the robustness and accuracy of object tracking, the SMILE mechanism introduces the concept of relative similarity [44] to further filter the instances in the positive bag, and then, the relative similarity is also used to weight the instance probability and then optimize the objective function.

First, in the SMILE mechanism, the similarity between the two instances $\mathbf{x}_{ij}$ and $\mathbf{x}_{iq}$ is defined as follows:

$$S\left(\mathbf{x}_{ij}, \mathbf{x}_{iq}\right) = 0.5\left(\mathrm{NCC}\left(\mathbf{x}_{ij}, \mathbf{x}_{iq}\right) + 1\right), \tag{2}$$

where NCC is the normalized correlation coefficient.

For any given instance $\mathbf{x}_{ij}$ in a bag, we construct an object model $E = \{\mathbf{x}_{11}, \ldots, \mathbf{x}_{1N^+}, \mathbf{x}_{01}, \ldots, \mathbf{x}_{0N^-}\}$ to represent the object and the background information observed so far. In this paper, $\mathbf{x}_{11}, \ldots, \mathbf{x}_{1N^+}$ are the object samples from the first frame to the previous frame. And $\mathbf{x}_{01}, \ldots, \mathbf{x}_{0N^-}$ are the instances in the negative bag in the previous frame. Given an arbitrary instance $\mathbf{x}_{ij}$ and the object model E. The similarity measures for the MIL problem are defined as follows:

① Similarity with the positive nearest neighbour:

$$S^+\left(\mathbf{x}_{ij}, E\right) = \max_{\mathbf{x}_{1n} \in E} S\left(\mathbf{x}_{ij}, \mathbf{x}_{1n}\right). \tag{3}$$

② Similarity with the negative nearest neighbour:

$$S^-\left(\mathbf{x}_{ij}, E\right) = \max_{\mathbf{x}_{0n} \in E} S\left(\mathbf{x}_{ij}, \mathbf{x}_{0n}\right). \tag{4}$$

③ Relative similarity:

$$S^r\left(\mathbf{x}_{ij}, E\right) = \frac{S^+\left(\mathbf{x}_{ij}, E\right)}{S^+\left(\mathbf{x}_{ij}, E\right) + S^-\left(\mathbf{x}_{ij}, E\right)}. \tag{5}$$

The relative similarity varies from 0 to 1, and the larger S^r indicates that the image patch is more likely to be the object. Given the positive relative similarity coefficient θ, and when $S^r(\mathbf{x}_{ij}, E) < \theta$, the image patch $\mathbf{x}_{ij}$ is removed from the positive bag, and otherwise, it is retained in the positive bag.

In addition, the SMILE mechanism also uses the relative similarity to weight the instance probability, in order to more accurately estimate the bag probability $p(y_i|X_i)$. The probability of the positive bag to be positive is calculated as follows:

$$p\left(y_1 = 1 | X_1\right) = \sum_{j=1}^{N_1} w_{1j} p\left(y_1 = 1 | \mathbf{x}_{1j}\right). \tag{6}$$

Different from the WMIL algorithm [10], our algorithm uses the relative similarity to weight the instances, and the weight of each instance in the positive bag is calculated as $w_{1j} = (1/c)S^r(\mathbf{x}_{1j}, E)$, where c is a normalization constant.

The negative bag probability to be negative is calculated as follows:

$$p\left(y_0 = 0 | X_0\right) = \sum_{j=1}^{N_0} w_0\left(1 - p\left(y_0 = 1 | \mathbf{x}_{0j}\right)\right). \tag{7}$$

Since all negative instances are far from the object, w_0 is set to be a constant [10].

As shown in Figure 1, we compress the original high-dimensional multi-scale Haar-like feature vector to the low-dimensional feature vector by the random projection matrix in Section 3.2, a very sparse measurement matrix, in order to extract a small number of effective Haar-like features [7]. In the SMILE mechanism, each weak classifier h_k corresponding to a Haar-like feature f_k in the compressed feature vectors is constructed based on the Bayesian theorem, as follows:

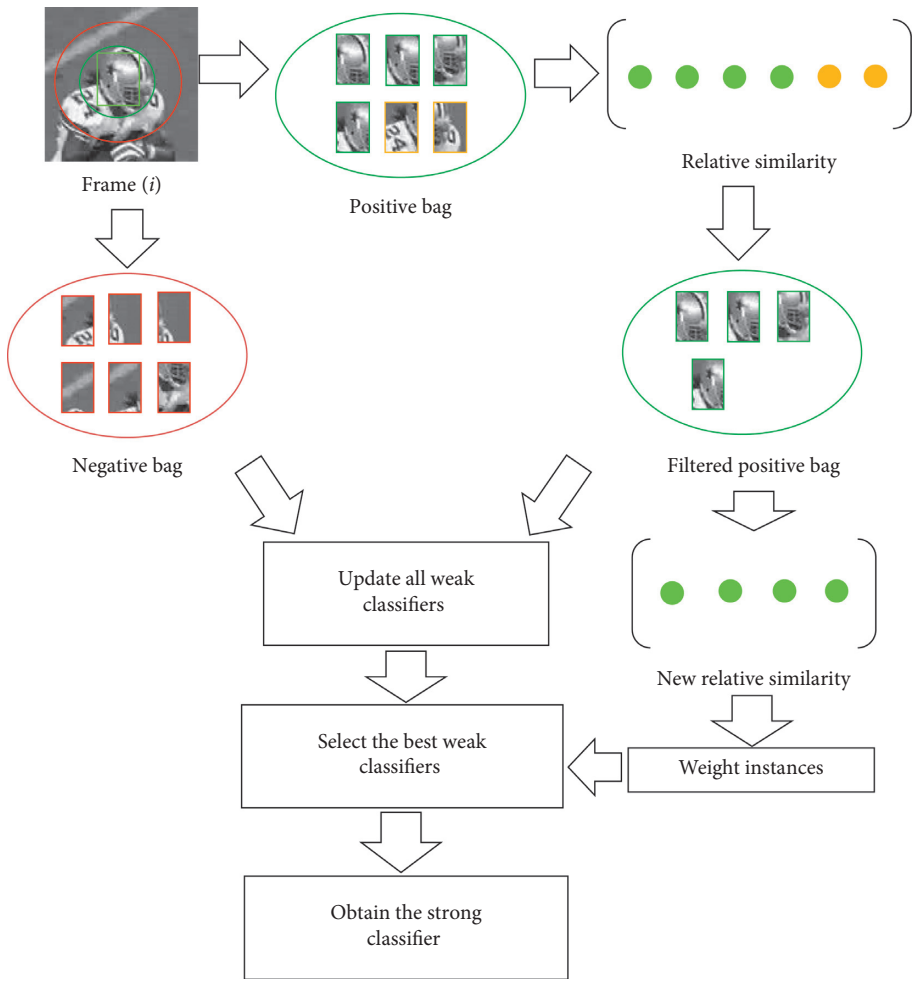


Figure 2: The SMILE mechanism.

$$h_k(\mathbf{x}_{ij}) = \ln\left(\frac{p(f_k(\mathbf{x}_{ij})|\, y_i = 1)}{p(f_k(\mathbf{x}_{ij})|\, y_i = 0)}\right), \tag{8}$$

where the conditional distributions are modelled as a Gaussian function, i.e., $p(f_k(\mathbf{x}_{ij})|y_i = 1) \sim N(\mu_1, \sigma_1)$ and $p(f_k(\mathbf{x}_{ij})|y_i = 0) \sim N(\mu_0, \sigma_0)$. The four parameters $\mu_1, \sigma_1, \mu_0, \sigma_0$ are updated with a learning rate parameter by the same schemes as the WMIL algorithm [10].

Then, the MCM algorithm selects the best weak classifiers by maximizing the log-likelihood function of bags $L = \sum_i (\log\, p(y_i|X_i))$. After the K iterations, the K best weak classifiers are integrated into a strong classifier H_g. In the multi-classifier fusion framework, we obtain the different G strong classifiers $\{H_1, H_2, \ldots, H_G\}$.

3.4. Multi-classifier Fusion Framework. In order to fuse the different G strong classifiers $\{H_1, H_2, \ldots, H_G\}$, the final strong classifier $H_{\text{final}} = H_{g^*}$ is calculated as follows:

$$g^* = \arg\min\left(\left|H_{g,\, g\in G} - \frac{1}{G}\sum_{g=1}^{G} H_g\right|\right). \tag{9}$$

In equation (9), the g represents the g-th strong classifier in the total G strong classifiers. The $1/G\sum_{g=1}^{G} H_g$ represents the mean response value of all G strong classifiers, and g^* is the subscript of the selected strong classifier. Thereby, the final strong classifier H_{final} is used to find the location of the sample with the largest response value in the search area as the tracking object. The pseudocode of the MCM algorithm is summarized in Algorithm 1.

4. Experimental Results

In this section, we first introduce the implementation details of the experiments. Secondly, we discuss the suitable number of the strong classifier. Finally, we evaluate the proposed algorithm and several competing algorithms on the two baseline datasets, i.e., OTB-2013 [56] and OTB-2015 [57], from the perspective of quantitative evaluation, time complexity analysis, and qualitative evaluation.

4.1. Datasets and Implementation Details. We evaluate the proposed tracking method and the six completing methods on two challenging object tracking datasets, i.e., OTB-2013

Input: t-th video frame.
Initialization: strong classifier $H = H^0 = 0$.
(1) Select a set of image patches $\chi^s = \{\mathbf{x}: \|d(\mathbf{x}) - d_t^*\| < s\}$ and calculate their feature vectors;
(2) Use the final strong classifier to estimate $p(y|\mathbf{x})$ for $\mathbf{x} \in X^s$;
(3) Update the object location $d_t^* = d(\arg\max_{\mathbf{x}\in\chi^s} p(y|\mathbf{x}))$ of the current frame and the object model E;
(4) Select a positive bag $X_1 = \{\mathbf{x}: \|d(\mathbf{x}) - d_t^*\| < \gamma\}$ and a negative bag $X_0 = \{\mathbf{x}: \gamma < \|d(\mathbf{x}) - d_t^*\| < \beta\}$;
(5) Initialize the iteration number $k = 1$;
(6) Randomly choose a random projection matrix $\mathbf{R} \in \mathbb{R}^{n\times m}$, and calculate the low-dimensional feature vector of each instance in the positive and negative bags;
(7) Normalize each instance in the positive bag, and calculate the similarity of any two instances $\mathbf{x}_{1j}$ and $\mathbf{x}_{1q}$ in equation (2);
(8) Calculate $S^+(\mathbf{x}_{1j}, E)$ in equation (3) and $S^-(\mathbf{x}_{1j}, E)$ in equation (4) for each instance in the positive bag;
(9) Calculate the relative similarity $S^r(\mathbf{x}_{1j}, E)$ in equation (5) for each instance in the positive bag;
(10) If $S^r(\mathbf{x}_{1j}, E) < \theta$, the image patch $\mathbf{x}_{1j}$ is deleted from the original positive bag;
(11) Update all the weak classifiers with the instances by using the filtered positive bag and the negative bag;
(12) Calculate the weight w_{1j} of each instance in the positive bag;
(13) Calculate the instance probability $p(y_i = 1|\mathbf{x}_{ij}) = \sigma(H^{k-1}(\mathbf{x}_{ij})) = 1/(1 + e^{-H^{k-1}(\mathbf{x}_{ij})})$;
(14) Calculate the weighted bag probability $p(y_i|X_i)$ according to equations (6) and (7);
(15) Calculate the log-likelihood function $L = \sum_i (\log\, p(y_i|X_i))$, and select the best weak classifier $h_k^* = \arg\max L(H^{k-1} + h)$;
(16) Repeat Steps 13–15 for K iterations. The k best weak classifier is integrated into the current strong classifier $H^k = H^{k-1} + h_k^*$;
(17) Obtain the g-th strong classifier $H_g = H^K$;
(18) Repeat Steps 5–17 for G iterations to generate the G strong classifier;
(19) Obtain the final strong classifier H_{g^*} by equation (9).
Output: the final strong classifier $H_{\text{final}} = H_{g^*}$.

Algorithm 1: Outline of our algorithm.

and OTB-2015, which contains the 50 and 100 challenging video sequences, respectively. The two datasets face different challenges in complex environments, such as rotation, occlusion, background clutter, and illumination variation. The search radius γ of the positive bag is 4.0 pixels, and the radii of the annular region of the negative bag are 4.0 pixels and 6.0 pixels, respectively, and the search radius s for searching the new object location is 25 pixels. The number of instances in the positive and negative bags is limited to be 45 and 50, respectively. The relative coefficient θ is set to be 0.7. The size of the sample box is a rectangle of 25×25 pixels. The weak classifier pool consists of 150 weak classifiers. The number of selected best weak classifiers is set to be 15. Our experiments are implemented on MATLAB 2016a, OpenCV 2.4.4, and visual studio 2010 with Intel Core i5-6300HQ, 2.30 GHz CPU and 8.0 GB RAM.

4.2. Parameter Analysis. We report the precision at an adaptive threshold [6, 13], i.e., $0.5 \times \min(w, h)$ when the number of strong classifiers is from 30 to 100 on the OTB-2013 and OTB-2015 datasets. Here, w and h denote the width and height of the object, respectively. This adaptive threshold roughly corresponds to the percentage of frames with at least a 50% overlap between the bounding box and the ground truth.

Figure 3 shows the precision under the different numbers of strong classifiers on the four video sequences, i.e., Bird1, Box, Crowds, and Soccer on the two datasets. From Figure 3, we can see that the tracking precision increases gradually and then begins to drop with the increase of the number of strong classifiers. When the number of strong classifiers is in [55, 85], the tracking precision is the best for most of the video sequences.

4.3. Comparisons with Representative Trackers. We evaluated the proposed trackers with six competing trackers, including CSK [11], CT [6], KCF [58], Struck [10], WMIL [16], and TLD [9]. Quantitative evaluation, time complexity analysis, and qualitative evaluation are reported in this section.

4.3.1. Quantitative Evaluation. The center location error and precision are employed to evaluate the experimental results of all the comparison algorithms. The center location error (CLE) is defined as the Euclidean distance between the object center location and the ground truth. The CLE is formally expressed as follows:

$$\text{CLE} = \frac{1}{N_f} \sum_{i=1}^{N_f} \text{dist}(\text{center}(i), gt(i)). \tag{10}$$

The function $\text{dist}(.,.)$ is used to calculate Euclidean distance. N_f is the total number of frames; $\text{center}(i)$ and $gt(i)$ represent the center location given by the tracking algorithm and the ground truth center location in each frame. The smaller the value of CLE is, the lower the error of the tracking algorithm is.

In addition, we also report the precision at an adaptive threshold, i.e., $0.5 \times \min(w, h)$. The precision plots show the percentage of the frames whose center location errors are lower than a certain threshold, defined as follows:

$$\text{precision}(\zeta) = \frac{1}{N} \sum_{i=1}^{N} \psi(\text{dist}(\text{center}(i), gt(i)) \le \zeta), \tag{11}$$

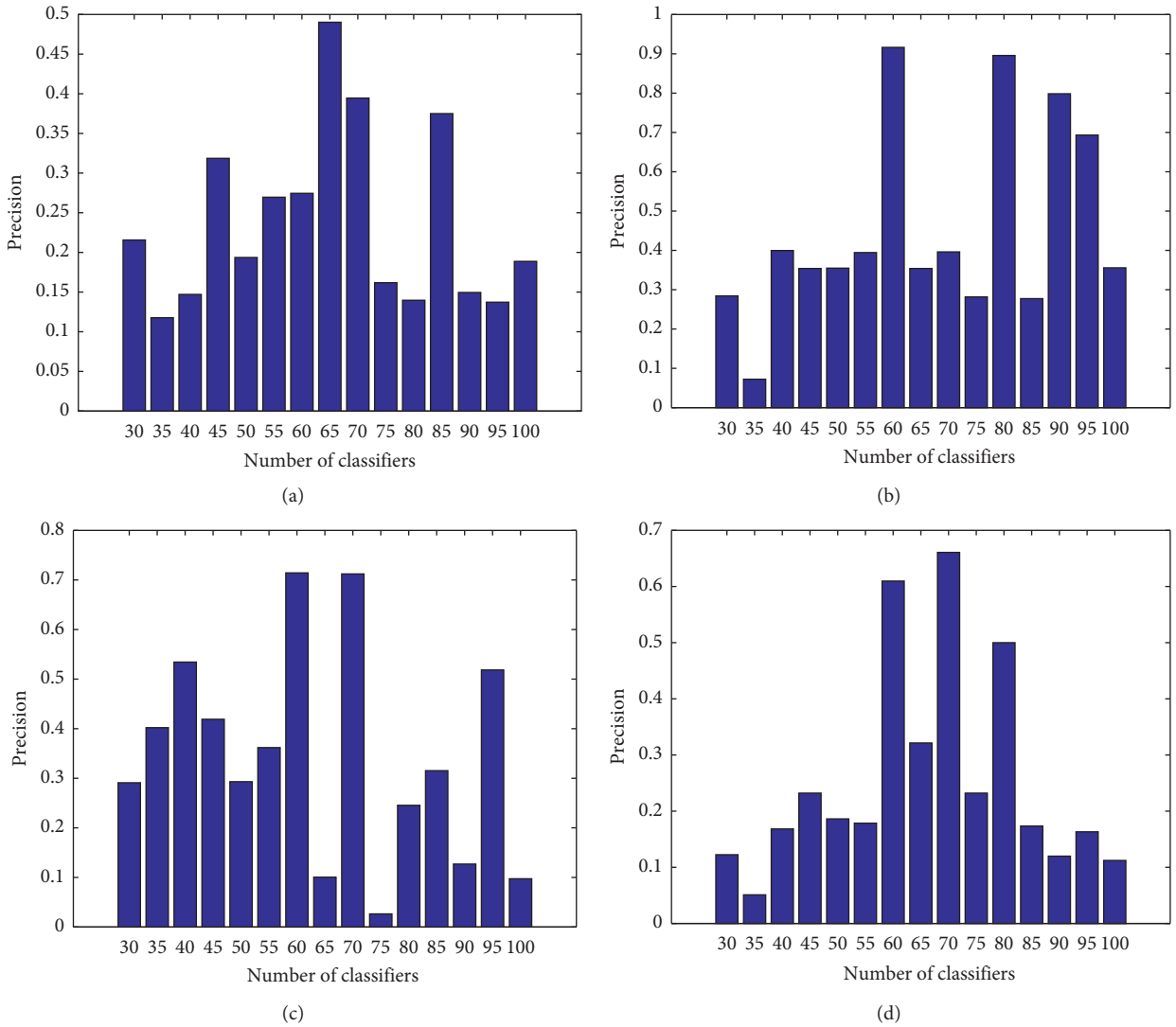


Figure 3: Precision at the different numbers of strong classifiers for our algorithm on the four video sequences. (a) Bird1, (b) Box, (c) Suv, and (d) Soccer.

where ζ represents the threshold. When $\text{dist}(.,.) \leq \zeta$, $\psi(\cdot) = 1$, and otherwise, $\psi(\cdot) = 0$.

In the experiments, we compare the center location error and the tracking precision of the proposed algorithm MCM with the competing trackers on the OTB-2013 and OTB-2015 datasets according to the 11 challenging factors, i.e., Illumination Variation (IV), Scale Variation (SV), Occlusion (OCC), Deformation (DEF), Motion Blur (MB), Fast Motion (FM), In-Plane Rotation (IPR), Out-of-Plane Rotation (OPR), Out-of-View (OV), Background Clutters (BC), and Low Resolution (LR).

Tables 1 and 2 show, respectively, the average center location error (in pixels) for each challenge factor on the OTB-2013 and OTB-2015 datasets. From the last row of Tables 1 and 2, we can see that the MCM algorithm obtains the lowest average center location error compared with the other competing algorithms. In the total of eleven challenge factors, the MCM has the lower center location errors than the other compared methods at the ten challenges on the

Table 1: Average center location error (in pixels) for each challenge factor in OTB-2013.

Challenge factor	CSK	CT	KCF	Struck	WMIL	TLD	MCM
IV	122.78	140.88	**57.42**	150.62	94.33	124.23	*76.59*
SV	450.79	92.21	*69.57*	118.74	78.92	130.93	**57.72**
OCC	523.59	103.39	*63.44*	122.27	79.26	150.62	**51.08**
DEF	721.04	147.85	**55.74**	161.01	99.33	186.27	*69.20*
MB	189.68	103.06	*81.99*	125.26	83.37	113.97	**64.67**
FM	204.85	106.38	*77.04*	119.60	78.23	128.46	**54.45**
IPR	146.50	101.37	*73.84*	124.34	79.45	125.53	**53.99**
OPR	473.08	111.67	*69.33*	131.17	80.29	155.43	**56.17**
OV	121.16	71.07	*66.26*	125.35	69.42	122.82	**45.54**
BC	661.00	142.34	86.08	161.67	*84.86*	113.14	**61.49**
LR	154.15	97.59	**50.22**	136.84	73.68	131.85	*50.31*
ALL	377.53	117.94	*69.14*	131.46	82.66	148.08	**55.49**

The last row represents the average center location error on all the 50 datasets. Bold fonts indicate the best performance while italic fonts indicate the second best.

Table 2: Average center location error (in pixels) for each challenge factor on the OTB-2015 dataset.

Challenge factor	CSK	CT	KCF	Struck	WMIL	TLD	MCM
IV	484.84	89.58	*64.37*	197.16	205.23	134.66	**48.05**
SV	379.01	79.47	*75.95*	180.59	186.45	129.80	**42.62**
OCC	429.49	78.67	*72.02*	202.60	192.44	133.90	**40.10**
DEF	481.93	93.87	*80.69*	227.46	179.21	142.79	**44.60**
MB	897.57	*86.94*	106.29	178.91	179.25	131.01	**43.89**
FM	147.24	*77.35*	83.93	176.75	181.55	128.54	**41.03**
IPR	100.36	72.08	*57.11*	181.02	187.67	110.84	**37.70**
OPR	580.14	69.84	*62.41*	204.64	189.66	120.64	**38.10**
OV	162.25	*87.60*	98.38	152.75	208.33	153.18	**36.30**
BC	481.65	98.96	*67.46*	186.94	201.99	143.19	**45.15**
LR	130.46	**49.92**	*55.30*	114.25	157.96	104.11	56.71
ALL	388.63	80.39	*74.90*	182.10	188.16	130.24	**43.11**

The last row represents the average center location error on all the 100 datasets. Bold fonts indicate the best performance while italic fonts indicate the second best.

OTB-2015 dataset and has the lower CLE values at the eight challenges on the OTB-2013 dataset. On the two datasets, the MCM works best in Out-of-View (OV) due to the employed relative similarity based on the object model, which can overcome the dramatic appearance changes to significantly improve the tracking performance for the most challenging environments on the two datasets. Overall, the center location errors of the MCM algorithm with multiple strong classifiers are lower than CT and MIL using the single strong classifier.

Tables 3 and 4 list, respectively, the average tracking precision at the threshold of $0.5 \times \min(w, h)$ of the competing algorithms under the different challenge factors on the OTB-2013 and OTB-2015 datasets. In the last row of Table 3, the average precision of the MCM algorithm is only lower than KCF by 0.043 on the OTB-2013 dataset. In Scale Variation (SV), Out-of-View (OV), and Low Resolution (LR), the precision of the MCM algorithm is higher than that of the KCF algorithm on the OTB-2013 dataset. In the last row of Table 4, the average precision of the MCM algorithm is 0.553, which is highest among all the comparison algorithms on the OTB-2015 dataset. In Scale Variation (SV), Occlusion (OCC), Deformation (DEF), Out-of-View (OV), and Background Clutters (BC), the precision of the MCM algorithm is higher than that of the other competing algorithms on the OTB-2015 dataset. Overall, by fusing multiple strong classifiers, the precision of the MCM algorithm is higher than that of CT and MIL using the single strong classifier on the two datasets. Therefore, the effectiveness of the multi-classifier fusion of the MCM method is verified.

Figures 4 and 5, respectively, show the center location error plots on several representative video sequences on the OTB-2013 and OTB-2015 datasets. In addition, as shown in Figures 4 and 5, we can intuitively see that the MCM tracker yields much lower center location errors for most of the frames than the other six algorithms on the two benchmarks. Figures 6 and 7, respectively, present the precision plots over thresholds in [0, 50] on eight representative sequences on the two datasets. Our algorithm has the higher precision than the other comparison algorithms among the thresholds [0, 50] for most of video sequences. This is because compressive sensing and the improved online SMILE mechanism play an important role in the MCM algorithm. As a whole, the proposed algorithm is more effective to avoid the severe drifting problem than the other competing trackers in the complex environments.

Table 3: Average precision at the threshold of $0.5 \times \min(w, h)$ for each challenge factor in OTB-2013.

Challenge factor	CSK	CT	KCF	Struck	WMIL	TLD	MCM
IV	*0.470*	0.237	**0.630**	0.062	0.408	0.375	0.451
SV	0.278	0.303	*0.446*	0.076	0.344	0.394	**0.459**
OCC	0.303	0.284	**0.550**	0.081	0.294	0.376	*0.468*
DEF	0.300	0.304	**0.580**	0.069	0.386	0.330	*0.450*
MB	0.203	0.216	*0.408*	0.080	0.340	**0.436**	0.393
FM	0.200	0.237	0.407	0.077	0.339	**0.429**	*0.421*
IPR	0.228	0.282	**0.502**	0.064	0.379	0.432	*0.451*
OPR	0.309	0.269	**0.539**	0.080	0.365	0.403	*0.461*
OV	0.203	0.370	0.375	0.095	0.460	*0.485*	**0.568**
BC	0.422	0.264	**0.575**	0.048	0.418	0.403	*0.546*
LR	0.291	0.232	0.441	0.070	0.391	*0.444*	**0.498**
ALL	0.330	0.284	**0.508**	0.066	0.371	0.433	*0.465*

The last row represents the average precision on all the 50 datasets. Bold fonts indicate the best performance while italic fonts indicate the second best.

Table 4: Average precision at the threshold of $0.5 \times \min(w, h)$ for each challenge factor on the OTB-2015 dataset.

Challenge factor	CSK	CT	KCF	Struck	WMIL	TLD	MCM
IV	0.472	0.334	**0.685**	0.051	0.033	0.532	*0.609*
SV	0.320	0.426	0.471	0.055	0.086	*0.533*	**0.541**
OCC	0.392	0.370	*0.550*	0.057	0.056	0.522	**0.588**
DEF	0.376	0.407	*0.543*	0.049	0.075	0.486	**0.582**
MB	0.200	0.337	0.365	0.085	0.107	**0.529**	*0.519*
FM	0.329	0.379	0.468	0.073	0.073	**0.568**	*0.555*
IPR	0.228	0.282	**0.502**	0.064	0.379	0.432	*0.451*
OPR	0.460	0.400	**0.602**	0.041	0.054	0.558	*0.582*
OV	0.248	0.380	0.313	0.069	0.073	*0.459*	**0.629**
BC	0.513	0.379	*0.626*	0.035	0.042	0.459	**0.643**
LR	0.387	0.305	**0.525**	0.076	0.070	*0.465*	0.386
ALL	0.357	0.364	*0.513*	0.060	0.095	0.505	**0.553**

The last row represents the average precision on all the 100 datasets. Bold fonts indicate the best performance while italic fonts indicate the second best.

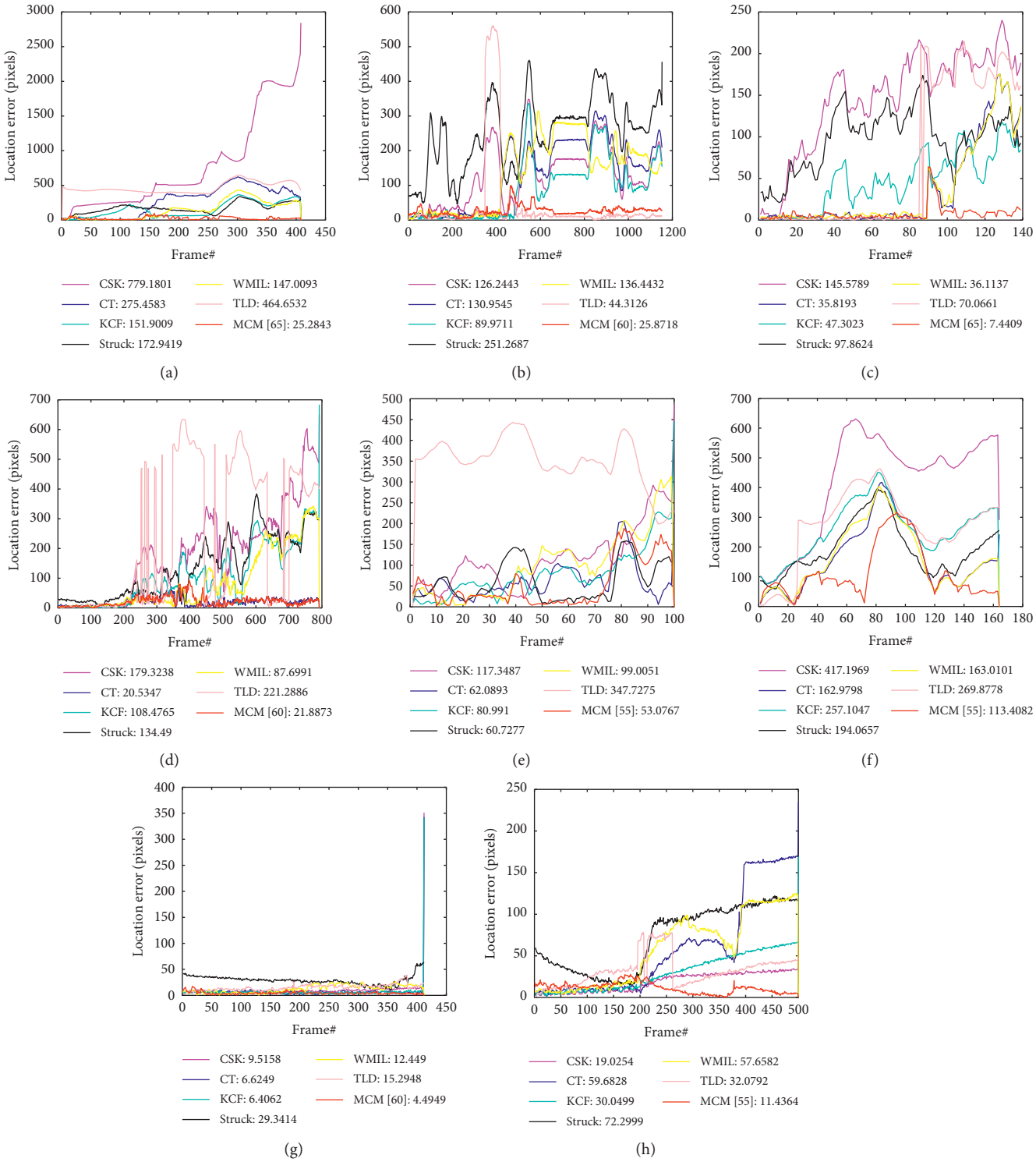


Figure 4: Center location error plots for six state-of-the-art trackers and our algorithm on several representative video sequences on the OTB-2013 dataset. In the upper right corner, the average center location errors of all the frames on each sequence are listed for all the comparison methods. (a) Bird1, (b) Box, (c) Couple, (d) Human6, (e) Matrix, (f) MotorRolling, (g) Walking, and (h) Walking2.

4.3.2. Time Complexity Analysis. We use the number of frames per second (FPS) to evaluate the time complexity of the algorithm. The larger the FPS is, the shorter the running time of the algorithm is, and the lower the time complexity of the algorithm is. Table 5 lists the average FPS values of different algorithms on the 100 challenging video sequences of the OTB-2015 dataset. As shown in Table 5, the tracking speed of the MCM algorithm is around 12 frames per second, and the MCM has the faster tracking speed than the Struck, WMIL, and TLD methods. Although it has the lower

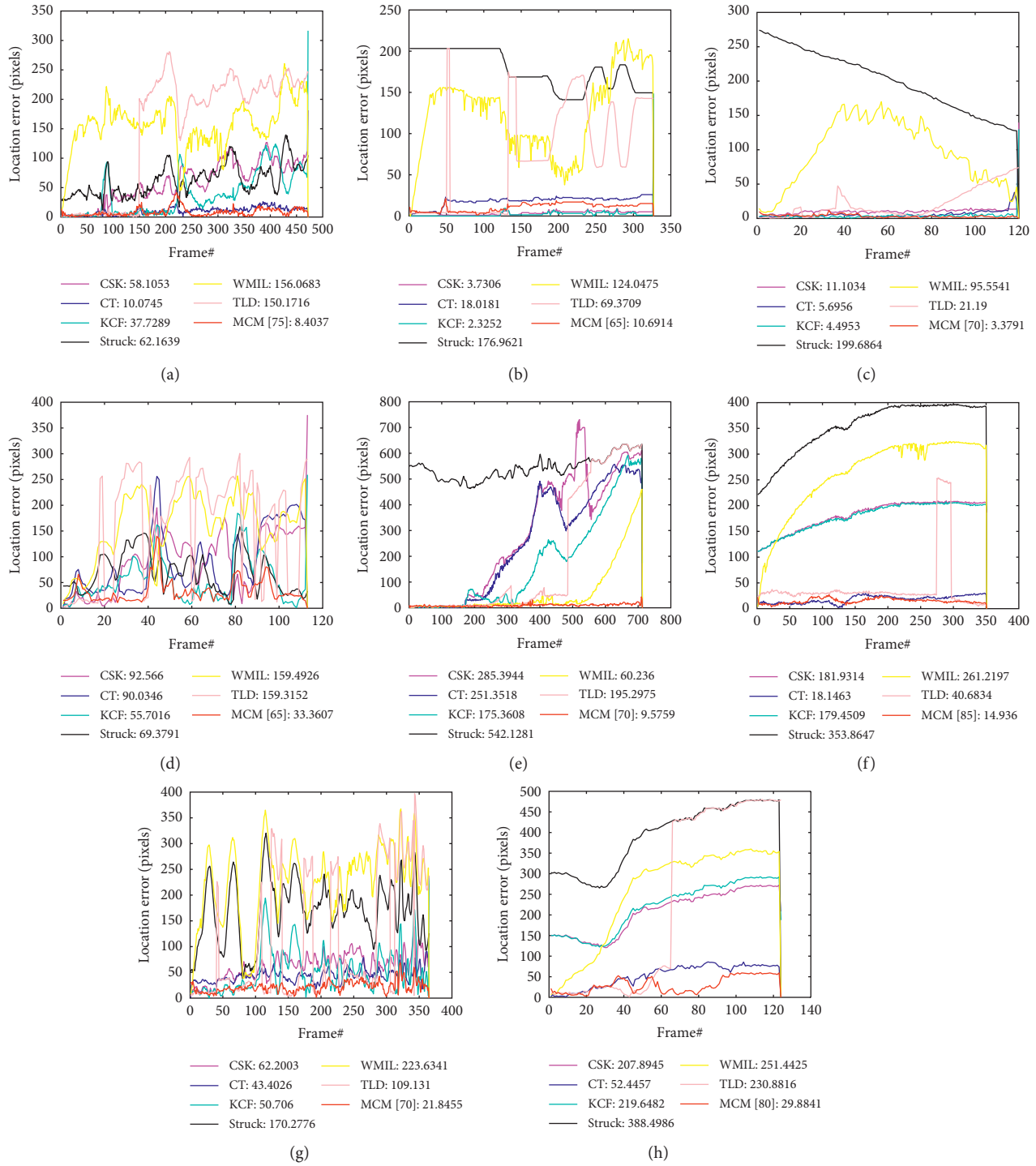


Figure 5: Center location error plots for six state-of-the-art trackers and our algorithm on several representative video sequences on the OTB-2015 dataset. In the upper right corner, the average center location errors of all the frames on each sequence are listed for all the comparison methods. (a) ClifBar, (b) Coupon, (c) Crossing, (d) DragonBaby, (e) Human5, (f) Singer1, (g) Tiger2, and (h) Trans.

tracking speed than the CSK, KCF, and CT algorithms, the MCM algorithm obtains the lower center location errors and the higher precision than the three compared algorithms, due to the multi-classifier fusion framework employed in this paper.

4.3.3. Qualitative Evaluation. Figure 8 summarizes the qualitative evaluation of the proposed algorithm and the six competing trackers under seven representative challenge sequences. In the Basketball video sequence (see Figure 8(a)), there are three kinds of challenging factors, i.e.,

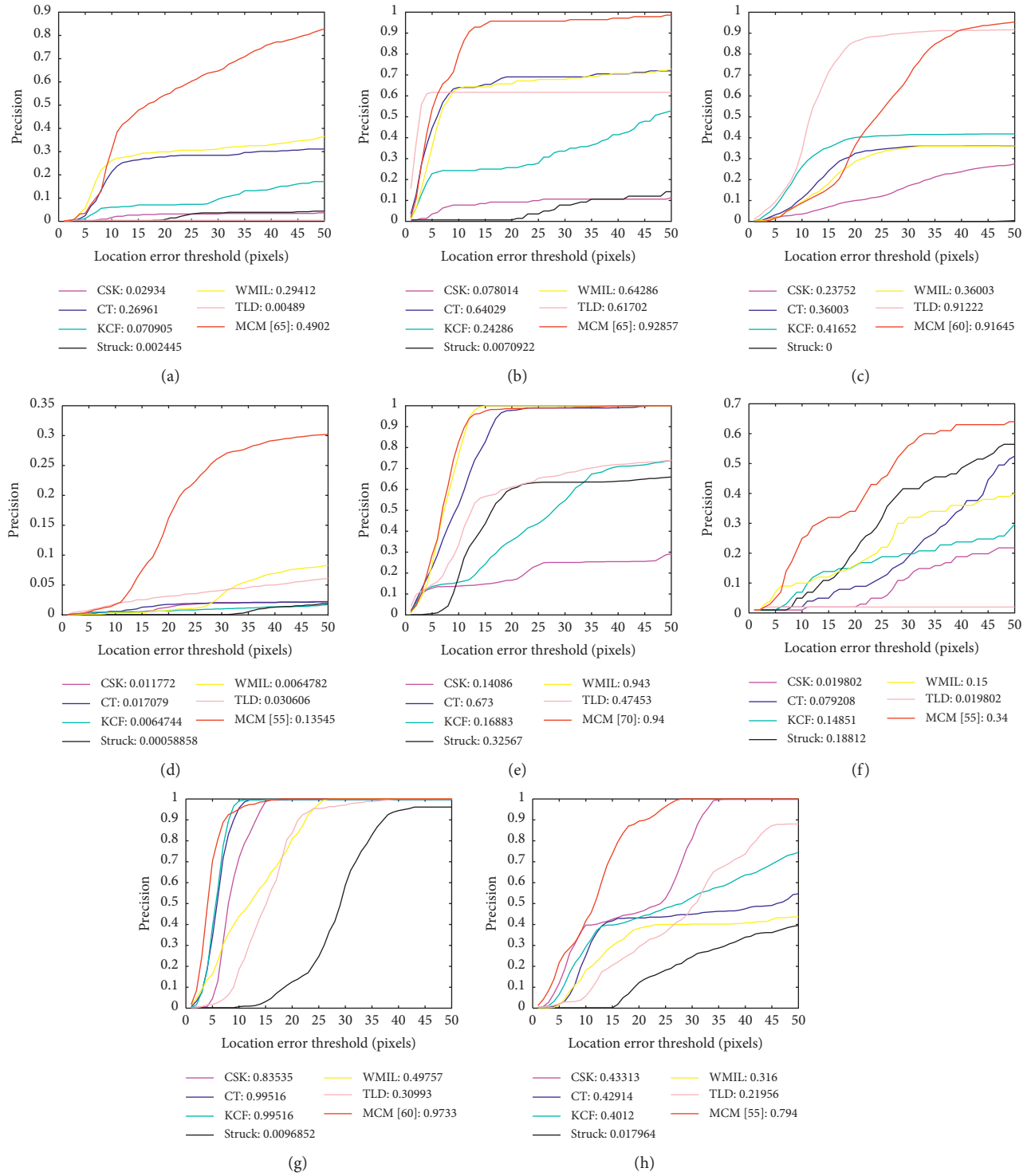


Figure 6: Precision plots for six state-of-the-art trackers and our algorithm on several representative video sequences on the OTB-2013 dataset. In the upper right corner, the precision at the threshold $0.5 \times \min(w, h)$ of all the comparison methods is listed. (a) Bird1, (b) Couple, (c) Box, (d) Human3, (e) Panda, (f) Matrix, (g) Walking, and (h) Walking2.

fast motion, scale variation, and occlusion. In the #17 frame, the Struck algorithm loses the object, and the KCF and WMIL methods are partially deviated from the object when the occlusion occurs. When the object moves fast and the scale variations happen, the KCF algorithm drifts heavily (see the #63 frame), and the CT and TLD algorithms also suffer from drifting (see the #247 frame). The CSK algorithm gradually deviates from the object and then the object is lost from the #489 frame. The CT algorithm also has a severe offset in the subsequent #629 frame. The proposed MCM

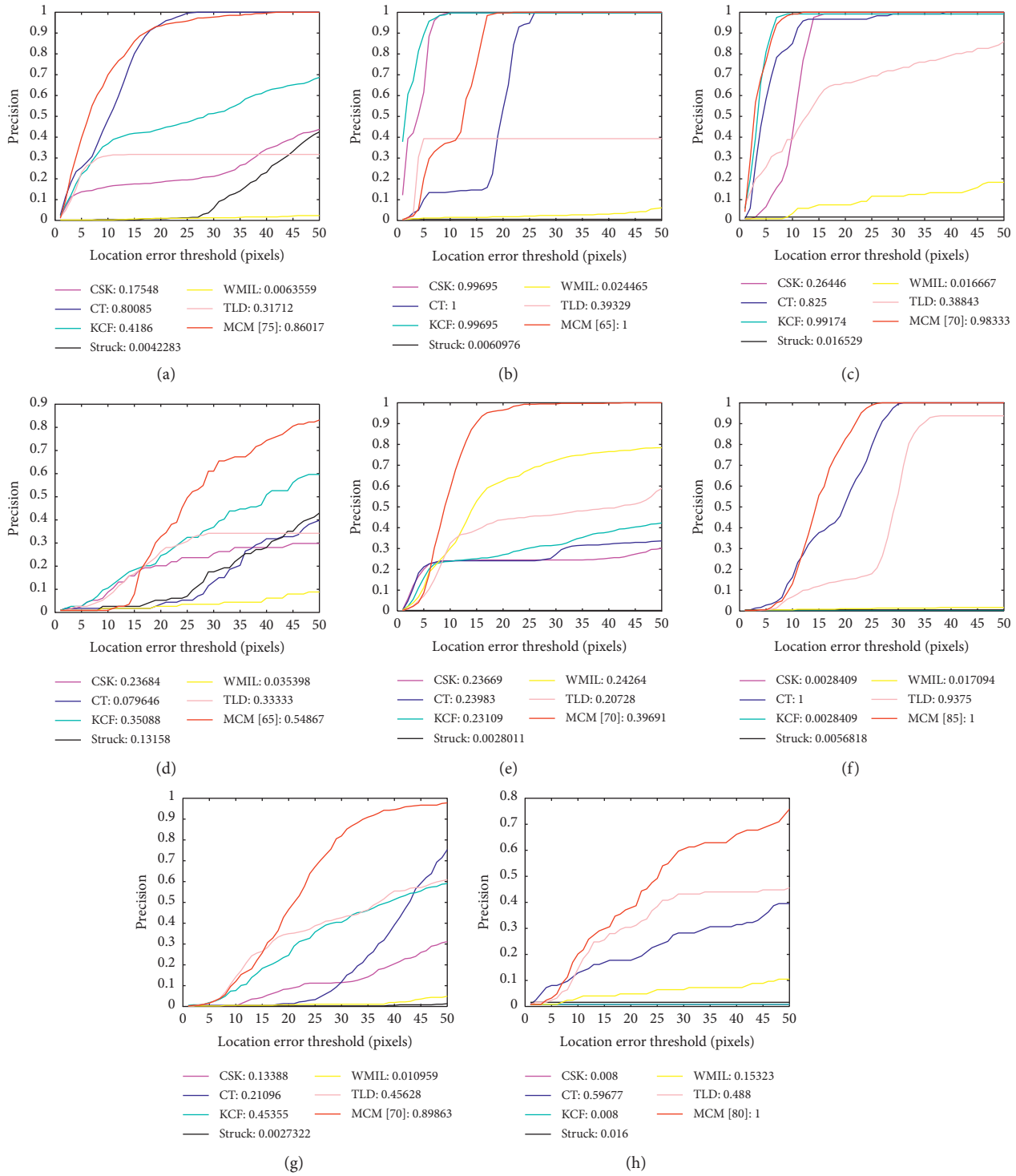
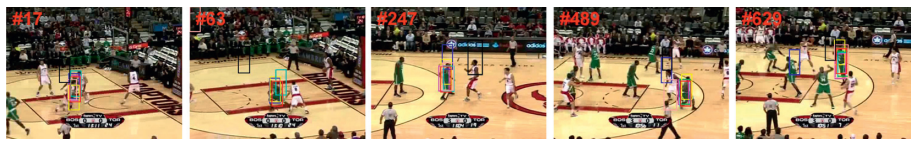


Figure 7: Precision plots for six state-of-the-art trackers and our algorithm on several representative video sequences on the OTB-2015 dataset. In the upper right corner, the precision at the threshold $0.5 \times \min(w, h)$ of all the comparison methods is listed. (a) ClifBar, (b) Coupon, (c) Crossing, (d) DragonBaby, (e) Human 5, (f) Singer1, (g) Tiger2, and (h) Trans.

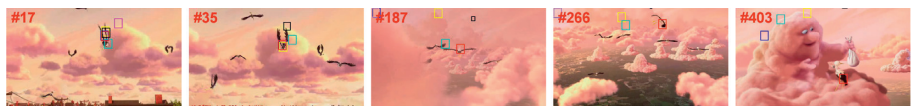
Table 5: The average number of frames per second (FPS) for the state-of-the-art trackers and our algorithm on the 100 challenging video sequences of the OTB-2015 dataset.

Methods	CSK	CT	KCF	Struck	WMIL	TLD	MCM
FPS	342	64	192	8	10	4	12



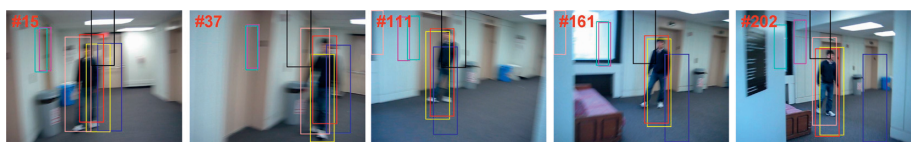
CSK Struck TLD
CT WMIL MCM
KCF

(a)



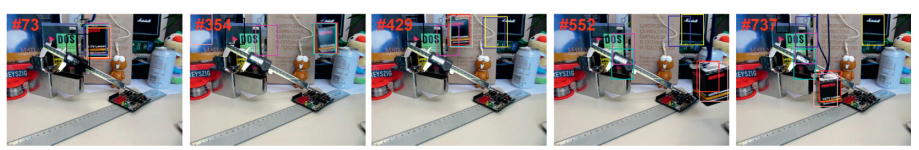
CSK Struck TLD
CT WMIL MCM
KCF

(b)



CSK Struck TLD
CT WMIL MCM
KCF

(c)



CSK Struck TLD
CT WMIL MCM
KCF

(d)

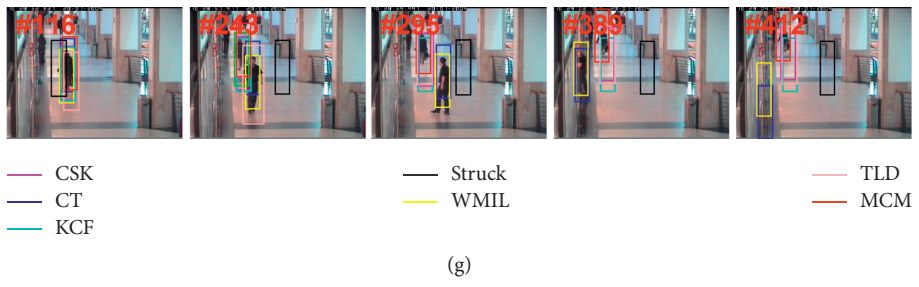
CSK Struck TLD
CT WMIL MCM
KCF

(e)

CSK Struck TLD
CT WMIL MCM
KCF

(f)

Figure 8: Continued.



(g)

Figure 8: Tracking results of CSK, CT, KCF, Struck, WMIL, TLD, and the proposed MCM algorithm on seven image sequences on the OTB-2013 and OTB-2015 datasets. From top to bottom, (a) Basketball, (b) Bird 1, (c) BlurBody, (d) Box, (e) Panda, (f) Singer2, and (g) Walking2.

algorithm performs well in most frames, where the object can be accurately tracked in the face of rapid movement and scale variation.

There are three tracking difficulties in the Bird1 sequence (see Figure 8(b)), i.e., fast motion of bird wings, long-term object occlusion, and rapid scale variation. In the initial fast motion, CSK loses the object firstly (see the #17 frame), and then the KCF and Struck algorithms drift severely (see the #35 frame). After long-term occlusion of cloud, only the MCM algorithm still tracks the object and the other algorithms are severely deviated from the ground truth (see the #187 frame). For the subsequent scale variations, the MCM algorithm still keeps tracking the object well (see the #266 and #403 frames).

In the BlurBody sequence (see Figure 8(c)), because the camera always shakes, some frames become blurred, which will have a great impact on the object tracking. During the tracking process, most algorithms have serious large-scale drifting, but the MCM algorithm is better to adapt to this situation.

In the Box video sequence (see Figure 8(d)), due to the occlusion problem in the moving process of the box, the CSK, TLD, and Struck algorithms drift successively, and then the object is lost (see the #73, #354, and #429 frames). The CT and WMIL and KCF algorithms miss the target in the subsequent frames with scale variations and rotations (see the #552 and #737 frames). However, the MCM algorithm has excellent performance in face of these challenge factors and can accurately track the target.

In the Panda sequence (see Figure 8(e)), a small Panda suffers from fast movements and occlusions. The Struck, CSK, and KCF algorithms have lost the object gradually, and the TLD algorithm fails due to scale variations during the tracking process (see the #160 and #609 frames), and the other algorithms are more accurate and stable to track the object.

A singer in the Singer2 sequence (see Figure 8(f)) has the large-scale variations and rotational deformations during the tracking process. In the scale variations of the object, the TLD, Struck, CSK, and CT algorithms work poor (see the #40, #107, and #269 frames), while the other algorithms can accurately track the target. In the movements of the singer, the KCF and WMIL algorithms are partially drifting from the object, and the MCM algorithm tracks the target more accurately during the tracking process.

In the Walking2 dataset (see Figure 8(g)), the main challenging factor is the occlusions of two persons. After the first occlusion, the Struck, TLD, CT, and WMIL algorithms drift to track the error object (see the #243 frame). After the second occlusion, the CSK method also begins to have a large offset (see the #389 frame). When the target becomes smaller and the scale variation occurs, the MCM algorithm is able to track the target accurately, and yet the other algorithms partially depart from the object (see the #412 frame).

In summary, the MCM algorithm outperforms the other compared trackers and can be more accurate and robust to track objects in the challenging factors of rapid motions, scale variations and object occlusions, and so on.

5. Conclusions

In this paper, we present an effective and robust object tracking algorithm, termed as MCM. We employ the multi-classifier fusion framework to integrate the advantages of multiple different discriminative strong classifiers, where the different random projection matrices are used to extract the low-dimensional feature vectors based on the compressive sensing technology. Then, an improved SMILE mechanism is presented to further filter and weight the instances in the positive bag by introducing the concept of relative similarity. Experimental results show that the proposed MCM algorithm performs favourably against several representative tracking algorithms on the two popular tracking benchmark datasets.

Data Availability

The data used to support the findings of the study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Natural Science Foundation of Fujian Province of China (nos. 2018J01576 and 2018J01574), the National Key R&D Program of China (no. 2017YFB1302400), and the National Natural Science Foundation of China (nos. 61672442 and 61773325). The authors

are grateful to the Computer Vision Lab, Hanyang University, Seoul, Korea, who provides the dataset publicly (Visual Tracker Benchmark, http://www.visual-tracking.net).

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *Acm Computing Surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[2] S. Chen, S. Li, S. Su, D. Cao, and R. Ji, "Online semi-supervised compressive coding for robust visual tracking," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 793–804, 2014.

[3] Z. Zhou, R. Zhang, and Z. Zhu, "RETRACTED: uncalibrated dynamic visual servoing via multivariate adaptive regression splines and improved incremental extreme learning machine," *ISA Transactions*, vol. 92, pp. 298–314, 2019.

[4] Z. Zhou, C. Wang, Z. Zhu, Y. Wang, and D. Yang, "Sliding mode control based on a hybrid grey-wolf-optimized extreme learning machine for robot manipulators," *Optik*, vol. 185, pp. 364–380, 2019.

[5] J. Bao, A. Song, and H. Tang, "Navigation method for reconnaissance robot based on vision object tracking," *Journal of Southeast University (Natural Science Edition)*, vol. 42, no. 3, pp. 399–405, 2012.

[6] B. Babenko, M. H. Ming-Hsuan Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[7] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proceedings of the 12th European Conference on Computer Vision*, pp. 864–877, Florence, Italy, October 2012.

[8] D. S. Bolme, J. R. Beveridge, B. A. Draper et al., "Visual object tracking using adaptive correlation filters," in *Proceedings of the Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, San Francisco, CA, USA, June 2010.

[9] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2011.

[10] S. Hare, S. Golodetz, A. Saffari et al., "Struck: structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, 2015.

[11] J. F. Henriques, R. C. R. Caseiro, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proceedings of the 12th European conference on Computer Vision*, pp. 702–715, Florence, Italy, October 2012.

[12] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4293–4302, Las Vegas, NV, USA, July 2016.

[13] S. Chen, S. Li, R. Ji, Y. Yan, and S. Zhu, "Discriminative local collaborative representation for online object tracking," *Knowledge-Based Systems*, vol. 100, pp. 13–24, 2016.

[14] M. J. Black and A. D. Jepson, "EigenTracking: robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[15] S. Avidan, "Support vector tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.

[16] K. Zhang and H. Song, "Real-time visual tracking via online weighted multiple instance learning," *Pattern Recognition*, vol. 46, no. 1, pp. 397–411, 2013.

[17] H. Li, C. Shen, and Q. Shi, "Real-time visual tracking using compressive sensing," in *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1305–1312, Colorado Springs, CO, USA, June 2011.

[18] R. Havangi, "Target tracking based on improved unscented particle filter with markov chain monte carlo," *IETE Journal of Research*, vol. 64, no. 6, pp. 873–885, 2018.

[19] L. Bertinetto, J. Valmadre, S. Golodetz et al., "Staple: complementary learners for real-time tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1401–1409, Las Vegas, NV, USA, June 2016.

[20] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised online boosting for robust tracking," in *Proceedings of the European Conference on Computer Vision*, pp. 234–247, Marseille, France, October 2008.

[21] B. Ma, L. Huang, J. Shen, and L. Shao, "Discriminative tracking using tensor pooling," *IEEE Transactions on Cybernetics*, vol. 46, no. 11, pp. 2411–2422, 2015.

[22] D. A. Ross, J. Lim, R. S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1–3, pp. 125–141, 2008.

[23] X. Mei and H. Ling, "Robust visual tracking and vehicle classification via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2259–2272, 2011.

[24] V. Tomas, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[25] L.-T. Nga, T. Le-Tien, and L. Mai, "A study on particle filter based on KLD-resampling for wireless patient tracking," *Industrial Engineering and Management Systems*, vol. 16, no. 1, pp. 92–102, 2017.

[26] R. T. Collins, Y. Yanxi Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

[27] N. C. Oza, "Online bagging and boosting," in *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2340–2345, Waikoloa, HI, USA, October 2005.

[28] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via online boosting," in *Proceedings of the British Machine Vision Conference*, vol. 1, p. 6, Edinburgh, UK, September 2006.

[29] G. Bhat, J. Johnander, M. Dartin et al., "Unveiling the power of deep tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 483–498, Munich, Germany, September 2018.

[30] Z. Cui, S. Xiao, J. Feng et al., "Recurrently target-attending tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1449–1458, Las Vegas, NV, USA, June 2016.

[31] Y. Li, L. Qi, and S. Tan, "Object tracking algorithm based on online semi-supervised boosting with structural constraints," *Computer Engineering & Applications*, vol. 53, no. 23, pp. 129–134, 2017.

[32] H. Nam, M. Baek, and B. Han, "Modeling and propagating CNNs in a tree structure for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[33] X. Lu, B. Ni, C. Ma, and X. Yang, "Learning transform-aware attentive network for object tracking," *Neurocomputing*, vol. 349, no. 15, pp. 133–144, 2019.

[34] M. Najafi, Z. Nadealian, S. Rahmanian, and V. Ghafarinia, "An adaptive distributed approach for the real-time vision-based navigation system," *Measurement*, vol. 145, pp. 14–21, 2019.

[35] L. Marata, J. Chuma, I. Ngebani, A. Yahya, and O. L. A. López, "Monte carlo mean for non-gaussian autonomous object tracking," *Computers & Electrical Engineering*, vol. 76, pp. 389–397, 2019.

[36] F. Jorquera, S. Hernández, and D. Vergara, "Probability hypothesis density filter using determinantal point processes for multi object tracking," *Computer Vision and Image Understanding*, vol. 183, pp. 33–41, 2019.

[37] G. Pan, G. Chen, W. Kang, and J. Hou, "Correlation filter tracker with siamese: a robust and real-time object tracking framework," *Neurocomputing*, vol. 358, pp. 33–43, 2019.

[38] Y. Xiao, J. Li, B. Du et al., "Robust correlation filter tracking with multi-scale spatial view," *Neurocomputing*, vol. 358, no. 17, pp. 119–140, 2019.

[39] Q. Qi, S. Zhao, W. Zhao et al., "High-speed video salient object detection with temporal propagation using correlation filter," *Neurocomputing*, vol. 356, no. 3, pp. 107–118, 2019.

[40] L. Wang and C. Pan, "Visual object tracking via a manifold regularized discriminative dual dictionary model," *Pattern Recognition*, vol. 91, pp. 272–280, 2019.

[41] B. Liu, Q. Liu, Z. Zhu, T. Zhang, and Y. Yang, "MSST-ResNet: deep multi-scale spatiotemporal features for robust visual object tracking," *Knowledge-Based Systems*, vol. 164, no. 15, pp. 235–252, 2019.

[42] J. Zhang, Y. Wu, X. Jin et al., "A fast object tracker based on integrated multiple features and dynamic learning rate," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5986062, 14 pages, 2018.

[43] Y. Huang, T. L. Song, W. J. Lee, and T. Kirubarajan, "Multiple detection joint integrated track splitting for multiple extended target tracking," *Signal Processing*, vol. 162, pp. 126–140, 2019.

[44] Z. Zhou, R. Zhang, and Z. Zhu, "Robust Kalman filtering with long short-term memory for image-based visual servo control," *Multimedia Tools and Applications*, vol. 78, no. 18, pp. 26341–26371, 2019.

[45] Z. Zhou, D. Wu, and Z. Zhu, "Object tracking based on kalman particle filter with LSSVR," *Optik*, vol. 127, no. 2, pp. 613–619, 2016.

[46] S. Chen, S. Zhu, and Y. Yan, "Robust visual tracking via online semi-supervised co-boosting," *Multimedia Systems*, vol. 22, no. 3, pp. 297–313, 2016.

[47] S. Chen, S. Li, S. Su, Q. Tian, and R. Ji, "Online MIL tracking with instance-level semi-supervised learning," *Neurocomputing*, vol. 139, pp. 272–288, 2014.

[48] Z. Zhou, J. Wang, Y. Wang et al., "Visual tracking using improved multiple instance learning with co-training framework for moving robot," *KSII Transactions on Internet & Information Systems*, vol. 12, no. 11, pp. 5496–5521, 2018.

[49] Z. Zhou, X. Gao, J. Xia et al., "Multiple instance learning tracking based on Fisher linear discriminant with incorporated priors," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, 2018.

[50] A. Bertoni and G. Valentini, "Ensembles based on random projections to improve the accuracy of clustering algorithms," in *Neural Nets*, pp. 31–37, Springer, Berlin, Heidelberg, 2005.

[51] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 274–281, Santa Barbara, CA, USA, May 2001.

[52] S. Dasgupta, "Learning mixtures of Gaussians," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 634–644, New York, NY, USA, October 1999.

[53] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[54] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[55] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.

[56] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: a benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2411–2418, Portland, OR, USA, June 2013.

[57] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

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