Drift-Free Tracking Surveillance Based on Online Latent Structured SVM and Kalman Filter Modules

Yung-Yao CHEN†a, Member and Yi-Cheng ZHANG†b, Nonmember

SUMMARY Tracking-by-detection methods consider tracking task as a continuous detection problem applied over video frames. Modern tracking-by-detection trackers have online learning ability; the update stage is essential because it determines how to modify the classifier inherent in a tracker. However, most trackers search for the target within a fixed region centered at the previous object position; thus, they lack spatiotemporal consistency. This becomes a problem when the tracker detects an incorrect object during short-term occlusion. In addition, the scale of the bounding box that contains the target object is usually assumed not to change. This assumption is unrealistic for long-term tracking, where the scale of the target varies as the distance between the target and the camera changes. The accumulation of errors resulting from these shortcomings results in the drift problem, i.e., drifting away from the target object. To resolve this problem, we present a drift-free, online learning-based tracking-by-detection method using a single static camera. We improve the latent structured support vector machine (SVM) tracker by designing a more robust tracker update step by incorporating two Kalman filter modules: the first is used to predict an adaptive search region in consideration of the object motion; the second is used to adjust the scale of the bounding box by accounting for the background model. We propose a hierarchical search strategy that combines Bhattacharyya coefficient similarity analysis and Kalman predictors. This strategy facilitates overcoming occlusion and increases tracking efficiency. We evaluate this work using publicly available videos thoroughly. Experimental results show that the proposed method outperforms the state-of-the-art trackers.

key words: object tracking, support vector machine (SVM), latent structured SVM, online learning, Kalman filter, hierarchical search

1. Introduction

Object tracking is essential in computer vision, and it has wide applications involving intelligent surveillance, human–computer interaction, action recognition, and so on [1]–[4]. However, this task is particularly challenging in real-world scenarios because of several uncontrollable factors, such as random variations in appearance, scale, pose, and illumination. Additional difficulties also arise from fast movement, background clutter and occlusions.

Tracking-by-detection has been widely used in the past decade [5], partially because of the rapid progress in object detection techniques. In some cases, the object model is known prior to tracking. Therefore, the goal of tracking-by-detection methods is to learn a classifier that can distinguish the target from the background, and the tracking task thus becomes repeated detection tasks applied over frames. The classifier of the object model can be learned either by inputting training samples in advance (i.e., model-based tracking) or by referencing a bounding box containing the object given in the first frame of a video (i.e., model-free tracking). However, if the target objects have variations in appearance during tracking, the initial object model may fail to identify them accurately which leads to tracking failure.

To mitigate the aforementioned problem, recent methods involve using online learning techniques that enable the classifier to update its object model rapidly [6], [7]. Specifically, the tracker searches within a local area around the object position detected in the previous frame by using a sliding window approach. Among all candidate object positions, the one that has the maximum classified confidence score is considered the current object position. Once the current position is determined, the online learning mechanism is constructed by automatically generating and labeling new samples around the object position and using them to update the classifier. To increase robustness in updating the classifier when samples are poorly labeled, several schemes are incorporated into the update stage, such as robust loss function [8], multiple instance learning [9], [10], and semi-supervised learning [11].

In this study, we extended the structured SVM tracking (Struck) method [12]. Struck is a tracking-by-detection method that entails training a structured SVM classifier and updating it by using an online latent learning–based scheme called LaRank [13]. The advantage of Struck is model-free tracking that no predefined object appearance model is required. In addition, Struck represents an improvement over the conventional SVM-based tracker because it incorporates a budget maintenance step. In the conventional tracker, as more new samples are accumulated for update during tracking, the unbounded growth of support vectors severely slows the updating of the SVM classifier. The budget maintenance scheme can limit the number of support vectors without substantially degrading the tracking performance. Struck has many advantages, but it still has room for improvement. The main concern is how to enhance the robustness of updating Struck. In Struck, the classifier is updated in each frame, and because of budget maintenance, a new support vector is created while an existing support vector is removed. However, if a false positive detection occurs, such an update impairs the tracker’s discrimination ability. Furthermore, in the next frame, Struck searches only in the area around the previous object position. Therefore, if the
incorrect object detected by the tracker is far from the correct one, the misdetection is unrecoverable, because in the next frame, the target object is outside the tracker’s search area. Although this problem can be solved by using a larger search window, processing more object candidates requires high computation power. Moreover, the occlusion problem remains unsolved in Struck; based on our observation, most false positive detections in Struck arise from temporary occlusion. The accumulation of minor errors results in the drift problem, i.e. the detection result drifting away from the target object.

Occlusion presents a challenge in object tracking because it causes the object appearance model to encounter drastic changes suddenly. To handle occlusion in online learning–based tracking, many recent studies focus on enabling the tracker to learn a partial-occluded object model by utilizing an occlusion detection approach. Zhong et al. [14] proposed an object model generated using a sparsity–based scheme that can detect partial occlusion. A patch of object model is considered to be occluded if the reconstruction error between the patch and its corresponding generative model is greater than a predefined threshold; then, its associated sparsity coefficient is set to zero. Zarezade et al. [15] employed an adaptive Markov model of occlusion to compute a patchwise likelihood measure, from which the probability of patchwise occlusion can be determined. By filtering out patches with a high probability of occlusion, a partial-occluded object model can be learned. Xu et al. [16] combined correlation filter–based tracking with occlusion detection, wherein the object model is first divided into 16 patches and trained individually. The peak-to-sidelobe ratio of each patch response, which measures the strength of a correlation peak, is calculated and is used to predict whether occlusion occurs. Other tracking with occlusion handling methods have been proposed in [17]–[23].

However, we argue that tracking with occlusion detection may be not suitable for Struck for two reasons. First, it is difficult for the tracker to distinguish between partial occlusion and unseen object appearances (e.g., a partially occluded head, and a head in some pose which has never appeared in previous frames). Second, most occlusion detection schemes divide the object model into patches and detect occlusion in individual patches. Simple similarity analysis cannot identify sophisticated occlusion problems; yet, the computation power required for a precise occlusion detection scheme is high and depends on the number of patches.

Figure 1 shows that compared with Struck, the proposed framework is more robust to temporary occlusion and the drift problem. Rather than adding occlusion detection to Struck, we propose integrating a model for predicting object motion into Struck. Most moving objects have distinct motion models. For example, a falling ball cannot rise suddenly, and a fast-driving car cannot stop instantly. Therefore, we apply Kalman filter (KF) [24] in the proposed framework to develop a model for predicting the target object’s motion by accounting for the spatiotemporal information. This motion model is used to reduce false positive detections and to counter occlusion. Because in each frame, the detected object must be one of the object candidates, we prevent false positive detection by employing an efficient object candidate selection scheme.

For fully utilizing the motion model, we propose a hierarchical search strategy that involves combining Bhattacharyya coefficient (BC) similarity [25] and a 2D KF module, which is used to estimate the object position in the subsequent frame. The concept underlying the proposed search strategy is simple: An object candidate that is closer to the position estimate is more likely to be the target object; thus, Struck should start from a small search region near the position estimate (first layer). The structured SVM classifier and Bhattacharyya analysis are used to examine the candidates jointly; if no candidate in the current region is verified as the target, the search region is extended (the next layer), and new object candidates are examined. This hierarchical search is repeated until the search region reaches a predefined maximum. Moreover, unlike conventional Struck, which is updated in each frame, we update the classifier only if the detection result satisfies the criteria of the SVM classifier, BC analysis, and motion model simultaneously. Failure to satisfy these criteria in the current frame implies the occurrence of occlusion, to counter which the predefined maximum search region of the next frame is doubled.

Currently, most surveillance systems are based on a static camera mounted at a fixed position. Because Struck has exhibited high discrimination in detecting objects, we designed an extended Struck framework for static camera–based surveillance that combines two KF modules. In addition to the aforementioned 2D KF module used to estimate the object position (detailed in Sect. 3.2), a 1D KF module is used to adjust the scale of the bounding box by accounting for the background model (detailed in Sects. 3.4 and 3.5). Experimental results demonstrate that the proposed method outperforms not only Struck but also other existing trackers, exhibiting fewer false positive detections and greater robustness against occlusion and requiring less computation time.

The reminder of this paper is organized as follow. Section 2 briefly reviews the related work. In Sect. 3, we present the proposed method which can be viewed as an improved version of the Struck tracker and is designed for the static
camera–based surveillance. In Sect. 4, we provide the experimental results. Finally, Sect. 5 concludes our work.

2. Related Work

Object tracking is widely used in computer vision areas. Because of its importance, many methods have been proposed to solve tracking problems. In this section, we first review some representative tracking methods proposed recently and then provide a general classification for most existing trackers. We also direct the reader to [26]–[28] for a more comprehensive review of tracking techniques.

A. PN Learning–Based Tracking: Kalal et al. [29] proposed a tracking-learning-detection (TLD) method. They developed a PN learning approach to identify the detector’s errors as PN experts, where P-experts estimate missed detections and N-experts estimate false alarms. The PN experts construct a new training set, and the detector is retrained to avoid these errors in the future. The PN learning technique has been adopted in many subsequent tracking methods. Ven et al. [30] proposed a multitarget tracker based on multi-PN learning, in which the dissimilarities and relations of tracked objects are incorporated. Sun et al. [31] combined TLD with KF to enhance the reliability of the TLD approach.

B. Dictionary Learning–Based Tracking: Xing et al. [32] proposed a tracking method that represents the object as dictionaries of templates using a $\ell 1$ tracker. The dictionaries are constructed according to lifespans: short term, medium term, and long term. Such a multififespan dictionary model enables the tracker to flexibly handle various scenarios. Xie et al. [33] proposed a tracking method involving sparse representation and online dictionary learning. They applied local sparse coding with an online updated discriminative dictionary for tracking and used keypoint matching refinement to increase the robustness of the tracker.

C. Compressive Sensing–Based Tracking: Li et al. [34] proposed a tracking method for reducing the computational complexity in the $\ell 1$ tracker. They exploited the signal recovery power of compressive sensing to reduce the dimensionality of the object model. Zhang et al. [35] proposed a real-time compressive tracking method where the object is represented on the basis of the features extracted in the compressive domain, and the restricted isometry property is exploited for reducing dimensionality.

D. Correlation Filter–Based Tracking: Galoogahi et al. [36] proposed using multichannel correlation filters to train a multichannel tracker, reducing both the training time and memory footprint. Using correlation filters in the frequency domain enables the efficient identification of the correlation among training samples. Henriques et al. [37] proposed a tracking method with kernelized correlation filters using circulant matrices. The tracker is based on kernel ridge regression, which can be considered an extension of linear correlation filters to the multichannel domain. Other correlation filter–based tracking methods have been proposed in [38], [39].

E. Classification of Object Tracking Methods: The literature on object tracking is extensive, and many other trackers do not belong to the aforementioned categories. Nevertheless, most tracking algorithms can generally be classified into two categories: generative [40], [41] and discriminative [42], [43].

The goal of generative trackers is to generate a model to represent the target and then use it to search for the most similar image patch with the model. Usually, the minimum reconstruction error is used to represent the similarity. For example, the aforementioned dictionary learning–based trackers can be considered generative trackers because their object models are modeled with a dictionary. Furthermore, $\ell 1$ trackers can be considered generative trackers because the object is represented by a sparse linear combination of templates.

By contrast, the goal of discriminative trackers is to learn a binary classifier that depicts the decision boundary between the target and background. For example, the aforementioned PN learning–based trackers can be considered adaptive discriminative trackers because they require a classifier to discriminate the object from the background and use a PN learning technique to adaptively update the classifier. In addition, most correlation filter–based trackers are discriminative trackers because the objective of using correlation filters is to reduce the redundancy of samples in training a classifier.

According to the preceding definitions, Struck is primarily a discriminative tracker because it contains a structured SVM classifier. However, in the proposed framework, we integrate a generative scheme (i.e., BC analysis) with Struck to improve the robustness against false positive detections and occlusion.

3. Proposed Method: Integrating Struck with Two KF Modules

This section presents the proposed tracking method designed for static camera–based surveillance. Figure 2 shows the overall framework. We improved Struck by designing an efficient object candidate selection scheme that combines a KF-based motion model (Sect. 3.2) and a hierarchical search strategy (Sect. 3.3), and by using a KF-based background modeling scheme to adjust the scale of the bounding box during tracking (Sect. 3.5).

Throughout this paper, we use boldface to indicate a 2D position, vector, or matrix. For example, we use $B^p$ to denote a bounding box (containing the target object) that is centered at position $p$, and we use $x^p \in \chi$ to denote the corresponding feature vector extracted from $B^p$. Moreover, we use the subscript $t$ to indicate the frame number. For example, we use $x^p_t$ to denote the feature vector ($x^p$) at the $t$th frame. In this work, the feature is extracted as the histogram of bounding box.
3.1 Struck Tracker [12]

The goal of object tracking is to identify the object position in the current frame \( p_t \) given the previous position \( p_{t-1} \) or to predict the offset \( y_t \in \psi \) between two consecutive object positions:

\[
y_t = p_t - p_{t-1}.
\] (1)

In Struck, the pair \( (x_{t-1}, y_t) \) is learned jointly by the structured SVM in each frame, where the compatibility between a feature vector \( x \) and an offset \( y \) is estimated using a discriminant function \( f: \chi \times \psi \rightarrow \mathbb{R} \). The offset \( \hat{y} \) in (1) is predicted by determining the maximum compatibility over all possible offsets:

\[
\hat{y}_t = \arg \max_{y \in \psi} f(x_{t-1}, y).
\] (2)

Struck uses a feature map \( \Phi \) to simplify the discriminant function as

\[
f(x, y) = \langle w, \Phi(x, y) \rangle
\] (3)

where \( w \) is a vector normal to the hyperplane. Therefore, (2) can be learned in a large-margin framework by solving the following optimization problem:

\[
\min_w \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \xi_t
\] (4a)

subject to

\[
\begin{align*}
&\forall t; \xi_t \geq 0, \\
&\forall t, y \neq y_t; \\
&\langle w, \Phi(x, y_t) \rangle - \langle w, \Phi(x, y) \rangle \geq \Delta(y_t, y) - \xi_t
\end{align*}
\] (4b)

where \( C \) is a regularization parameter that controls the margin, \( \xi_t \) denotes slack variables, and \( \Delta(y_t, y) \) is the label cost indicating a loss function. Struck then solves (4) by executing an online learning–based SMO-step algorithm [44]. Moreover, a budget maintenance method is employed to prevent the unbounded growth of support vectors in online learning.

Despite being a complete tracking-by-detection model, Struck has unsolved problems. First, in each frame, detection is performed in (2) by maximizing compatibility over all possible offsets in a search space \( \psi \); however, how to address occlusion and how to determine the size of \( \psi \) are not specified. Second, each object candidate corresponds to a possible offset and a bounding box, but the appropriate scale of the bounding box is not specified.

Struck updates its classifier by using new samples from the object candidates in each frame. However, because of the aforementioned problems, such updating is not robust to temporary occlusion; consequently, the drift problem may arise and ultimately lead to loss of tracking.

3.2 2D KF Module for Object Motion Model Prediction

In this work, we incorporate two distinct KF modules into Struck. To prevent confusion, they are referred to as 1) \( 1D \) KF module for real-time background modeling; and 2) \( 2D \) KF module for object motion prediction. We directly implement the KF function of OpenCV library because the code is optimized and has been widely used in many real-time computer vision applications.

KF is a recursive algorithm that infers the estimated state of a dynamic system from previous measurements. KF has four crucial symbols: 1) \( \hat{a}_{t-1} \) is the estimate of state \( a_t \) given measurements \( z_{t-1}, z_{t-2}, \ldots \); 2) \( \hat{a}_0 \) is the corrected \( \hat{a}_{t-1} \) given new measurement \( z_t \); 3) \( P_{t|t-1} \) is the estimate of the error covariance matrix corresponding to state \( a_t \) given measurements \( z_{t-1}, z_{t-2}, \ldots \); and 4) \( P_{f|t} \) is the corrected \( P_{f|t-1} \) given new measurement \( z_t \). The subscript \( t \) denotes the
frame number.

Figure 3 illustrates the framework of KF, which consists of two phases within each recursion and involves using the following matrices.

- Matrix \( F \) is the transition matrix, which converts the previous recursion state to the current recursion state.
- Matrix \( Q \) is the covariance matrix from unknown process noise.
- Matrix \( H \) is the system matrix, relating to the measurement.
- Matrix \( R \) is the noise covariance matrix, relating to the measurement.

The recursive formulas of KF are described as follows. Given \( \hat{a}_{t-1} \) and \( P_{t-1} \) at frame \( t \), the forward state (\( \hat{a}_{t} \)) and forward error covariance (\( P_{t} \)) are predicted as (i) state prediction

\[
\hat{a}_{t} = F \hat{a}_{t-1},
\]

and (ii) covariance prediction

\[
P_{t} = FP_{t-1}F^{T} + Q.
\]

After current measurement \( z_{t} \) is obtained, in preparation for the next recursion, the preceding predicted values are corrected by (iii) gain calculation

\[
K_{t} = P_{t}H^{T}(HP_{t}H^{T} + R)^{-1},
\]

(iv) state correction

\[
\hat{a}_{t} = \hat{a}_{t-1} + K_{t}(z_{t} - H\hat{a}_{t-1}),
\]

and (v) covariance correction

\[
P_{t} = (I - K_{t}H)P_{t-1}.
\]

In the 2D KF module, the object motion model is viewed as the state to be estimated, and the object position detected by Struck is viewed as the measurement \( z_{t} \), which is used to correct the state estimate and modify the motion model constantly by repeating (5) through (9).

Given the previous object position, the current object position can be estimated from (5), and this position estimate provides the best initial guess of the object position; however, the final position is determined by the SVM classifier of Struck, as explained in the following section.

3.3 Hierarchical Search Strategy for Object Candidate Selection

Object candidate selection is crucial to both the detection and update procedures of Struck because it determines all the input samples. Conventional Struck searches for object candidates by using a sliding window with a fixed window size (e.g., 30 pixels) [12]. By contrast, we propose a hierarchical search strategy that involves utilizing the motion prediction (i.e., position estimate) of the 2D KF module.

The architecture of the hierarchical search consists of concentric circles centered at the position estimate [i.e., the dashed circle in Fig. 4(a)]. For implementation, only the search radius of the first frame is fixed as 30 pixels. After the first frame, the maximum search radius is defined as the distance between the previous position and the position estimate [Fig. 4(a)], and because it originates from the 2D KF module, it is referred to as the KF radius (\( KFr \)). For a case where an object stops for a time, the minimum \( KFr \) is set as 6. We divide the KF circle into three nonoverlapping hierarchies and generate three corresponding object candidate pools:

\[
(1^{st} \text{ hierarchy}) \{ y \in Z^{2} : ||y||^{2} \leq (KFr/3)^{2} \}
\]

\[
(2^{nd}) \{ y \in Z^{2} : (KFr/3)^{2} < ||y||^{2} \leq (2KFr/3)^{2} \}
\]

\[
(3^{rd}) \{ y \in Z^{2} : (2KFr/3)^{2} < ||y||^{2} \leq KFr^{2} \}
\]

In Fig. 4(b), the three hierarchies are shaded orange, blue, and red, respectively. Each of them contains several possible offsets. Figure 4(c) provides two examples of the object candidate, which is the bounding box centered at a possible offset. As suggested in [12], we consider only integer-valued offsets.

In Struck, all the object candidates are estimated using (2) simultaneously, and thus the search range cannot be too wide; otherwise, the processing time becomes too long because of the high number of candidates. To mitigate this problem, the proposed framework uses a hierarchical search in which the object candidates are estimated in patches, thus providing more flexibility in adaptively adjusting the search range. This flexibility is achieved by introducing a similarity comparison scheme called BC analysis:

\[
BC(p, q) = \sum_{x=0}^{255} \sqrt{p(x)q(x)},
\]

where \( p \) and \( q \) are the target and candidate histograms, respectively. The BC value ranges from 0 to 1, and a high BC value indicates that the two histograms are highly correlated.

As Fig. 5 shows, a predicted offset being accepted implies no occlusion; thus in the next frame, the 2D KF module moves to the next recursion. Specifically, the position estimate is corrected using (8), error covariance is corrected using (9), and a new position is estimated using (5). In addition, this detection result is used in Struck’s update stage.
Hierarchical architecture. (a) The KF circle centered at the position estimate. (b) Three nonoverlapping regions (i.e., three hierarchies). (c) Example of two object candidates, which are from the first and second hierarchies, respectively. In (c), the white dots indicate all possible offsets.

Hierarchical search strategy, where the thresholds are defined empirically as $BC_1 = 0.85$, $BC_2 = 0.8$, $BC_3 = 0.75$.

Conversely, no predicted offset being accepted implies occlusion; thus in the next frame, the center of the search area stays, 2D KF module remains in the same recursion, and Struck is not updated. In addition, to recover from occlusion, the maximum search radius is expanded to $2KFr$. Specifically, the ranges of the three hierarchies are set as $[0, 2KFr/3]$, $(2KFr/3, 4KFr/3]$, and $(4KFr/3, 2KFr]$, respectively.

Three properties of the proposed hierarchical search are notable. First, it reduces false positive detections and overcomes temporary occlusions by accounting for object motion. Second, if the object does not change its motion, it is likely detected in the first candidate pool; this considerably reduces the processing time because fewer candidates are estimated in comparison with conventional Struck. Third, the three thresholds in Fig. 5 are in descending order $(BC_1 > BC_2 > BC_3)$. Therefore, a predicted offset may not be accepted in the first candidate pool because its BC value is insufficient; however, the same offset may be accepted in the second candidate pool because of the lower threshold.

3.4 1D KF Module for Background Modeling

Scaling is an unavoidable problem because the scale of a target object may vary frame by frame as the object moves, which renders a bounding box with a fixed size inappropriate for tracking. According to [12], to resolve the scaling problem, we can consider a set of scaling factors. However, this increases the number of object candidates, and it is difficult for Struck to distinguish between bounding boxes centered at the same position but with slightly different scales.

Since the proposed framework is designed for static camera–based surveillance, background modeling techniques can facilitate resolving the scaling problem. We adopt the technique described in [45], which is a real-time background modeling scheme with self-update ability. In [45], the background image is modeled without prior knowledge by tracking each pixel’s intensity through 1D KF. The underlying assumption is simple: Typically, the intensity of background pixels does not change quickly, whereas that of foreground objects does. Therefore, the pixels with intensities close to 1D KF predictions in consecutive frames are segmented to the background pixels, and the entire background image is reconstructed by collecting every portion of the background pixels.

3.5 Bounding Box Scale Adjustment

The scheme in [45] entails using the first few frames to reconstruct the entire background image; in this work, we use the first twenty frames. During this procedure, we adopt the method described in [12], i.e. each scaling factor of the set $[0.95, 0.96, \ldots, 1.05]$ is considered to solve the scaling problem.

However, after the 20th frame, we use background subtraction to adjust the scale of the bounding box (Fig. 6). Because in [12], the bounding box is tested at most 5% larger, in this work, the bounding box is first expanded by 5% larger [Fig. 6 (c), left], and the foreground mask [Fig. 6 (c), center], i.e. a binary image containing the pixels belonging to the target object, is generated by subtracting the background image.
Fig. 6  Bounding box scaling. (a) Frame #21, where the object is detected as the red box. (b) Background image for $t = 21$. (c) (from left to right) Expanded box, foreground mask, and minimum bounding box (i.e., scaled box to be updated) after connected component labeling.

[Fig. 6(b)]. The morphological operator closing is applied to this mask to remove the noise-related pixels, and the connected components labeling operator is applied to identify the minimum bounding box [Fig. 6 (c), right], which is used to update the bounding box scale. The minimum permissible scale is set as 0.95; that is, if the scale of the minimum bounding box is less than 0.95 of that of the bounding box in the previous frame, we use the scaling factor of 0.95 to replace it.

We observed that although the background image generated from the method in [45] sometimes has noise pixels [Fig. 6 (b)], it is effective for scale adjustment by expanding the current box followed by background subtraction and the closing operator; most noise pixels are disconnected after the closing operator. In addition, the scale is controlled within the range of $[0.95, 1.05]$ to avoid excessive adjustment. Also for the method in [40], avoid excessive adjustment. Also for the method in [45], when the target object stops for a time, it may be causing background subtraction to fail in adjusting the bounding box scale. This problem is solved by accounting for spatiotemporal information: If the object remains at the same position for several consecutive frames, the background model is not updated until the object moves again.

4. Experimental Results and Discussion

We compared the proposed method with the following state-of-the-art trackers: Struck[12], TLD combined with KF (TLD-KF) [31], real-time compressive tracking (RTCT) [35], and tracking with kernelized correlation filters (KCF) [37]. The original source codes of these trackers were downloaded from online websites [46]–[49]. For fair comparison, the default parameters inherent in the source code or the settings suggested by the authors were employed. All the trackers are implemented in C++ language on the same i7 Quad-Core machine with 3.4 GHz CPU and 4 GB RAM.

Figure 7 shows screenshots of the tracking results obtained using various methods. For evaluation, we collected ten static camera–based videos from publicly available datasets [50], [51]. Table 1 summarizes the features of each video sequence.

4.1 Performance of Our Algorithm

Table 2 presents a comparison of the frame rate (frames per second, FPS) between Struck and the proposed method. The computation speed of the proposed method is higher than that of Struck (7.05-fold on average, red number shown in bottom right of Table 2) because of the proposed hierarchical search strategy.

As described in the second to fifth columns of Table 2, tracking the target in each frame by using the hierarchical search entails four conditions: detection in the first, second, and third hierarchies, and occlusion. For example, the test video PETS09-S2L1 has 400 frames. The first frame is used to define the bounding box of the target. Among the remaining 399 frames, detection is completed in the second hierarchy for 215, detection is completed in the third hierarchy for 135, and occlusion occurs in 23. The blue number in parentheses indicates the percentage relative to the 399 frames.

Notably, in approximately 6% of PETS09-S2L1, occlusion occurs, and the search radius consequently must be expanded. By contrast, in more than 60% of the frames, detection is completed in the first two hierarchies, and thus the tracker need not search the entire KF circle. On average, 81.74% of detections are completed in the first two hierarchies, with 31.06% being completed in the first hierarchy and 50.68% being completed in the second hierarchy (last row of Table 2). These results demonstrate that the proposed hierarchical search strategy enables flexible and adaptive object candidate selection.

Figure 8 illustrates the object’s motion trajectory, where blue squares represent position estimates from the 2D KF module, and red squares represent actual object positions determined by the proposed tracker. Each position estimate is close to the corresponding detected position, indicating that the 2D KF module prediction precisely captures the object’s motion. This observation validates the efficiency of hierarchical search, demonstrating that in most cases, the tracker does not require a large search radius.

Table 3 lists the average frame rates of various
4.2 Comparison with Other Trackers Using Quantitative Measures

We apply two objective metrics to quantitatively evaluate the performance of the five trackers. The first metric is the distance precision (DP) rate, which can be expressed as

$$\text{DP} = \frac{1}{N} \sum_{t=1}^{N} K(\text{CLE}_t, \theta_1),$$  \hspace{1cm} (14)

where $N$ is the total number of frames in a video. The $K$ function $K(\text{CLE}_t, \theta_1)$ is defined as

$$K(\text{CLE}_t, \theta_1) = \begin{cases} 1, & \text{if } \text{CLE}_t \leq \theta_1 \\ 0, & \text{otherwise} \end{cases},$$  \hspace{1cm} (15)

where $\theta_1$ is the distance threshold (set as 15 pixels), and the center location error $\text{CLE}_t = d(C^i_{\text{Tracker}}, C^i_{\text{GT}})$ is the distance between the center of the tracked object $C^i_{\text{Tracker}}$ and the center of the ground truth $C^i_{\text{GT}}$ in the $t$th frame. Therefore, a higher DP rate denotes a more precise tracking result.

The second metric is the overlap success (OS) rate, which can be expressed as

$$\text{OS} = \frac{1}{N} \sum_{t=1}^{N} K(\text{WO}_t, \theta_2),$$  \hspace{1cm} (16)

where $\theta_2$ is the overlap threshold (set as 0.5)
where $\theta_2$ is the overlap threshold (set as 0.5), and the $K$ function is same as (15). The wrong overlap $W_{O_t}$ is defined as the one’s complement of the overlap ratio:

$$W_{O_t} = 1 - \frac{B_{t}^{\text{Tracker}} \cap B_{t}^{\text{GT}}}{B_{t}^{\text{Tracker}} \cup B_{t}^{\text{GT}}},$$

where $B_{t}^{\text{Tracker}}$ represents the tracked bounding box, and $B_{t}^{\text{GT}}$ represents the ground truth bounding box in the $t$th frame. The $W_{O}$ value ranges from zero to one: If the two bounding boxes overlap exactly, $W_{O}$ is zero, and if the two bounding boxes do not overlap, $W_{O}$ is one. Therefore, a higher OS rate denotes a more accurate tracking result. Figure 9 illustrates the concepts of the CLE and overlap ratio.

Tables 4 and 5 present comparisons of the DP and OS results, respectively. As illustrated in the last row of Table 4, in the average DP, the proposed method achieves relative improvements of 39.45% compared with TLD-KF (TLD-KF, 41.92%; proposed method, 81.37%), 32.93% compared with RTCT (RTCT, 48.44%; proposed method, 81.37%), 17.35% compared with KCF (KCF, 64.02%; proposed method, 81.37%), and 31.91% compared with Struck (Struck, 49.46%; proposed method, 81.37%). The proposed method exhibits higher performance than all the other trackers.

As illustrated in the last row of Table 5, in the average OS, the proposed method achieves relative improvements of 43.56% compared with TLD-KF (TLD-KF, 34.58%; proposed method, 78.16%), 32.77% compared with RTCT (RTCT, 45.39%; proposed method, 78.16%), 17.62% compared with KCF (KCF, 60.54%; proposed method, 78.16%), and 41.49% compared with Struck (Struck, 36.67%; proposed method, 78.16%). In most videos, the proposed method attains the highest DP and OS values. These fasts validate the robustness of the proposed method against different challenging video features.

The proposed method improves Struck on integrating the 1D and 2D KF modules into the system, i.e., adding the schemes of bounding box scale adjustment (using 1D KF-based background modeling) and hierarchical search strategy (using 2D KF-based motion prediction). It is worth noting that the contribution of each type of KF modules. Therefore, we also implemented two other methods: 1) Struck with 1D KF module only; and 2) Struck with 2D KF module only. Table 6 presents comparisons of the average frame rate, DP and OS results, respectively.

For the comparison between Struck with/without 1D KF module, when background modeling is integrated, the improvement in frame rate is significantly because it saves the time of considering different scaling factors (Conventional Struck considers eleven scaling factors at each frame). The improvements in DP and OS are also noticeable that the noise of inappropriate box scaling is reduced. For the comparison between Struck with/without 2D KF module, when hierarchical search is integrated, the improvements in frame rate, DP and OS are all significantly. The proposed hierarchical search strategy provides effective object candidate selection in each frame, which reduces false positive detections, prevents drift, and increases tracking efficiency. As detailed in Fig. 2, the proposed method successfully
incorporates two KF modules into Struck. Therefore, it achieves the best performance in each row of Table 6.

4.3 Comparison of Specific Video Features

As Table 1 shows, each test video has distinct challenging features. This section discusses the robustness of the five trackers against specific features.

1) Problems With Scale and Pose Change: We evaluated robustness against variations in scale and pose by using the Walking video. Figure 10 (a) depicts the tracking results. All trackers exhibit high robustness against this problem, except for TLD-KF, which uses an optical flow method for initial detection and drifts away from the object after frame #315. Figure 11 shows plots of the DP and OS under different thresholds for the Walking video. The DP rates [Fig. 11 (a)] indicate that RTCT, KCF, Struck, and the proposed method can precisely track the target throughout the video. However, as evidenced by the OS rates [Fig. 11 (b)], the performance of RTCT, KCF, and Struck degrades because they do not adjust the scale of the bounding box as the proposed method does.

2) Problems With Occlusion: We evaluated robustness against occlusion by using the Walking 2 video. This video also has a scale problem because the woman continually walks away from the camera. However, the more challenging part of this video begins in frame
CHEN and ZHANG: DRIFT-FREE TRACKING SURVEILLANCE BASED ON ONLINE LATENT STRUCTURED SVM AND KALMAN FILTER MODULES

Fig. 13 Comparison of DP and OS plots under different thresholds by using the PETS09-S2L1 video. (a) DP plot. (b) OS plot.

Fig. 14 Comparison of DP and OS plots under different thresholds by using the Lemming video. (a) DP plot. (b) OS plot.

#183 [Fig. 10 (b)], where a man appears from the left side door and occludes the woman several times. The proposed tracker can handle this problem by accounting for BC similarity: When the target object is occluded, the tracker stops moving arbitrarily and updating at that frame. Thus, once the target appears again, our tracker can detect it precisely. Because RTCT, KCF, and Struck do not have a mechanism for overcoming occlusion, they tend to detect the incorrect object, update their classifiers inappropriately, and then drift away.

3) Problems With Deformation and Cluttered: We evaluated robustness against deformation and clutter by using the PETS09-S2L1 video. This video has a clutter problem beginning in frame #25 where many pedestrians with similar clothing gather. The deformation problem results from pose variation. As Fig. 10 (c) shows, all trackers except for the proposed method track the incorrect objects (other pedestrians). The proposed tracker can handle clutter because detection is complete only if it satisfies the criteria of BC similarity, the motion model (i.e., within the KF circle), and the SVM classifier simultaneously. Although other pedestrians have a similar appearance, their motion does not match that of the object. KCF yields the poorest result because it does not have any recovery mechanism; therefore, once it tracks an incorrect object, it loses the target object until the end.

4) Problems With Abrupt Motion: We evaluated robustness against abrupt motion by using the Lemming video. This video is one of the most challenging videos in this study because it also has clutter and occlusion [Fig. 10 (d)]. Moreover, motion blur accompanies abrupt motion. Consequently, at the end of this video, only the proposed tracker precisely tracks the target. Our tracker can overcome the aforementioned problems because the SVM classifier has high discrimination ability and the added KF modules make the update stage robust, thereby preventing false positive detection. TLD-KF, RTCT, and Struck cannot handle abrupt motion because of inappropriate search radius. Initially, KCF exhibits robust tracking; however, when occlusion occurs, it fails to track the object and cannot recover.

5. Conclusion and Future Work

The update stage plays a crucial role in online learning–based trackers; by using each detected object and nearby samples, the tracker can constantly update (modify) its classifier for adaptive tracking. However, false positive detections hinder the robustness of updating, and according to our observations, most false positive detections result from temporary occlusion. Although tracking with occlusion detection is widely employed [14]–[23], it may be not suitable for Struck, as explained in Sect. 1. This paper proposes integrating two KF modules with Struck to make its update stage more robust. This study has four main contributions.

• We demonstrate how to incorporate KF modules into the Struck framework (Fig. 2). In comparisons of DP and OS rates, the proposed method markedly outperforms other state-of-the-art trackers. For all ten test videos, the proposed tracker can precisely track the target to the end without drifting away. These observations validate the robustness of the proposed tracker.

• We propose a hierarchical search that involves using the 2D KF module to effectively select object candidates in each frame. By manipulating the candidate selection approach, we reduce false positive detections substantially and increase tracking efficiency.

• To overcome occlusion, in addition to implementing the SVM classifier inherent to Struck, the proposed method accounts for the object’s motion and executes BC analysis. We limit the updating of Struck to increase the discriminative power of the tracker gained through online learning.

• We propose a bounding box scaling scheme for static camera–based video that involves using the 1D KF module. This scheme reduces the noise that arises from inappropriate box scaling during tracking.

Despite showing many advantages, the proposed approach has several limitations that must be addressed in our future work. First, although the proposed hierarchical search strategy exhibits substantial robustness against temporary occlusion (e.g., Walking 2 and Lemming videos), tests for long-term occlusion have not been conducted. Second, the current framework is designed for static
camera–based surveillance; thus, we can adopt Scott’s technique [45] to model the background in real time and solve the bounding box scaling problem. However, this technique is ineffective in the moving camera case. In addition, even for the static camera case, we observed that if the scene is crowded, the performance of Scott’s technique degrades, with foreground noise increasing in the reconstructed background image. We plan to develop another background modeling scheme and identify a solution for bounding box scaling in moving camera–based tracking.

References

[1] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, SC, USA, June 2000.

[2] S. Noh, J. Park, and Y. Joo, “Intelligent tracking algorithm for maneuvering target using Kalman filter with fuzzy gain,” IET Radar, Sonar & Navigation, vol.1, no.3, pp.241–247, June 2007.

[3] C. Yang, R. Dauriswami, and L. Davis, “Fast multiple object tracking via a hierarchical particle filter,” Proc. IEEE Int. Conf. Computer Vision, Beijing, China, Oct. 2005.

[4] A. Bruhn, J. Weickert, and C. Schnörr, “Lucas/ Kanade Meets Horn/Schunck: combining local and global optic flow methods,” Int. J. Computer Vision, vol.61, no.3, pp.211–231, Feb. 2005.

[5] S. Avidan, “Support vector tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.8, pp.1064–1072, Aug. 2004.

[6] L. Zhang and L. Maaten, “Preserving Structure in Model-Free Tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol.36, no.4, pp.756–769, April 2014.

[7] Z. Wang, H. Wang, J. Tan, P. Chen, and C. Xie, “Robust object tracking via multi-scale patch based sparse coding histogram,” Multimedia Tools and Applications, vol.76, no.10, pp.12181–12203, May 2017.

[8] D. Wang, H. Lu, and M.-H. Yang, “Least soft-threshold squares tracking,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, Portland, OR, USA, pp.2371–2378, June 2013.

[9] 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.

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

[11] K. Zhang, W. Wu, T. Chen, N. Strobelt, and D. Comaniciu, “Robust object tracking using semi-supervised appearance dictionary learning,” Pattern Recognition Letters, vol.62, pp.17–23, Sept. 2015.

[12] S. Hare, S. Golodetz, A. Safarri, V. Vineet, M.-M. Cheng, S.L. Hicks, and P.H.S. Torr, “Struck: structured output tracking with kernels,” IEEE Trans. Pattern Anal. Mach. Intell., vol.38, no.10, pp.2096–2109, Oct. 2016.

[13] A. Bordes, N. Usunier, and L. Bottou, “Sequence labelling SVMs trained in one pass,” Proc. Euro. Conf. Machine Learning and Knowledge Discovery in Databases, Antwerp, Belgium, pp.146–161, Sept. 2008.

[14] W. Zhong, H. Lu, and M.-H. Yang, “Robust object tracking via sparsity-based collaborative model,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, Providence, RI, USA, pp.1838–1845, June 2012.

[15] A. Zarezade, H.R. Rabiee, A. Soltani-Farani, and A. Khajenaghaz, “Patchwise joint sparse tracking with occlusion detection,” IEEE Trans. Image Process., vol.23, no.10, pp.4496–4510, Oct. 2014.

[16] Y. Xu, J. Wang, Y. Li, Z. Miao, M. He, and Y. Zhang, “Scale-adaptive visual tracking with occlusion detection,” Proc. IEEE Int. Conf. Signal Processing, Chengdu, China, pp.938–942, Nov. 2016.

[17] X. Mei, H. Ling, Y. Wu, E.P. 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, July 2013.

[18] J. Shin and D. Kim, “Hybrid approach for facial feature detection and tracking under occlusion,” IEEE Signal Process. Lett., vol.21, no.12, pp.1486–1490, Dec. 2014.

[19] T. Zhang, K. Jia, C. Xu, Y. Ma, and N. Ahuja, “Partial occlusion handling for visual tracking via robust part matching,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, Columbus, OH, USA, pp.1258–1265, June 2014.

[20] C. Sun, D. Wang, and H. Lu, “Occlusion-aware fragment-based tracking with spatial-temporal consistency,” IEEE Trans. Image Process., vol.25, no.8, pp.3814–3825, Aug. 2016.

[21] D. Wang, H. Lu, and M.-H. Yang, “Robust visual tracking via least soft-threshold squares,” IEEE Trans. Circuits Syst. Video Technol., vol.26, no.9, pp.1709–1721, Sept. 2016.

[22] Y. Xu, J. Wang, Y. Li, Z. Miao, and Y. Zhang, “One-step backtracking for occlusion detection in real-time visual tracking,” IEEE Electronics Letters, vol.53, no.5, pp.318–320, March 2017.

[23] H. Fan and H. Ling, “Parallel tracking and verifying: a framework for real-time and high accuracy visual tracking,” Proc. IEEE Int. Conf. Computer Vision, Venice, Italy, Oct. 2017.

[24] E. Kalamn, “A new approach to liner filtering and prediction problems,” Basic Engineering (Series D), vol.82, pp.318–320, 1960.

[25] I.B. Ayed, K. Punithakumar, and S. Li, “Distribution matching with the Bhattacharyya similarity: a bound optimization framework,” IEEE Trans. Pattern Anal. Mach. Intell., vol.37, no.9, pp.1777–1791, Sept. 2015.

[26] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: a benchmark,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, Portland, OR, USA, pp.2411–2418, June 2013.

[27] M. Kristan et al., “The visual object tracking VOT2014 challenge results,” Proc. Euro. Conf. Computer Vision, Zurich, Switzerland, pp.191–217, Sept. 2014.

[28] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” IEEE Trans. Pattern Anal. Mach. Intell., vol.37, no.9, pp.1834–1848, Sept. 2015.

[29] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-Learning-Detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol.34, no.7, pp.1409–1422, July 2012.

[30] J. Ven, A. Kreutzmann, S. Schrader, and F. Dylla, “Multi-PN-learning for tracking applications,” Proc. IEEE Int. Conf. Control Automation Robotics & Vision, Singapore, pp.804–809, Dec. 2013.

[31] C. Sun, S. Zhu, and J. Liu, “Fusing Kalman filter with TLD algorithm for target tracking,” Proc. Chinese Control Conference, Hangzhou, China, pp.3736–3741, July 2015.

[32] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan, “Robust object tracking with online multi-lifestyle dictionary learning,” Proc. IEEE Int. Conf. Computer Vision, Sydney, NSW, Australia, pp.665–672, Dec. 2013.

[33] Y. Xie, W. Zhang, C. Li, S. Lin, Y. Qu, and Y. Zhang, “Discriminative object tracking via sparse representation and online dictionary learning,” IEEE Trans. Cybern., vol.44, no.4, pp.539–553, April 2014.

[34] H. Li, C. Shen, and Q. Shi, “Real-time visual tracking using compressive sensing,” Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, pp.1305–1312, June 2011.

[35] K. Zhang, L. Zhang, and M.-H. Yang, “Real-time compressive tracking,” Proc. Euro. Conf. Computer Vision, Florence, Italy, pp.864–877, Oct. 2012.

[36] H.K. Galoogahi, T. Sim, and S. Lucey, “Multi-channel correlation filters,” Proc. IEEE Int. Conf. Computer Vision, Sydney, NSW, Australia, pp.3072–3079, Dec. 2013.

[37] J.P. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol.37, no.3, pp.583–596, March 2015.
[38] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Hierarchical convolutional features for visual tracking,” Proc. IEEE Int. Conf. Computer Vision, Santiago, Chile, pp.3074–3082, Dec. 2015.

[39] M. Danelljan, G. Bhat, F. Khan, and M. Felsberg, “ECO: efficient convolution operators for tracking,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, Honolulu, Hawaii, USA, July 2017.

[40] D. Riahi and G.-A. Bilodeau, “Multiple object tracking based on sparse generative appearance modeling,” Proc. IEEE Int. Conf. Image Processing, QC, Canada, pp.4017–4021, Sept. 2015.

[41] C. Tian, X. Gao, W. Wei, and H. Zheng, “Visual tracking based on the adaptive color attention tuned sparse generative object model,” IEEE Trans. Image Process., vol.24, no.12, pp.5236–5248, Dec. 2015.

[42] D. Du, L. Zhang, H. Lu, X. Mei, and X. Li, “Discriminative hash tracking with group sparsity,” IEEE Trans. Cybern., vol.46, no.8, pp.1914–1925, Aug. 2016.

[43] S. Duffner and C. Garcia, “Using discriminative motion context for online visual object tracking,” IEEE Trans. Circuits Syst. Video Technol., vol.26, no.12, pp.2215–2225, Dec. 2016.

[44] A. Bordes, L. Bottou, P. Gallinari, and J. Weston, “Solving multi-class support vector machines with LaRank,” Proc. ACM Int. Conf. Machine Learning, Corvalis, Oregon, USA, pp.89–96, June 2007.

[45] J. Scott, M.A. Pusateri, and D. Cornish, “Kalman filter based video background estimation,” Proc. IEEE Applied Imagery Pattern Recognition Workshop, Washington, DC, USA, pp.1–7, Oct. 2009.

[46] http://www.robots.ox.ac.uk/~jsao/circulant/index.html
[47] http://www4.comp.polyu.edu.hk/~cszlzhang/CT/CT.htm
[48] https://github.com/samhare/struck
[49] https://github.com/a1a2y3/rtunTLD/tree/master/runTLD
[50] http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html
[51] https://motchallenge.net/

Yung-Yao Chen received his B.S. (2004) and M.S. (2006) degrees in Electrical and Control Engineering from the National Chiao Tung University from Hsinchu, Taiwan, and his Ph.D. degree in Electrical Engineering from Purdue University, USA. Before joining National Taipei University of Technology, he has worked in HP Labs - Printing and Content Delivery Lab (HPL - PCDL). He is currently an Assistant Professor in the Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei, Taiwan. His current research interests include digital halftoning, computer vision, pattern recognition, color image processing, imaging systems, machine learning, and human-computer interaction.

Yi-Cheng Zhang received his B.S. degree in Electronic Engineering from the Ming Chuan University of Education, Taipei, Taiwan, and his M.S. degree in the Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei, Taiwan. His current research interests include computer vision, image processing, and machine learning.