Multihypothesis Trajectory Analysis for Robust Visual Tracking

Dae-Youn Lee[†], Jae-Young Sim[‡], and Chang-Su Kim[†]
[†]School of Electrical Engineering, Korea University, Seoul, Korea
[‡]School of Electrical and Computer Engineering,
Ulsan National Institute of Science and Technology, Ulsan, Korea
daeyounlee@mcl.korea.ac.kr, jysim@unist.ac.kr, changsukim@korea.ac.kr

Abstract

The notion of multihypothesis trajectory analysis for robust visual tracking is proposed in this work. We employ multiple component trackers using texture, color, and illumination invariant features, respectively. Each component tracker traces a target object forwardly and then backwardly over a time interval. By analyzing the pair of the forward and backward trajectories, we measure the robustness of the component tracker. To this end, we extract the geometry similarity, the cyclic weight, and the appearance similarity from the forward and backward trajectories. We select the optimal component tracker to yield the maximum robustness score, and use its forward trajectory as the final tracking result. Experimental results show that the proposed multihypothesis tracker (MTA) improves the robustness and the accuracy of tracking, outperforming the state-of-the-art trackers on a recent benchmark dataset.

1. Introduction

For decades, numerous visual tracking algorithms have been proposed [30, 29, 20]. Most tracking algorithms experience a drift problem due to various reasons, including non-discriminative feature descriptors, occlusions, and sudden illumination changes. For more robust tracking, recent tracking algorithms attempt to overcome such interferences.

Since a certain feature type may fail to distinguish a target object from its background depending on input video sequences, multiple trackers using different features can be combined adaptively to achieve robust tracking [19, 24, 16, 17]. More specifically, the tracked positions of multiple trackers are compared and processed to yield an overall estimated position. Then, all the trackers are updated using the information at the estimated position. However, if the multiple-tracker algorithm loses the position of a target object in a frame because of any interruption, the tracking error may propagate to future frames.

Recently, tracking systems with memory [23, 21, 31], which can refine past trajectories or appearance models of a tracker, have been proposed to suppress the error propagation. In the long-term trackers in [23, 21], most probable positions of a target object are memorized in each frame, and then the trajectory of the target object is estimated by dynamic programming, which considers both the confidence of each position and the temporal relation between positions in consecutive frames. Also, in [31], several appearance models from past frames are recorded and processed to yield a proper appearance model and reduce tracking errors. However, these trackers with memory employ fixed feature descriptors, which cannot effectively separate a target object from its background in some sequences. Therefore, they still yield inaccurate tracking results depending on the feature selection.

In this paper, we propose a novel multihypothesis tracking algorithm, which combines the concept of the 'tracking using multiple trackers' with that of the 'tracking with memory.' We employ three forward trackers using different features, which are based on texture information, color information, and illumination invariant information, respectively. From frame $t - \tau$ to frame t, each forward tracker traces a target object independently of the other trackers. Then, at frame t, each backward tracker is initialized at the estimated position of the corresponding forward tracker, and then computes a backward trajectory in a time-reversed manner. To select the best tracking result among the three forward trackers, we calculate their robustness scores. To this end, we extract the geometric similarity, the cyclic weight, and the appearance similarity from each pair of the forward and backward trajectories. After selecting the best forward trajectory, the appearance models of all forward trackers revert to the previous conditions at frame $t - \tau$, and are updated using the bounding boxes along the selected forward trajectory. When all forward trajectories have low geometric and appearance similarities in consecutive frames, the forward trackers are not updated and the search range is increased for next frames. The main contributions of this work are as follows.

- Novel multihypothesis trajectory analysis to extract the best trajectory from a set of multiple trackers.
- Design of the robustness score of a pair of forward and backward trajectories, based on the geometric similarity, the cyclic weight, and the appearance similarity.
- Pattern analysis of geometric similarities and appearance similarities along trajectories to detect and handle tracking failures.

The rest of the paper is organized as follows: Section 2 briefly reviews related work. Section 3 explains the STRUCK tracker [10], on which the proposed algorithm is based. Section 4 describes the proposed algorithm, and Section 5 discusses experimental results. Section 6 draws conclusions.

2. Related Work

Tracking-by-Detection: To track a target object by detecting it over time, a support vector machine (SVM) classifier, trained on about $10,000$ images of vehicles and non-vehicles offline, is employed in [1]. Since a target object changes its appearance over time in general, a classifier is updated online using positive and negative samples around tracked target positions in [2]. The STRUCK tracker [10] adopts a structured SVM classifier to integrate learning and tracking harmoniously and avoid ambiguities in sample labelling. Also, to prevent a drift problem due to classifier updates, occlusions are detected by analyzing long-term trajectories of points within a bounding box in [12].
Backward Tracking: In [27], a time-reversibility constraint, based on the geometric similarity, is used to improve the Kanade-Lucas-Tomasi (KLT) feature tracker. The geometric similarity measures the distance between a feature point and its estimated position after the forward and then the backward tracking. In an ideal case, the distance should be zero. The TLD tracker [15] employs several KLT feature trackers, which start from points on a rectangular grid within a bounding box and estimate their motion vectors between consecutive frames. Half of the motion vectors are discarded based on the geometric similarity. Then, the median of the remaining motion vectors becomes the estimated displacement vector of the bounding box. The geometric similarity is also used to detect tracking failures in [25].

The proposed algorithm also performs backward tracking to improve the robustness of forward tracking. However, whereas the conventional algorithms [27, 15, 25] detect tracking failures between two consecutive frames based on the geometric similarity only, the proposed algorithm quantifies the reliability of a forward trajectory by employing the geometric similarity, the circularity, and the appearance similarity between a pair of the forward and backward trajectories over a time interval τ.
Multiple Trackers: The co-tracking algorithm [24] trains multiple SVM classifiers using different feature types and combine their tracking results to achieve robust tracking. In [16], multiple trackers are designed using multiple observation and motion models, and then integrated into a single overall tracker in an interactive Markov Chain Monte Carlo framework. Also, in [17], multiple trackers are adaptively sampled from a tracker space and combined. The unifying algorithm [9] exploits the relation among individual trackers by measuring the consistency of each tracker between two successive frames and the pair-wise correlation among different trackers.
Tracking with Memory: In [23, 21], candidate positions of a target object are recorded for a number of recent frames. Those positions become nodes, which are connected by edges between consecutive frames. Then, an optimal trajectory is extracted by dynamic programming. The MEEM algorithm [31] employs multiple base trackers and memorizes their former states, so that it can restore the tracking process when a tracker has been updated with false positives. Also, the fusion algorithm [4] provides a refined trajectory of an object, by combining the trajectories of the conventional algorithms in a recent benchmark [28], based on dynamic programming.

3. STRUCK Tracker

This section briefly reviews the STRUCK tracker [10]. Let $\mathbf{x}$ denote the position of the bounding box of a target object, and $\mathbf{d}$ denote the displacement vector of the bounding box from previous to current frames. STRUCK uses a discriminant function of the form $f(\mathbf{x}, \mathbf{d}) = \mathbf{w}^t \mathbf{\Phi}(\mathbf{x}, \mathbf{d})$, where $\mathbf{\Phi}(\mathbf{x}, \mathbf{d})$ is a joint feature map of $\mathbf{x}$ and $\mathbf{d}$, and $\mathbf{w}$ is the normal vector to a hyperplane. The discriminant function is simplified to

$$f(\mathbf{x}, \mathbf{d}) = \sum_{i,j} \beta_{i,j} k \left(\mathbf{\Phi}(\mathbf{x}^{(i)}, \mathbf{d}^{(j)}), \mathbf{\Phi}(\mathbf{x}, \mathbf{d}) \right) \qquad (1)$$

where $(\mathbf{x}^{(i)}, \mathbf{d}^{(j)})$ is a support vector, and $k(\cdot)$ is a joint kernel function to transform a linear classifier into a nonlinear one. Also, $\beta_{i,j} > 0$ for a positive support vector and $\beta_{i,j} < 0$ for a negative one.

STRUCK estimates the position $\mathbf{x}_t$ of the bounding box at frame t as $\mathbf{x}_t = \mathbf{x}_{t-1} + \hat{\mathbf{d}}_t$. The estimated displacement vector $\hat{\mathbf{d}}_t$ is given by

$$\hat{\mathbf{d}}_t = \arg\max_{\mathbf{d}_t} f(\mathbf{x}_{t-1}, \mathbf{d}_t), \qquad (2)$$

which maximizes the discriminant function. After estimating $\mathbf{x}_t$, systematically labelled training samples are generated from frame t and used to update the discriminant function, based on the online SVM techniques [5, 6].

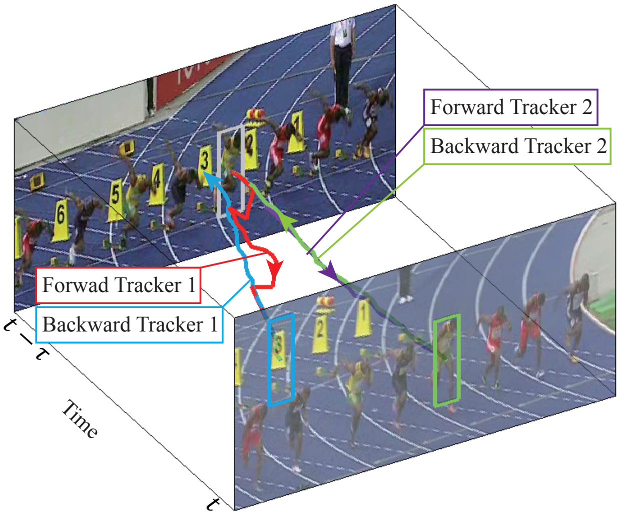


Figure 1. The forward-backward trajectory analysis: the purple forward tracker is successful, while the red one is not.

4. Proposed Multihypothesis Tracker

Tracking may fail, when a target object changes its appearance or is occluded by other objects. For example, suppose that a target object is gradually occluded by a non-target object. Then, the appearance model is also slowly corrupted with the features of the non-target object, and the tracker fails eventually. However, it is difficult to discriminate such gradual corruption of the appearance model from genuine changes in object appearance. To overcome this difficulty, we employ a backward tracker, which detects a specified object in the reverse order of time. We initialize the backward tracker at the detected position of the forward tracker, and obtain the backward trajectory. By comparing the backward trajectory with the forward one, we can tell approximately whether the forward tracker succeeded or not. Moreover, we employ multiple forward trackers that provide multiple trajectory hypotheses. Based on the forward-backward analysis, we select the best forward trajectory to improve the accuracy and the robustness of tracking. Thus, we refer to the proposed algorithm as the multihypothesis tracker (MTA).

Figure 1 illustrates two trajectory hypotheses from frame $t-\tau$ to frame t. In this example, we track the target athlete within the gray bounding box at frame $t-\tau$. Red and purple curves depict the forward trajectories, obtained by two different forward trackers, and blue and green curves are the backward trajectories of the corresponding backward trackers, respectively. Note that the red tracker fails to trace the target, and the associated appearance model is continuously updated with inaccurate samples. At time t, the blue tracker is initialized at the blue bounding box. It then follows the non-target object backwardly in the reverse order of time, and the backward tracking result at time $t-\tau$ is thoroughly different from the target. In contrast, the green tracker provides the result at time $t-\tau$, which matches the original target object successfully. Therefore, we select the purple forward tracker as a valid one and discard the result of the red tracker to achieve robust object tracking.

4.1. Multiple Component Trackers

We employ three component trackers Γ_1, Γ_2, and Γ_3, which are based on STRUCK. These trackers use different feature descriptors, and are employed to determine object trajectories independently. The component trackers exploit the following features:

Γ_1: The first tracker uses a Haar-like feature to represent texture information of a target object. The Haar-like feature consists of six different types at two scales over 4×4 blocks in a bounding box [10]. Thus, the feature vector has the dimension of 192. Each element in the vector is normalized to the range $[-1, 1]$.

Γ_2: The second tracker uses color histograms to consider the local color distribution of the target object. A bounding box is also divided into 4×4 blocks, and the CIE Lab color histogram with 48 bins is extracted from each block. Hence the feature vector has the dimension of 768.

Γ_3: The third tracker employs an illumination invariant feature similarly to [31, 8]. First, we obtain a gradient magnitude image from the L channel image. Then, we obtain a cumulative histogram of the magnitudes. By employing the cumulative histogram, we transform the magnitude image into a rank image, where each magnitude is replaced by the corresponding value of the cumulative histogram. The single rank image and the three-channel CIE Lab image are integrated into a four-channel image. Finally, the 1024-dimensional feature vector is obtained by spatially down-sampling the four-channel bounding box into a 16×16 block.

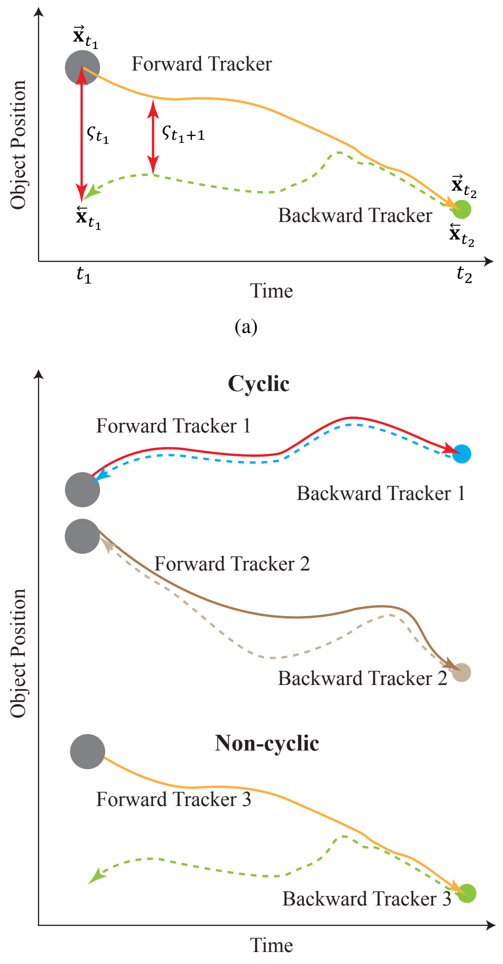
To measure the similarity of feature vectors $\mathbf{u}$ and $\mathbf{v}$, we combine them using the intersection kernel, given by

$$k(\mathbf{u}, \mathbf{v}) = \frac{1}{D} \sum_{i=1}^{D} \min(u_i, v_i) \tag{3}$$

where D is the feature dimension.

4.2. Trajectory Analysis – Robustness Score

The component trackers Γ_1, Γ_2, and Γ_3 yield three trajectory hypotheses. We measure the robustness of each tracker, and choose the trajectory hypothesis of the most robust tracker as the final trajectory.



(a)

(b)

Figure 2. Trajectory analysis: (a) the geometric similarities between a pair of forward and backward trajectories and (b) the cyclic property of a pair of forward and backward trajectories.

Let us describe how to compute the robustness score of a tracker. First, the tracker traces a target object forwardly from previous to current frames. Let $\overrightarrow{\mathbf{x}}_t$ denote the bounding box position at frame t, which is estimated by the tracker in this forward manner. The forward trajectory from frame t_1 to frame t_2 is denoted by

$$\overrightarrow{\mathbf{X}}_{t_1:t_2} = \{\overrightarrow{\mathbf{x}}_{t_1}, \overrightarrow{\mathbf{x}}_{t_1+1}, \ldots, \overrightarrow{\mathbf{x}}_{t_2}\} \tag{4}$$

where $t_1 < t_2$. Then, at the position $\overrightarrow{\mathbf{x}}_{t_2}$ in frame t_2, we initialize the same tracker to trace the target backwardly in the reverse order of time. Let $\overleftarrow{\mathbf{x}}_t$ be the backwardly estimated position at frame t. The backward trajectory from frame t_2 to frame t_1 is denoted by

$$\overleftarrow{\mathbf{X}}_{t_2:t_1} = \{\overleftarrow{\mathbf{x}}_{t_2}, \overleftarrow{\mathbf{x}}_{t_2-1}, \ldots, \overleftarrow{\mathbf{x}}_{t_1}\}\,. \tag{5}$$

Note that, at the last frame t_2 in the interval $[t_1, t_2]$, we have $\overrightarrow{\mathbf{x}}_{t_2} = \overleftarrow{\mathbf{x}}_{t_2}$.

We check the reliability of the forward trajectory using the backward trajectory, by employing three kinds of measurements: geometric similarity, cyclic weight, and appearance similarity. As shown in Figure 2(a), the geometric similarity ς_t at frame t is defined as

$$\varsigma_t = \exp\left(-\frac{||\overrightarrow{\mathbf{x}}_t - \overleftarrow{\mathbf{x}}_t||^2}{\sigma_1^2}\right) \tag{6}$$

using the distance between the forward position $\overrightarrow{\mathbf{x}}_t$ and the backward position $\overleftarrow{\mathbf{x}}_t$, where $\sigma_1^2 = 500$. Ideally, the backward trajectory should be identical with the forward one. In such a case, the geometric similarity ς_t becomes 1.

Next, we evaluate the cyclic weight of the forward and backward trajectories $\overrightarrow{\mathbf{X}}_{t_1:t_2}$ and $\overleftarrow{\mathbf{X}}_{t_2:t_1}$. Although $\overrightarrow{\mathbf{x}}_{t_2} = \overleftarrow{\mathbf{x}}_{t_2}$, the end position of the backward trajectory may be inconsistent with the start position of the forward trajectory, *i.e.* $\overleftarrow{\mathbf{x}}_{t_1} \neq \overrightarrow{\mathbf{x}}_{t_1}$, due to a tracking failure. In such a case, the concatenation of the two trajectories does not form a cycle. In Figure 2(b), trackers 1 and 2 form cycles, while tracker 3 does not. Tracker 1 yields identical forward and backward trajectories, forming a cycle, which indicates a high likelihood of successful tracking. In this case, the geometric similarity ς_t is 1 during the whole interval $[t_1, t_2]$. Tracker 2 also forms a cycle, but the backward trajectory deviates from the forward one in the middle of the interval, in which ς_t decreases. However, this may be due to a temporary occlusion and, overall, tracker 2 may be successful. In contrast, the non-cyclic trajectory of tracker 3 informs of a likely tracking failure.

In practice, we first compute the overlap ratio of two corresponding bounding boxes, given by

$$\zeta_t = \frac{\Delta(\overrightarrow{\mathbf{x}}_t, \overleftarrow{\mathbf{x}}_t)}{\Delta(\overrightarrow{\mathbf{x}}_t) + \Delta(\overleftarrow{\mathbf{x}}_t)} \tag{7}$$

where $\Delta(\overrightarrow{\mathbf{x}}_t)$ and $\Delta(\overleftarrow{\mathbf{x}}_t)$ are the areas of the bounding boxes at $\overrightarrow{\mathbf{x}}_t$ and $\overleftarrow{\mathbf{x}}_t$, respectively, and $\Delta(\overrightarrow{\mathbf{x}}_t, \overleftarrow{\mathbf{x}}_t)$ denotes the overlapped area of the two boxes. When $\zeta_t \leq 0.3$, it is declared that $\overrightarrow{\mathbf{x}}_t$ and $\overleftarrow{\mathbf{x}}_t$ do not match. We count the number Ω of mismatched frames within a short interval $[t_1, t_1 + \epsilon]$ to check whether $\overrightarrow{\mathbf{X}}_{t_1:t_2}$ and $\overleftarrow{\mathbf{X}}_{t_2:t_1}$ form a cycle. Then, we define the cyclic weight χ of $\overrightarrow{\mathbf{X}}_{t_1:t_2}$ and $\overleftarrow{\mathbf{X}}_{t_2:t_1}$ as

$$\chi = \begin{cases} 10^6 & \text{if } \Omega \text{ is 0 or 1,} \\ 1 & \text{otherwise.} \end{cases} \tag{8}$$

Note that 10^6 is an arbitrary big number, so as to set the cyclic weight χ quite differently between the two cases and thus discriminate cyclic trajectories from non-cyclic ones. Also, to compute χ, we consider only the first ϵ frames in the forward trajectory to allow temporary mismatches of

$\overrightarrow{\mathbf{x}}_t$ and $\overleftarrow{\mathbf{x}}_t$ for a short duration within $[t_1, t_2]$. In this work, $\epsilon = 4$ when $t_2 - t_1 = 30$.

Also, we define the appearance similarity to measure the reliability of the backwardly estimated position $\overleftarrow{\mathbf{x}}_t$ in $\overleftarrow{\mathbf{X}}_{t_1:t_2}$. Suppose that the trajectory $\overrightarrow{\mathbf{X}}_{1:t_1}$ from the first frame of an input sequence to frame t_1 is already finalized using the multiple trackers. We maintain a set of four image patches, $\mathcal{S}_{1:t_1}$, which are selected from the bounding boxes along $\overrightarrow{\mathbf{X}}_{1:t_1}$. The bounding box in the first frame is selected by default, and three other boxes are updated to yield the highest discriminant function values up to frame t_1. Let $P(\mathbf{x})$ denote the image patch centered at $\mathbf{x}$. Then, the appearance similarity of $P(\overleftarrow{\mathbf{x}}_t)$ to the set $\mathcal{S}_{1:t_1}$ is defined as

$$\phi_t = \exp\left(-\frac{\sum_{Q\in\mathcal{S}_{1:t_1}} \left\|K \bullet (P(\overleftarrow{\mathbf{x}}_t) - Q))\right\|^2}{4wh\sigma_2^2}\right), \quad (9)$$

where $\sigma_2^2 = 900$, and w and h are the width and height of a bounding box, respectively. K is a Gaussian weight mask, and "$\bullet$" denotes the pixel-by-pixel weight multiplication. A small ϕ_t indicates that the bounding box at $\overleftarrow{\mathbf{x}}_t$ changes its appearance quickly from the previous ones, possibly indicating a tracking error.

Finally, we combine the geometric similarities, the cyclic weight, and the appearance similarities to quantify the robustness of the tracker during $[t_1, t_2]$, given by

$$\Psi_{t_1:t_2} = \chi \sum_{t=t_1}^{t_2} \varsigma_t \phi_t. \quad (10)$$

A large robustness score $\Psi_{t_1:t_2}$ informs that the forward trajectory $\overrightarrow{\mathbf{X}}_{t_1:t_2}$ of the tracker is reliable.

4.3. Tracking by Optimal Trajectory Selection

We perform the tracking by analyzing a pair of forward and backward trajectories, obtained by each component tracker. However, to alleviate the computational burden, we perform the trajectory analysis on each set of τ consecutive frames, but we share the boundary frame between two consecutive sets. More specifically, if the trajectory analysis is performed on the interval $[t - \tau, t]$, it is performed on the next interval $[t, t+\tau]$ by sharing frame t. Therefore, the trajectory analysis routine is called at every τ-th frame only.

For the interval $[t - \tau, t]$, we first obtain the three forward trajectories and then the corresponding backward trajectories, by employing the component trackers Γ_1, Γ_2, and Γ_3, respectively. Then, we compute the robustness scores in (10) for the three trackers, and select the optimal tracker to yield the highest score. Let

$$\overrightarrow{\mathbf{X}}^*_{t-\tau:t} = \left\{\overrightarrow{\mathbf{x}}^*_{t-\tau}, \overrightarrow{\mathbf{x}}^*_{t-\tau+1}, \ldots, \overrightarrow{\mathbf{x}}^*_t\right\} \quad (11)$$

denote the forward trajectory of the optimal tracker. This is used as the tracking result. Then, we revert all component trackers to the previous conditions at frame $t - \tau$, and update their classifiers using the samples along the finalized trajectory $\overrightarrow{\mathbf{X}}^*_{t-\tau:t}$. However, to prevent the use of corrupted samples, we skip the update at frame t if the corresponding product $\varsigma^*_t \phi^*_t$ of the geometric similarity and the appearance similarity is less than a pre-specified threshold $\delta_1 = 0.2$. Then, all three component trackers start at $\overrightarrow{\mathbf{x}}^*_t$ for the next interval $[t, t+\tau]$.

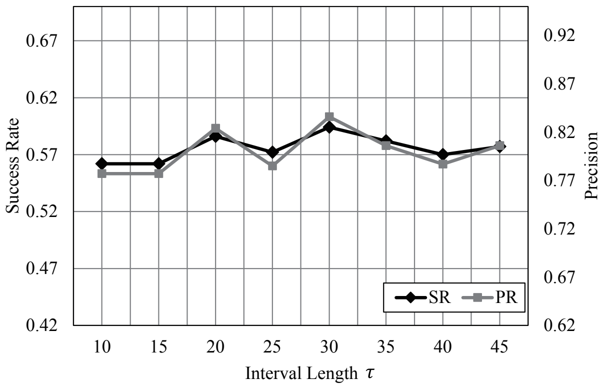


Figure 3. The average success rates (SR) and precision rates (PR) on the benchmark dataset [28], according to the interval length τ.

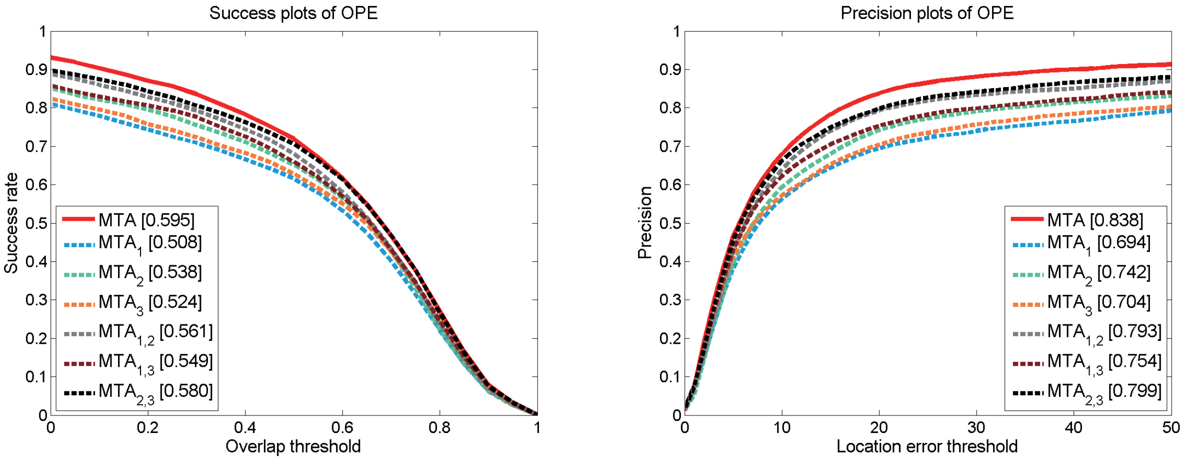
When a non-target object is located near a target object, we cannot select the optimal tracker effectively with a short interval length τ. The interval length should be long enough for the non-target object to be sufficiently separated from the target object. On the other hand, with a longer interval length, the impacts of a single tracking failure becomes more severe. Therefore, it is important to select an appropriate τ. Figure 3 plots the average success rates (SR) and the average precision rates (PR) in terms of τ [28]. SR is the area under the curve of a success plot, and PR is the percentage of the frames in which the estimated positions of a target object are within 20 pixels from the ground truth. Notice that both SR and PR are maximized when $\tau = 30$. We hence set $\tau = 30$.

4.4. Failure Handling

When the cyclic weight in (8) for the optimal tracker is 1, we declare that a tracking failure occurs in the current interval. We also detect a tracking failure, when all component trackers have $\varsigma_t \phi_t \leq \delta_2$ in consecutive frames for a longer duration than $2\tau/3$. In this work, $\delta_2 = 0.004$. When a tracking failure is detected, we do not update the classifiers of all component trackers. Moreover, when a tracking failure occurs, the target object may be located outside the ordinary search range. Thus, we increase the size of the search range from R to $4 \times R$, but check every one out of 64 sample positions within the increased search range to reduce the computational complexity.

Table 1. Comparison of the average success rates (SR) and the average precision rates (PR) on the benchmark sequences in [28]. Eight conventional algorithms are tested: STRUCK [10], SCM [32], TLD [14], ASLA [13], CXT [7], VTD [16], KCF [11], and MEEM [31]. The last four columns correspond to the proposed algorithm: MTA uses all component trackers Γ_1, Γ_2, and Γ_3, while MTA$_i$ uses a single Γ_i only $(i = 1, 2, 3)$. The best result is highlighted in bold face, and the second best result is underlined.

	STRUCK	SCM	TLD	ASLA	CXT	VTD	KCF	MEEM	MTA$_1$	MTA$_2$	MTA$_3$	MTA
SR	0.475	0.499	0.437	0.434	0.426	0.416	0.514	0.579	0.508	0.538	0.524	**0.595**
PR	0.647	0.649	0.608	0.532	0.575	0.576	0.740	0.836	0.694	0.742	0.704	**0.838**



(a)

(b)

Figure 4. The performances of the proposed algorithm using different combinations of component trackers: (a) success plots and (b) precision plots on the benchmark dataset [28]. MTA uses all three combinations, MTA$_i$ means that only Γ_i is used, and MTA$_{i,j}$ means that Γ_i and Γ_j are used.

Table 2. Comparison of the proposed algorithm with the conventional state-of-the-art algorithms on non-benchmark sequences. For each test, the two numbers are the PR(SR) results. The best results are highlighted in bold face.

	STRUCK	KCF	MEEM	MTA
Badminton [18]	0.65(0.46)	0.22(0.15)	0.52(0.37)	**0.89(0.63)**
Bird2 [26]	0.10(0.10)	0.56(0.64)	0.99(0.75)	**1.00(0.77)**
Board [22]	0.68(0.69)	0.70(0.73)	0.60(0.72)	**0.79(0.80)**
GirlMov [26]	0.19(0.18)	0.08(0.08)	0.87(0.63)	**0.92(0.67)**
SnowBoard [18]	0.18(0.17)	0.08(0.10)	0.19(0.14)	**0.41(0.28)**
Surfer [3]	0.97(0.58)	**1.00(0.68)**	0.98(0.62)	0.95(0.50)
Youngki [18]	0.06(0.15)	0.07(0.21)	0.59(0.55)	**0.67(0.60)**
Average	0.40(0.33)	0.39(0.37)	0.68(0.54)	**0.80(0.61)**

5. Experimental Results

We test the proposed algorithm on the recent benchmark dataset [28], which consists of 50 test sequences in challenging conditions, *e.g.* illumination variation, occlusion, and out-of-view. We compare the proposed algorithm with eight conventional trackers: STRUCK [10], SCM [32], TLD [14], ASLA [13], CXT [7], VTD [16], KCF [11], and MEEM [31]. The first six trackers are in the benchmark, while the last two trackers are added because of their excellent tracking results.

Table 1 compares the average SR and PR of the proposed algorithm with those of the conventional algorithms. The proposed algorithm includes the tracking failure handler in Section 4.4 to suppress error propagation and achieve robust tracking. Thus, we perform the one-pass evaluation (OPE) [28] to demonstrate the advantages of the proposed algorithm. In Table 1, the proposed algorithm is tested in four ways: MTA uses all component trackers Γ_1, Γ_2, and Γ_3, while MTA$_i$ uses a single Γ_i only $(i = 1, 2, 3)$. Note that the component trackers are based on STRUCK. However, even though a single component tracker is used, the proposed algorithm outperforms STRUCK due to the tracking failure handler. Moreover, when Γ_1, Γ_2, and Γ_3 are combined based on the multihypothesis trajectory analysis, MTA provides better performance than all the conventional algorithms. Especially, MTA yields 25.3% better SR and 29.5% better PR than STRUCK does. Table 2 shows that the proposed MTA outperforms the state-of-the-art STRUCK, KCF, and MEEM trackers on non-benchmark test sequences as well.

Figure 4 shows the success plots and the precision plots of the proposed algorithm using different combinations of

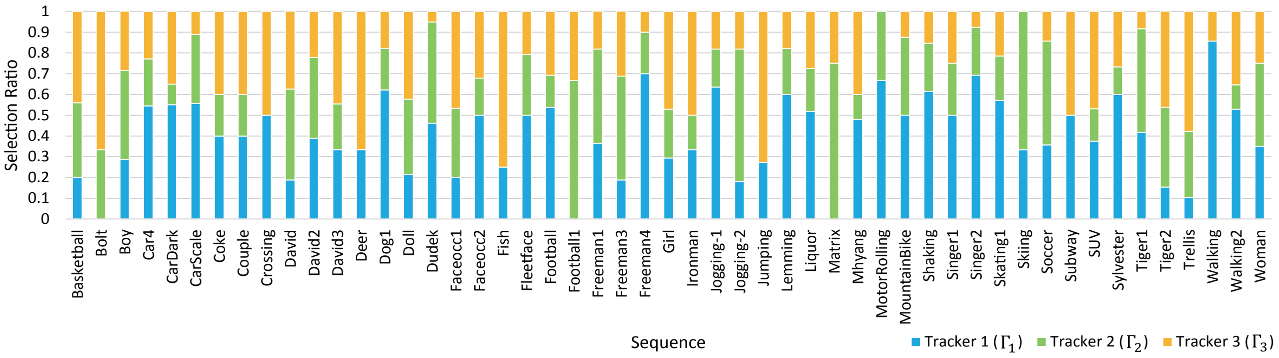


Figure 5. The selection ratios of the component trackers for each test sequence.

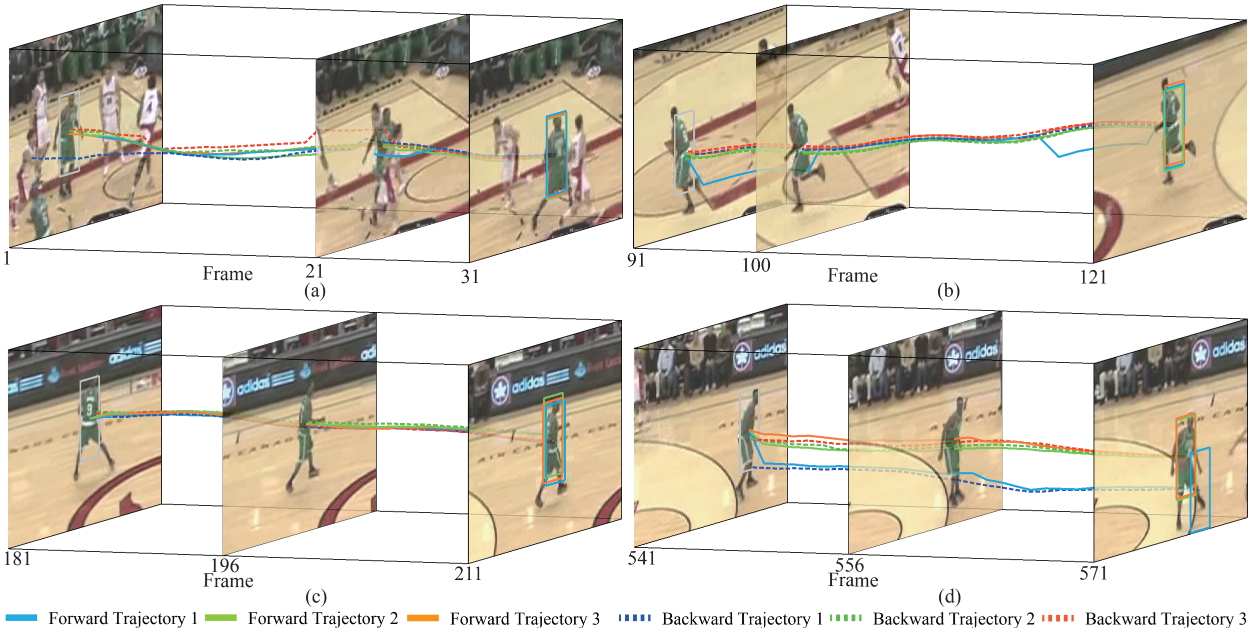
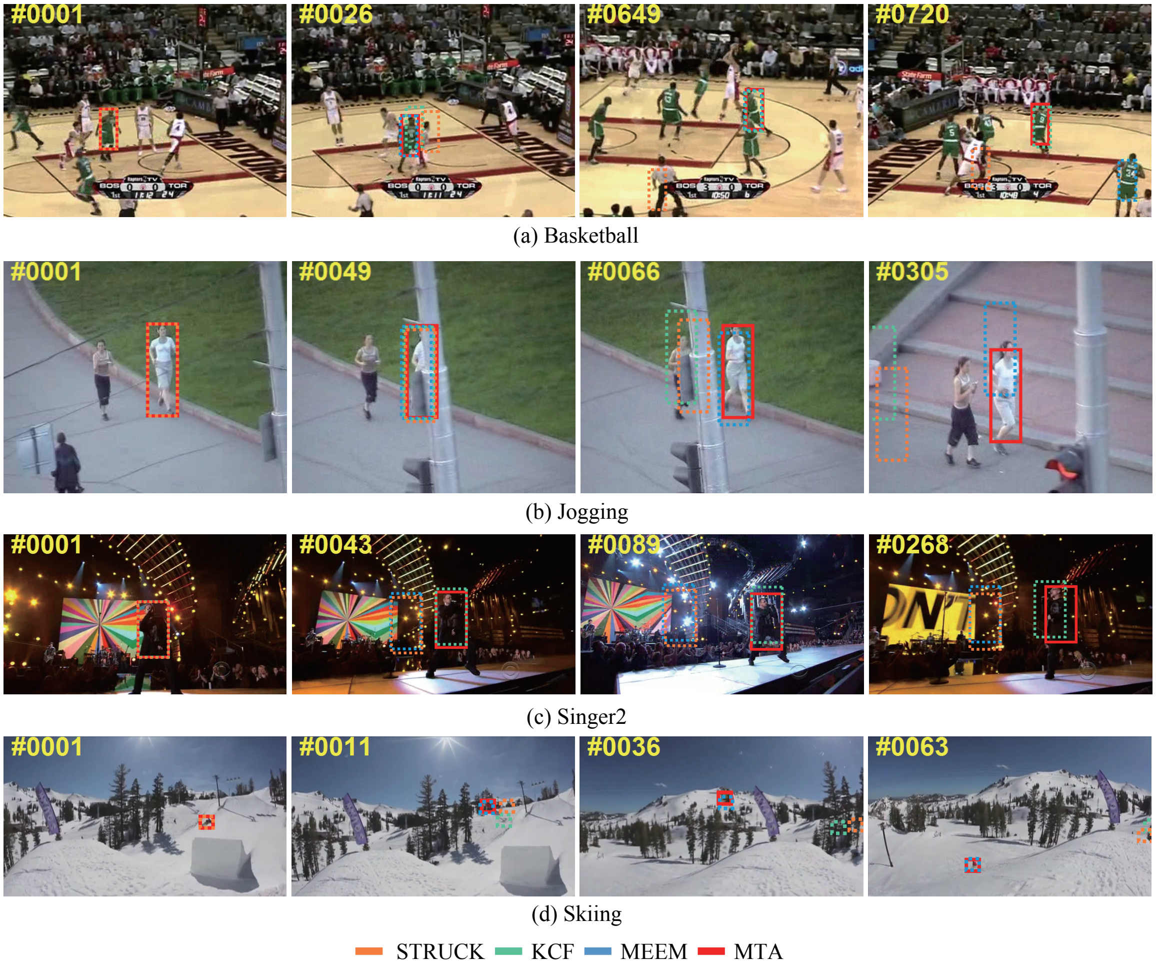


Figure 6. Examples of forward and backward trajectories on the "Basketball" sequence.

component trackers. We test every possible combination. By employing Γ_2, MTA_2 uses color features to provide more accurate tracking results than the other trackers MTA_1 and MTA_3, which use Haar-like and illumination invariant features, respectively. However, when two component trackers Γ_1 and Γ_3 are used, $\text{MTA}_{1,3}$ yields better results than MTA_2. Furthermore, by combining all three Γ_1, Γ_2, and Γ_3, MTA provides the best results. In general, the proposed multihypothesis trajectory analysis provides better results, as it employs more component trackers.

Figure 5 shows the selection ratios of the component trackers Γ_1, Γ_2, and Γ_3 for each sequence. In some sequences, all forward trajectories of a certain tracker are not selected. For example, Γ_1 is not chosen in the "Bolt" sequence, since the Haar-like feature descriptor fails to distinguish the target athlete from the nearby numbered cone, as shown in Figure 1. On average, the selection ratios of Γ_1, Γ_2, and Γ_3 are 41%, 28%, and 31%, respectively. Since each component tracker has its own advantages in different scene scenarios, the selection ratios of the three trackers are similar to one another.

Figure 6 illustrates the tracking process of the proposed MTA on the "Basketball" sequence. Solid and dotted lines depict forward and backward trajectories, respectively. Also, blue, green, and orange colors represent the component trackers Γ_1, Γ_2, and Γ_3, respectively. In this example, the forward trajectories of Γ_2, Γ_3, Γ_1, and Γ_2 are selected as the final ones in Figure 6(a), (b), (c), and (d), respectively. In Figure 6(a), since the target player in the green uniform is partially occluded by another player in the white uni-



(a) Basketball

(b) Jogging

(c) Singer2

(d) Skiing

Figure 7. Tracking examples of the proposed MTA and the conventional STRUCK, KCF, and MEEM trackers.

form at frame 21, the backward trajectory of Γ_1 follows the white player after the occlusion. The backward trajectory of Γ_3 also becomes unstable during the occlusion. Therefore, based on the multihypothesis analysis, the forward trajectory of Γ_2 is selected finally.

Figure 7 compares tracking results of the proposed MTA with those of STRUCK, KCF, and MEEM qualitatively. The "Basketball" and "Jogging" sequences have cluttered backgrounds and occlusions. Therefore, the conventional trackers fail to track the target objects correctly. In contrast, the proposed MTA selects reliable trajectories by exploiting the geometric similarities, the cyclic weights, and the appearance similarities, and tracks the objects successfully. Also, in the "Singer2" and "Skiing" sequences, some features cannot distinguish the targets from the backgrounds. Hence, the conventional trackers suffer from failures depending on the feature selection. However, the proposed MTA robustly tracks the targets using multiple trackers with different features.

6. Conclusions

In this paper, we proposed the multihypothesis trajectory analysis for robust visual tracking. The proposed MTA uses three component trackers. Over a time interval, each component tracker computes the forward trajectory and then the backward trajectory. By analyzing the forward and backward trajectories, MTA extracts the geometry similarities, the cyclic weight, and the appearance similarities, which are then combined into the robustness score of the tracker. MTA then selects the optimal tracker to yield the highest robustness score, and then uses its forward trajectory as the final tracking result. Experimental results demonstrated that the proposed MTA achieves more accurate and robust tracking than the conventional state-of-the-art trackers.

Future research issues include the application of the multihypothesis trajectory analysis to other component trackers. For example, several state-of-the-art trackers can be used as component trackers, while the current implementation uses three component trackers that are all based on STRUCK.

References

[1] S. Avidan. Support vector tracking. *TPAMI*, 26(8):1064–1072, Aug. 2004. 2

[2] S. Avidan. Ensemble tracking. *TPAMI*, 29(2):261–271, Feb. 2007. 2

[3] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *TPAMI*, 33(8):1619–1632, Aug. 2011. 6

[4] C. Bailer, A. Pagani, and D. Stricker. A superior tracking approach: Building a strong tracker through fusion. In *ECCV*, pages 170–185. 2014. 2

[5] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *ICML*, pages 89–96, 2007. 2

[6] A. Bordes, N. Usunier, and L. Bottou. Sequence labelling SVMs trained in one pass. In *ECML-PKDD*, pages 146–161, 2008. 2

[7] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, pages 1177–1184, 2011. 6

[8] U. Fecker, M. Barkowsky, and A. Kaup. Histogram-based prefiltering for luminance and chrominance compensation of multiview video. *TCSVT*, 18(9):1258–1267, Sep. 2008. 3

[9] Y. Gao, R. Ji, L. Zhang, and A. Hauptmann. Symbiotic tracker ensemble toward a unified tracking framework. *TCSVT*, 24(7):1122–1131, Jul. 2014. 2

[10] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, 2011. 2, 3, 6

[11] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. (in press) *TPAMI*, 2015. 6

[12] Y. Hua, K. Alahari, and C. Schmid. Occlusion and motion reasoning for long-term tracking. In *ECCV*, pages 172–187, 2014. 2

[13] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, pages 1822–1829, 2012. 6

[14] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, pages 49–56, 2010. 6

[15] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *TPAMI*, 34(7):1409–1422, Jul. 2012. 2

[16] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010. 1, 2, 6

[17] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *ICCV*, pages 1195–1202, 2011. 1, 2

[18] J. Kwon and K. M. Lee. Wang-Landau Monte Carlo-based tracking methods for abrupt motions. *TPAMI*, 35(4):1011–1024, Apr. 2013. 6

[19] I. Leichter, M. Lindenbaum, and E. Rivlin. A probabilistic framework for combining tracking algorithms. In *CVPR*, pages II–445–II–451 Vol.2, 2004. 1

[20] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. *ACM TIST*, 4(4):58:1–58:48, Oct. 2013. 1

[21] Y. Lu, T. Wu, and S.-C. Zhu. Online object tracking, learning, and parsing with and-or graphs. In *CVPR*, pages 3462–3469, 2014. 1, 2

[22] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In *CVPR*, pages 723–730, 2010. 6

[23] J. Supančič and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013. 1, 2

[24] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *ICCV*, pages 1–8, 2007. 1, 2

[25] L. Wang, H. Yan, H.-Y. Wu, and C. Pan. Forward-backward mean-shift for visual tracking with local-background-weighted histogram. *TITS*, 14(3):1480–1489, Sep. 2013. 2

[26] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *ICCV*, pages 1323–1330, 2011. 6

[27] H. Wu, R. Chellappa, A. Sankaranarayanan, and S. Zhou. Robust visual tracking using the time-reversibility constraint. In *ICCV*, pages 1–8, 2007. 2

[28] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418, 2013. 2, 5, 6

[29] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823 – 3831, 2011. 1

[30] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM CSUR*, 38(4), Dec. 2006. 1

[31] J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *ECCV*, pages 188–203, 2014. 1, 2, 3, 6

[32] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845, 2012. 6