Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2018: 1–15
© The Author(s) 2018
DOI: 10.1177/1729881418756238
journals.sagepub.com/home/arx
SAGE

Visual tracking via improving motion model and model updater

Wanli Xue[1], Zhiyong Feng[2], Chao Xu[2], Tong Liu[2], Zhaopeng Meng[2] and Chengwei Zhang[1]

Abstract

Motion model and model updater are two necessary components for online visual tracking. On the one hand, an effective motion model needs to strike the right balance between target processing, to account for the target appearance and scene analysis, and to describe stable background information. Most conventional trackers focus on one aspect out of the two and hence are not able to achieve the correct balance. On the other hand, the admirable model update needs to consider both the tracking speed and the model drift. Most tracking models are updated on every frame or fixed frames, so it cannot achieve the best performance. In this article, we solve the motion model problem by collaboratively using salient region detection and image segmentation. Particularly, the two methods are for different purposes. In the absence of prior knowledge, the former considers image attributes like color, gradient, edges, and boundaries then forms a robust object; the latter aggregates individual pixels into meaningful atomic regions by using the prior knowledge of target and background in the video sequence. Taking advantage of their complementary roles, we construct a more reasonable confidence map. For model update problems, we dynamically update the model by analyzing scene with image similarity, which not only reduces the update frequency of the model but also suppresses the model drift. Finally, we use these improved building blocks not only to do comparative tests but also to give a basic tracker, and extensive experimental results on OTB50 show that the proposed methods perform favorably against the state-of-the-art methods.

Keywords

Visual tracking, motion model, model updater, image segmentation, saliency detection, image similarity

Date received: 22 August 2017; accepted: 14 December 2017

Topic: Special Issue—3D Vision for Robot Perception
Topic Editor: Antonio Fernandez-Caballero
Associate Editor: Shengyong Chen

Introduction

Visual object tracking is an important problem in computer vision.[1–6] It is a task that estimates the trajectory of a target in a video sequence.[7] The tracker has no prior knowledge of the target to be tracked such as category and shape. Despite extensive research on visual tracking, it remains challenging problems in handling complex target appearance changes caused by pose, occlusion (OCC), illumination, and motion.

An important insight is that the tracking problem should be considered as a balance between the scene and the target. Specifically, it is not only accurate description of the target appearance but also a comprehensive access to the scene information. Another important vision is that the video is highly relevant but redundant. And, finding

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
[2] School of Computer Software, Tianjin University, Tianjin, China

Corresponding author:
Chao Xu, School of Computer Software, Tianjin University, Tianjin 300354, China.
Email: xuchao@tju.edu.cn

another balance between the relevant and redundant is important for improving tracking speed and accuracy.

How to find these balances of the above issues? We need to think of the tracking problem itself. The simplest way is to observe it through the bioinspired perspective. When people track a target, their eyes will pay more attention to the target itself, and for the surrounding environment, they will have a general division. For example, when we track a pedestrian, we will be more concerned about his characteristics (such as the color of clothes, the size of the body, etc.), and for the surrounding environment, we generally only need to know some rough scenes (such as tall buildings, trees, etc.). In the process of tracking, we can feel that we are not very concerned about every moment because the brain will be very clever to remove some of the approximate fragment, and focus on the target and scene changes.

Based on these intuitions, we know that we should solve the sample selection problem from the perspective of spatial dimension and tackle the tracking process problem from the perspective of time dimension. And according to the research of Wang et al., the tracker is composed of several modules: motion model, feature extractor, observation model, model updater, and ensemble post-processor.[8] In particular, motion model and model updater (MMMU) contain many details that can affect the tracking result, but they are rarely considered in most of the tracking methods. Coincidentally, our above ideas can be used to enrich these two components and thence this article will focus on them.

First, the motion model generates object proposals, it samples from the raw input image to forecast the possible candidate locations so as to confirm the scope of target searching. An effective sample selection mechanism can provide high-quality training samples which make the tracker recovers from failure and estimates appearance changes accurately. Hence, it is important to get more accurate samples in motion model. We develop a collaborative method based on image segmentation and salient region detection to analyze the appearance samples, the former is used to obtain more comprehensive scene information and the latter is used to find more accurate target samples. This method differs significantly from the existing motion model, such as the sliding window, which is prone to drifting in fast motion (FM) or large deformation (DEF) video. Specifically, we employ simple linear iterative clustering (SLIC) algorithm for image segmentation[9] and exploit frequency-tuned saliency analysis algorithm (FT) for salient region detection.[10]

Second, it is critical to enhance the model updater of a tracker that adopts tracking-by-detection approach. Most of the tracking methods update the observation model in each frame, it reduces efficiency and more critical is that poor tracking results can cause the classifier to be contaminated, thus causing drift. Different from other tracking paradigms that update model in a fixed manner such as updated every two frames, we formulate a simple and quick method to update observation model dynamically with image similarity. It not only improves the tracking speed but also increases the accuracy. We will introduce a perceptual hash (pHash) algorithm[11] in detail for image similarity.[12]

The overview of our approach is illustrated in Figure 1. The original image is subjected to image segmentation processing and salient region detection, respectively, in motion model. Followed by cooperative learning, the processed information will be sent to the tracking framework. After getting the tracking result, the current scene and the estimated target will be compared with their own recent stable values to determine whether to update the classifier in model updater.

In summary, our main contribution is to address the traditional tracking problems by effectively ameliorating the MMMU:

1. In order to make more rational use of the visual properties of the image, image segmentation is used to obtain more meaningful atomic regions in the field of color; salient region detection is used to describe human's visual attention mechanism which involves distance, color, intensity, and texture. We use both methods to handle tracking scenes and targets in motion model thus achieving a more balanced appearance for visual tracking.
2. We propose a novel method to determine whether the estimated target is reliable in the time dimension and make a decision whether to update the observation model by using the hash of image similarity.

Then, we have validated the two components separately (CT Tracker[13] with improved motion model and CF2 Tracker[14] with improved model updater) and designed a basic tracker named MMMU tracking. Experimental results on OTB50[15] show that the improved components are valid and our basic tracker performs favorably against most of the state-of-the-art methods.

Related work

Most trackers use statistical learning techniques to take charge of constructing robust object descriptors and building effective mathematical models for target identification.[16–23] As estimated object position is converted into labeled samples, it is hard to give the accurate estimation of the object position. Wang et al. used an inverse sparse representation formulation and a locally weighted distance metric to propose a sparsity-based tracking algorithm.[24] Hare et al.[25] integrate the labeling positive and negative samples procedure into the learner by using online kernelized structured output support vector machine (SO-SVM; Struck). Choi et al. exploited an arbitration algorithm between a finite impulse response (FIR) filter and optical flow by adaptive neuro-fuzzy inference system for tracking research.[26] And there are also many tracking algorithms[27]

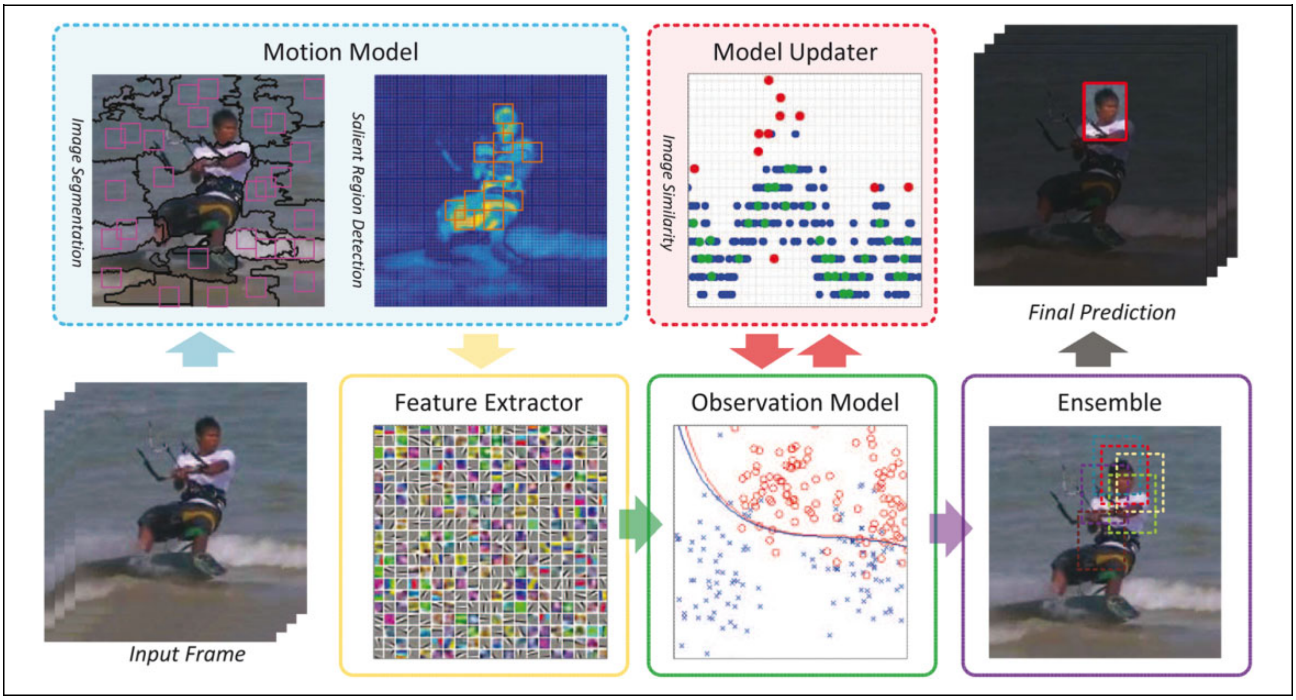


Figure 1. The overview of our MMMU tracking. By improving the motion model, we employ image segmentation and salient region detection methods for obtaining comprehensive and accurate sample set. The target samples are in the orange rectangle box, and the scene samples are in the purple rectangle box. In the model update component, we use image similarity method to get very similar frames (the green dots) and large difference frames (the red dots). By reducing the update of these frames, the tracking method can achieve better speed and accuracy. MMMU: motion model and model updater.

that focus on appearance and motion model definition to deal with the complex scene and avoid drifting.

Compressed sensing theory is introduced into visual object tracking by Zhang et al. CT extracts Haar-like features in the compressed domain as the input characteristics to the classifier. It aims to design an effective appearance model and first compresses sample images of the foreground target and the background using the same sparse measurement matrix to efficiently extract the low-dimensional object descriptors.[13]

In general, targets are chosen as the positive samples to update the classifier, and low correlation samples may often be included since they are not correct enough, which causes the failure of the updating of the classifier.[28] Therefore, sample selection is an important task for alleviating drift in the motion model. Additionally, massive amounts of training samples would hinder the online updating of the classifier without an appropriate sample selection strategy. Liu et al. design a sparsity-constrained sample selection strategy to choose some representative support samples from a large number of training samples on the updating stage.[29] It is necessary to integrate the samples contribution into the optimization procedure when observing the appearance of the target.[30]

Most discriminative trackers apply continuous learning strategy, where the observation model is updated rigorously in every frame. Research results show that excessive update strategy will lead to both lower frame rates and degradation of robustness because of over-fitting in the recent frames.[31,32] So we refine the strategy of model updater by analyzing the stability of scene.

Our approach

Different from most of the appearance-based detection methods,[33] we solve the robustness tracking problem by improving the model updater and the motion model.

Motion model improvement

Image segmentation. In order to obtain a comprehensive scene sample, a simple way is to gather the image scene into several blocks according to a certain rule and then select the appropriate sample on each block to ensure the comprehensiveness of the sample set. We use image segmentation to solve this problem.

Image segmentation process clusters pixels by the similarity of their feature and divides the raw image into several specific regions that may correspond to the tracked object. Superpixel is a kind of image segmentation algorithm, which provides a convenient primitive to compute local image features. SLIC is a popular superpixel algorithm which is fast, easy to use, and produces high-quality segmentations.[9]

We segment the first frame into N superpixels. A color histogram is extracted as the feature vector f_i for each superpixel $\mathrm{sp}(i)(i = 1, \ldots, N)$. We choose mean shift to cluster these superpixels. We can obtain n_{sp} $(n_{\mathrm{sp}} < N)$ different clusters and each cluster center $f_c(j)(j = 1, \ldots, n)$ by employing mean shift algorithm on the feature pool $F = \{f_i | i = 1, \ldots, N\}$.

Each cluster $\mathrm{clst}(j)(j = 1, \ldots, n_{\mathrm{sp}})$ is the set that includes its own cluster members $\{f_i | f_i \in \mathrm{clst}(j)\}$. Therefore, each cluster is represented by $f_c(j)$ and $\mathrm{clst}(j)$ in the feature space. The members of each cluster $\mathrm{clst}(j)$ correspond to different superpixels in the image region. The weight $w(j)(j = 1, \ldots, n_{\mathrm{sp}})$ is assigned to each cluster center $f_c(j)$ by exploiting a prior knowledge of the targets bounding box in the first frame, which indicates the likelihood that superpixel members of $\mathrm{clst}(j)$ belong to the target area. We count two scores for each cluster $\mathrm{clst}(j)$: $s^+(j)$ and $s^-(j)$.[9] The former denotes the size of the overlapping area that all superpixel members of each cluster $\mathrm{clst}(j)$ cover the bounding box, and accordingly the latter denotes the size of all superpixel members outside the target area. The weight is normalized between -1 and 1 and calculated as follows

$$w(j) = \frac{s^+(j) - s^-(j)}{s^+(j) + s^-(j)}, \forall j = 1, \ldots, n_{\mathrm{sp}} \tag{1}$$

And positive values indicate high confidence to assign $f_c(j)$ to target. To obtain a confidence map for the t th frame, we first segment a surrounding region of the target into n_{sp} superpixels and then compute every superpixel confidence value. The surrounding region of the target is a square area, and its side length is $\eta\sqrt{S}$, where η is the constant parameter to control the size of this surrounding area and S is the area size of the target. The confidence value of a superpixel depends on two factors: the distance between this superpixel and the cluster center in the feature space, and the weight of the corresponding cluster center.[9] $C_t(i)$ is the confidence value for superpixel i at the t th frame.[9] For $\forall i = 1, \ldots, n_{\mathrm{sp}}$, the confidence value of each superpixel is computed as follows

$$C_t(i) = \mathrm{argmax}_{1 \leq j \leq n}\{e^{||f_t(i) - f_c(j)||} \times w(j)\} \tag{2}$$

where $f_t(i)$ and $f_c(j)$ denote the feature vector of the i th superpixel in the t th frame and the j th cluster center in the first frame, respectively. Intuitively, the nearer the feature of a superpixel $f_t(i)$ is close to the targets cluster center $f_c(j)$, the more likely this superpixel belongs to the target area.

In the t th frame, each pixel shares the same confidence value $C_t(i)$. The surrounding area of the target is scanned with a sliding window that has the same size as the bounding box. At each position, the sum of the confidence value in this sliding window is computed, which demonstrates evidence for separating the target from the background. Then, the location of sliding window with the maximum value response will be selected as the new candidate location.[9]

From the visualization in Figure 2, the objects with the same attributes in the scene are aggregated together, such as lawns, shrubs, leaves, and "baby monster." Thus, we can select the comprehensive scene samples easily. Although this method can obtain image regions with a certain common trait, and these regions can provide comprehensive and reasonable scene samples, but it is not suitable for dividing the target area. For example, it often mixes some parts from the scene and target. Therefore, we need other way to solve the target sample selection.

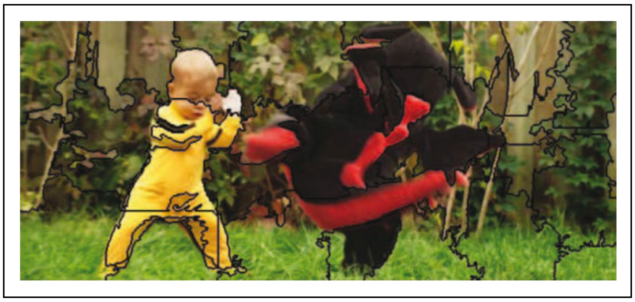


Figure 2. *Dragon baby* processed by SLIC. SLIC: simple linear iterative clustering.

Salient region detection. Through the above analysis, the superpixels result is not stable, it only provides a coarse over-segmented image. So, we need an effective method to focus on the target appearance. And, saliency is intentionally regarded as visual attention, and it is determined as the local contrast of an image region with respect to its neighborhood.[10] The study of saliency detection comes from biological research. It is utilized to interpret complex scenes now. Scene analysis technique is integrated into visual tracking pipeline, which will significantly improve the performance, because it can separate the target from the background using high-quality saliency maps.

Frequency-tuned saliency analysis algorithm (FT) method can emphasize the largest salient objects and uniformly highlight whole salient regions.[10] The frequency-tuned saliency analysis is formulated as follows

$$S_t(x,y) = ||I_\mu - I_{\omega hc}(x,y)|| \tag{3}$$

where I_μ is the mean image feature vector of color and luminance, $I_{\omega hc}(x,y)$ is the corresponding image pixel vector value in the Gaussian blurred version of the original image (after testing, we used a 5×5 separable binomial kernel), $|| \cdot ||$ is the L2 norm, and the color space is LAB.

As shown in Figure 3, we can intuitively observe that the kite surfer's pants receives the maximum response, and his body gets the maximum response, and legs and torso also have the high correlation. It basically constitutes the target. It is worth noting that the spray and the water are well distinguished, and these meaningful distinctions help us to choose samples exactly.

After analyzing the two components, we need to combine the motion model with the model updater.

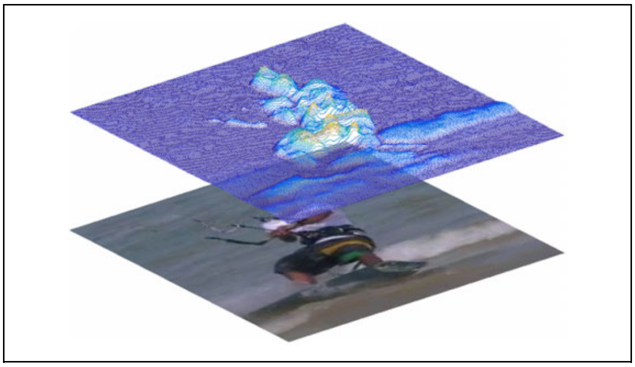


Figure 3. After processing by saliency analysis algorithm, the target becomes more pronounced. And it is easier to obtain accurate target samples.

To select high-quality samples, we construct a target–background confidence map according to the similarity of superpixels in the surrounding region of the target between new frame and the first frame. Then it is refined by salient region detection result, and the confidence map can facilitate tracker to distinguish the target and the background accurately.[34] Finally, to accelerate the tracker, we control the model updater by judging the stability of scene and computing image similarity between frames.

As shown in Figure 2, the result of superpixels is not stable. It only provides a coarse over-segmented image. To get the likelihood that superpixel members whether belong to the target area, we still need the prior knowledge of the targets bounding box in the first frame. Figure 3 shows that salient region detection provides the probability of each pixel belonging to the foreground target, the result can be used to refine confidence map, we know that each pixel of the i th superpixel in the t th frame shares the same confidence value $C_t(i)$, and each pixel has a sailency value in map S_t. Therefore, the fusion weight $C\,\mathrm{map}_t$ can be formulated as follows

$$C\,\mathrm{map}_t = \phi w_{\mathrm{ct}}(i) + (1-\phi) w_{\mathrm{st}} \tag{4}$$

where $w_{\mathrm{ct}}(i)$ and w_{st} are the min–max normalization of $C_t(i)$ and S_t, respectively; here, we use a simple linear fusion method. After several tests, we determined $\phi = 0.5$.

Model updater improvement

Image *similarity.* We have integrated superpixels segmentation and salient region detection into motion model, and this procedure improves the performance of the base model. However, there is a computational overhead, which will slow down the base model. And another obvious fact is that update classifier frame-by-frame not only reduces the tracking speed but also may "pollute" the classifier.

Algorithm 1. pHash algorithm.

```
1. Resize the input image to 32 × 32;
2. Reduce the image to grayscale;
3. Separate the image into a collection of frequencies and scalars;
4. Get the lowest frequencies that just keep the top-left 8 × 8 of
   the above output;
5. Calculate the average of the output in step 4, and set the 64
   hash bits by this strategy: If the value is less than the average,
   the hash bits of the corresponding position will be set to 0,
   otherwise it is set to 1;
6. Construct the 64 bits into a vector.
```

So we refine the strategy of model update to accelerate our tracker, the classifier will only be updated when the scene is not stable (background significantly changes). We analyze the stability of scene by comparing the similarity of incoming frame with the previous frames. Here, we use a pHash algorithm[11] to get the fingerprints of images which has several properties: Images can be scaled larger or smaller, have different aspect ratios, even minor coloring differences.[35]

But, these images will still match similar images. The fingerprint result will not vary as long as the overall structure of the image remains the same.[35] The main steps of pHash are summarized in algorithm 1, and we can compare the difference of images by computing hamming distance of their hash vector.

We use *Basketball* sequence to show the advantages of pHash in achieving stable scenes and target. This sequence mainly describes the Celtics player Rajon Rondo's defense process in an NBA game. As illustrated in Figure 4, Rondo's entire defense process is divided into 11 scenes by his movements such as turning, jogging, running, landing, defense, and so on. Take scene 3 (about from frames #140 to #180) in Figure 4 as an example, when the opponent took the ball and ready to attack, Rondo quickly ran back to the defensive position. For easy observation, we select the scene range which is twice the target size and has the same center. We plot the Hamming distance of each frame with its previous two frames by using pHash, as shown in Figure 5; we find that the distribution of these data is consistent with the analysis of the video sequence changes in Figure 4. In particular, some data anomalies such as frames #650, #678, and #707. Besides the shortcomings of the algorithm, we know that this phenomenon occurs because the color and shape of the image have undergone great changes, as illustrated in Figure 4 "specialness."

Update strategy. Let h_t represents the image's hash vector calculated by algorithm 1 in frame t, and p_t denotes the hamming distance of current frame with its previous two frames, it is predicted by

$$p_t = |h_t - \frac{h_{t-1} + h_{t-2}}{2}|, t \geq 3 \tag{5}$$

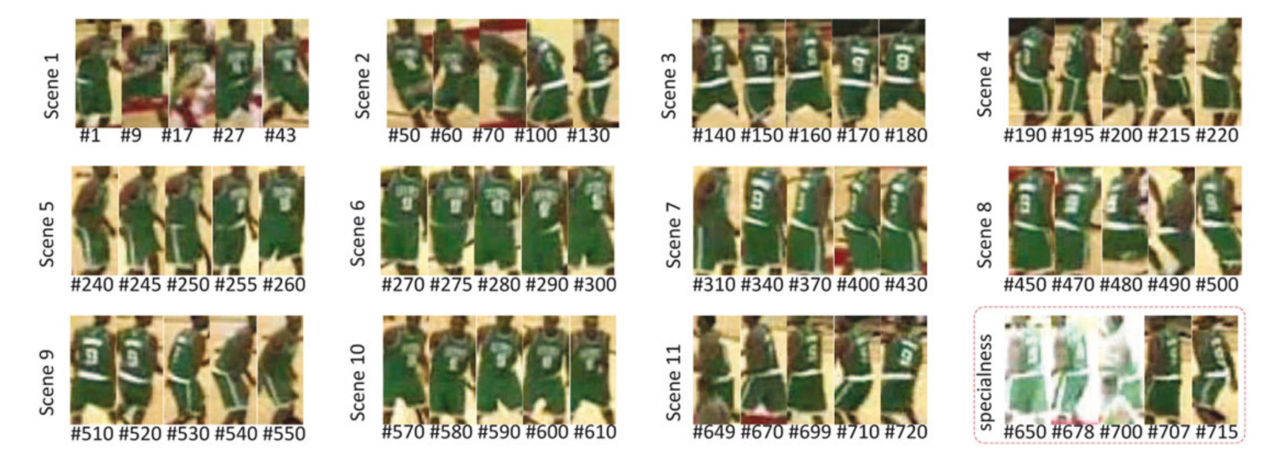


Figure 4. Several major changes to the target in *Basketball* sequence.

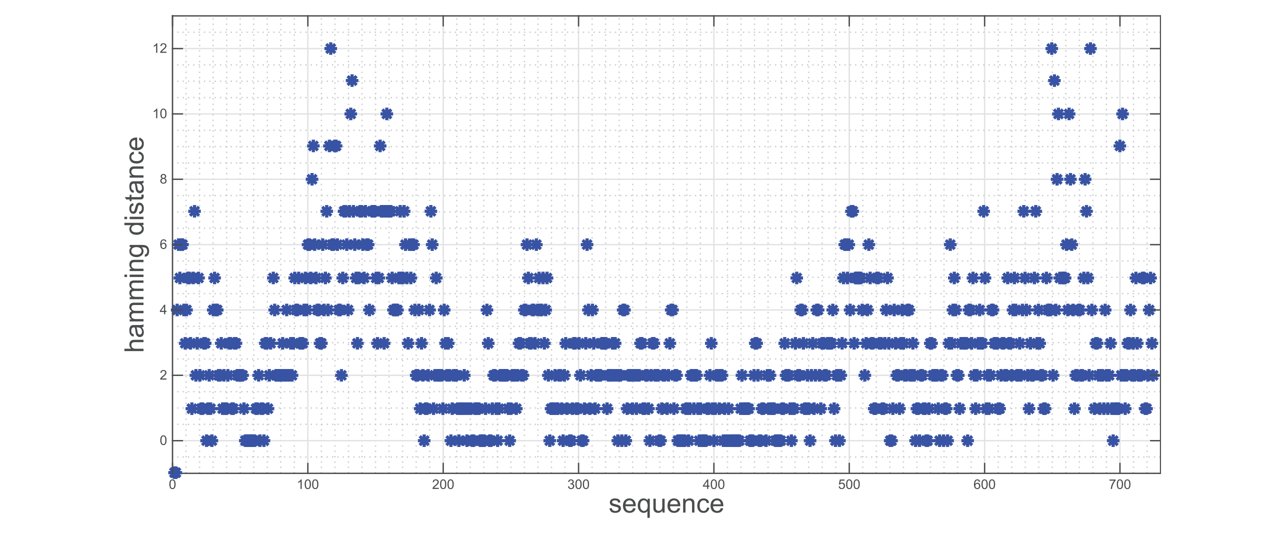


Figure 5. Hamming distance of each frame's scene with its previous two frames in *Basketball* sequence.

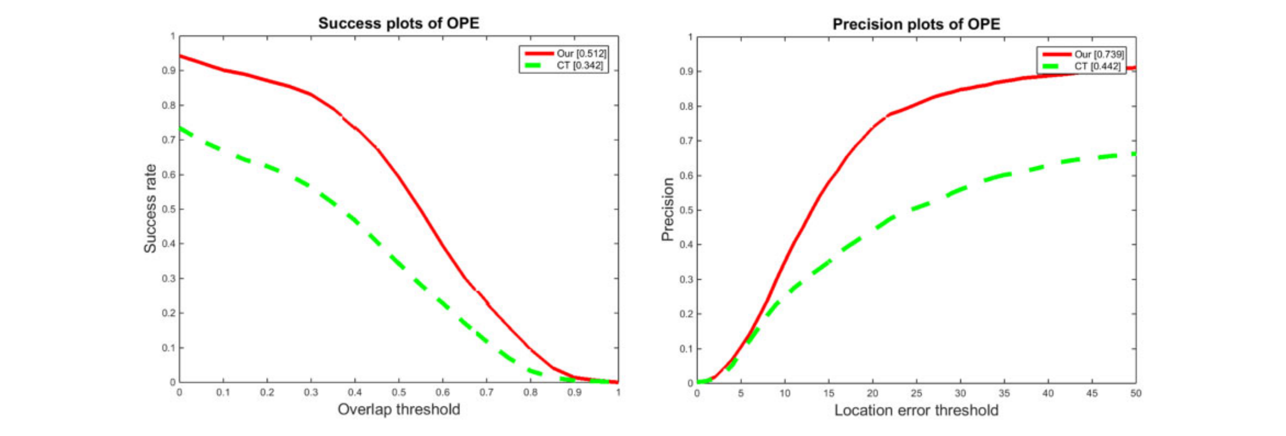


Figure 6. The success plots and precision plots for our improved CT and CT (without image segmentation, salient region detection, and image similarity).

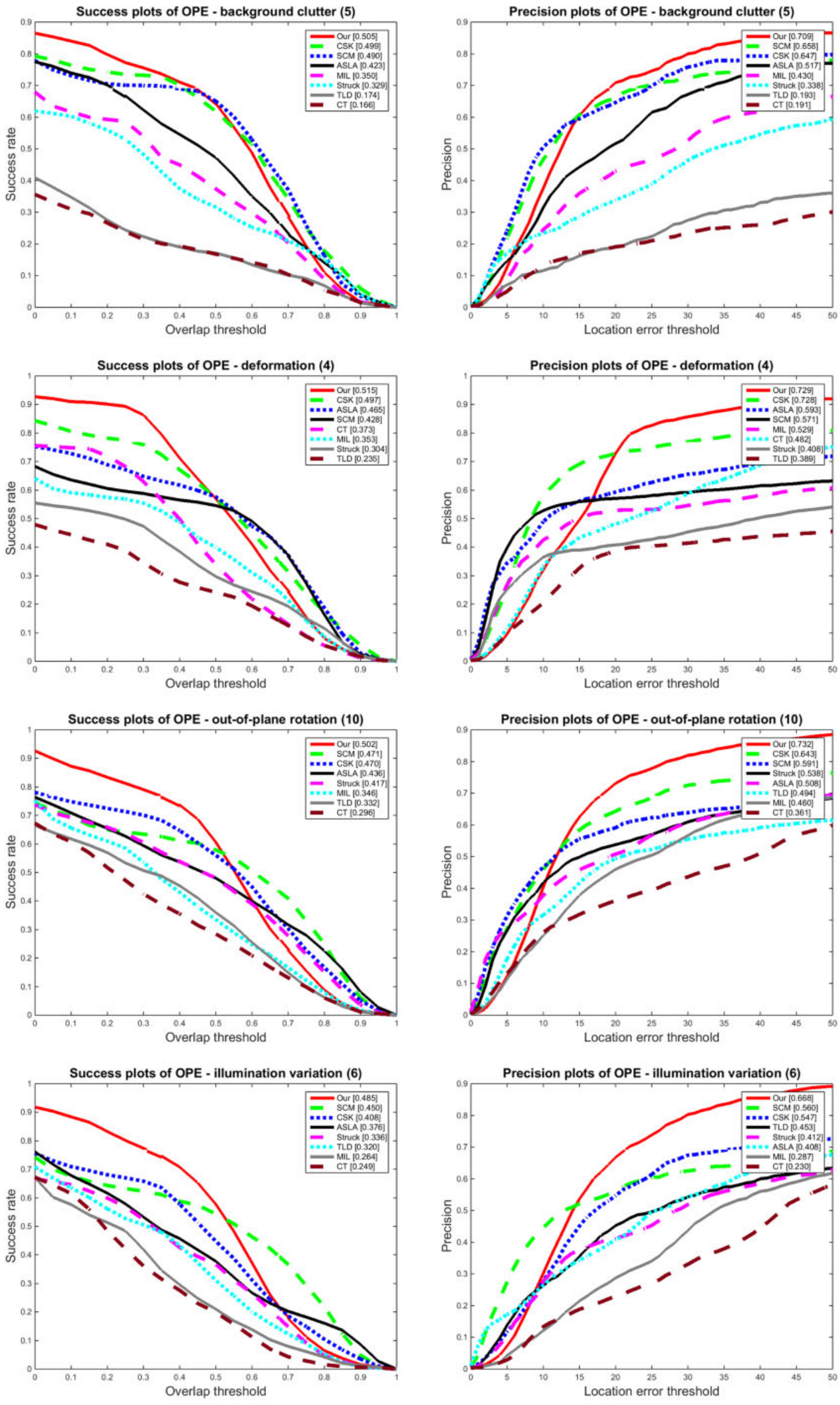


Figure 7. The success plots (left column) and precision plots (right column) between our improved CT tracker and some other classic trackers.

It is worth noting that when $t < 3$, p_t is meaningless. Based on this, we can define p_{tt} and p_{ts}, respectively, which is used to represent the hamming distance of the target and the scene. Then, we define the threshold f_c, and when it has a value of 1, the classifier updates, and when it has a value of 0, the classifier pauses the update. Our

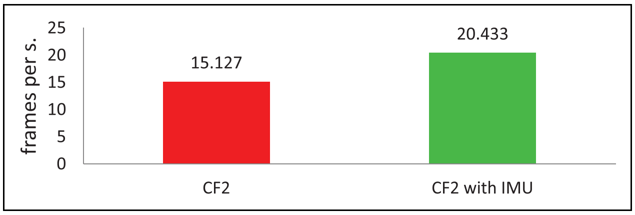


Figure 8. Under the same GPU conditions, the CF2 with improved model updater (CF2 with IMU) and CF2 operating speed comparison.

update strategy is based on the facts as follows: First, if the images are sufficiently similar, we do not make changes; second, if the difference between images is particularly large, we still stop updating the classifier. So, we set the update strategy as follows

- when $1 \leq t < 3, f_c = 1$;
- when $3 \leq t < 5$

$$f_c = \begin{cases} 1 & \begin{matrix} \delta \geq p_{\text{tt}} \geq \varepsilon \\ \& \\ \delta \geq p_{\text{ts}} \geq \varepsilon \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

when $5 \leq t$

$$f_c = \begin{cases} 1 & \begin{matrix} \delta \geq p_{\text{tt}} \geq \min\left\{ \varepsilon, \dfrac{\sum\limits_{i=3}^{t} p_{tt}}{t-3} \right\} \\ \& \\ \delta \geq p_{\text{ts}} \geq \min\left\{ \varepsilon, \dfrac{\sum\limits_{i=3}^{t} p_{ts}}{t-3} \right\} \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where ε is a threshold for the similarity between images and δ is a threshold for the difference between images. Based on the above strategy, we reduce the update of very similar frames to achieve faster-tracking speed and avoid updating the frame with very large difference to ensure that the classifier will not be mistakenly trained. As a result, our strategy can reduce the loss of target drift.

Tracking framework

Basic tracker MMMU. We build our tracking framework based on the works of Wang et al.[8] We also divide the tracking framework into five parts: motion model, model updater, feature extractor, ensemble post-processor, and observation model.[8] We have already elaborated on MMMU. As for the feature extractor, we choose (HOG + Raw Color) and this feature representation simply concatenates the Histogram of Oriented Gradient (HOG) and raw color features. In ensemble post-processor section, we use *Online Trajectory Optimization methods*, which is from the study by Bailer et al.[36] Our observation model is inspired by Struck[25] which exploits SO-SVM.

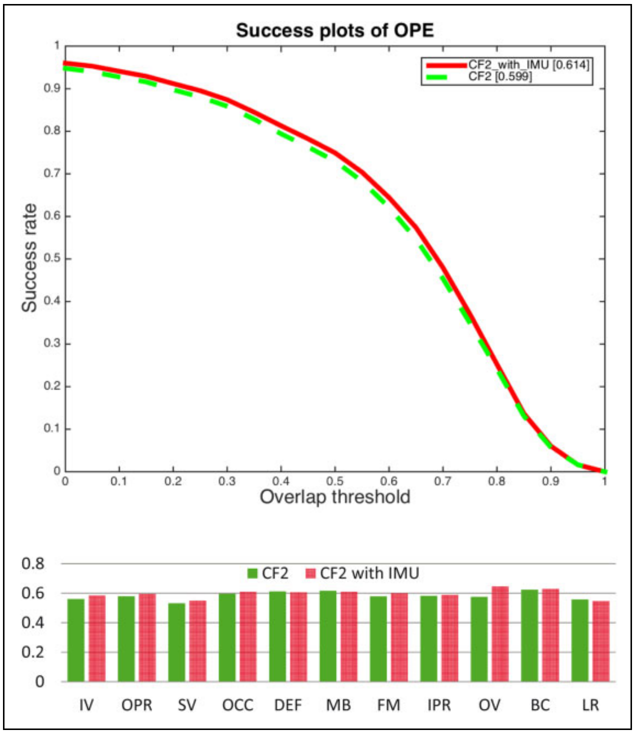


Figure 9. The success plots of OPE (top row) and the area under receiver operating characteristic curve (AUC) scores over 11 attributes (bottom row) made by improved CF2 tracker (model updater) and CF2 on OTB50. OPE: one-pass evaluation.

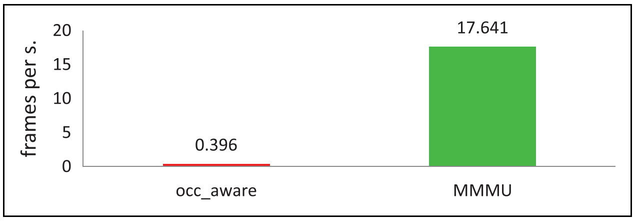


Figure 10. Under the same CPU conditions, the MMMU and occ aware tracker operating speed comparison.

We define an arbitrary sample s as follows in frame t: $s = (x, y, r)^{\mathrm{T}}$, where the coordinate points of the tracking rectangle are denoted by $(x, y)^{\mathrm{T}}$ and its scale is r. Unlike the maximum likelihood FIR filter proposed by Pak et al.[37] Although their approach to obtain the filter by maximizing the likelihood function is innovative, we think a linear model F can produce higher scores to samples in our framework. Here, $\widehat{s_j}$ is used to estimate the state of the target

$$\widehat{s_j} = \arg\max_{s} F(t, s) = \langle w, \phi(t, s) \rangle \tag{8}$$

where $\phi(t, s)$ represents the feature map of sample s, and w is the appearance parameter. We want the target score

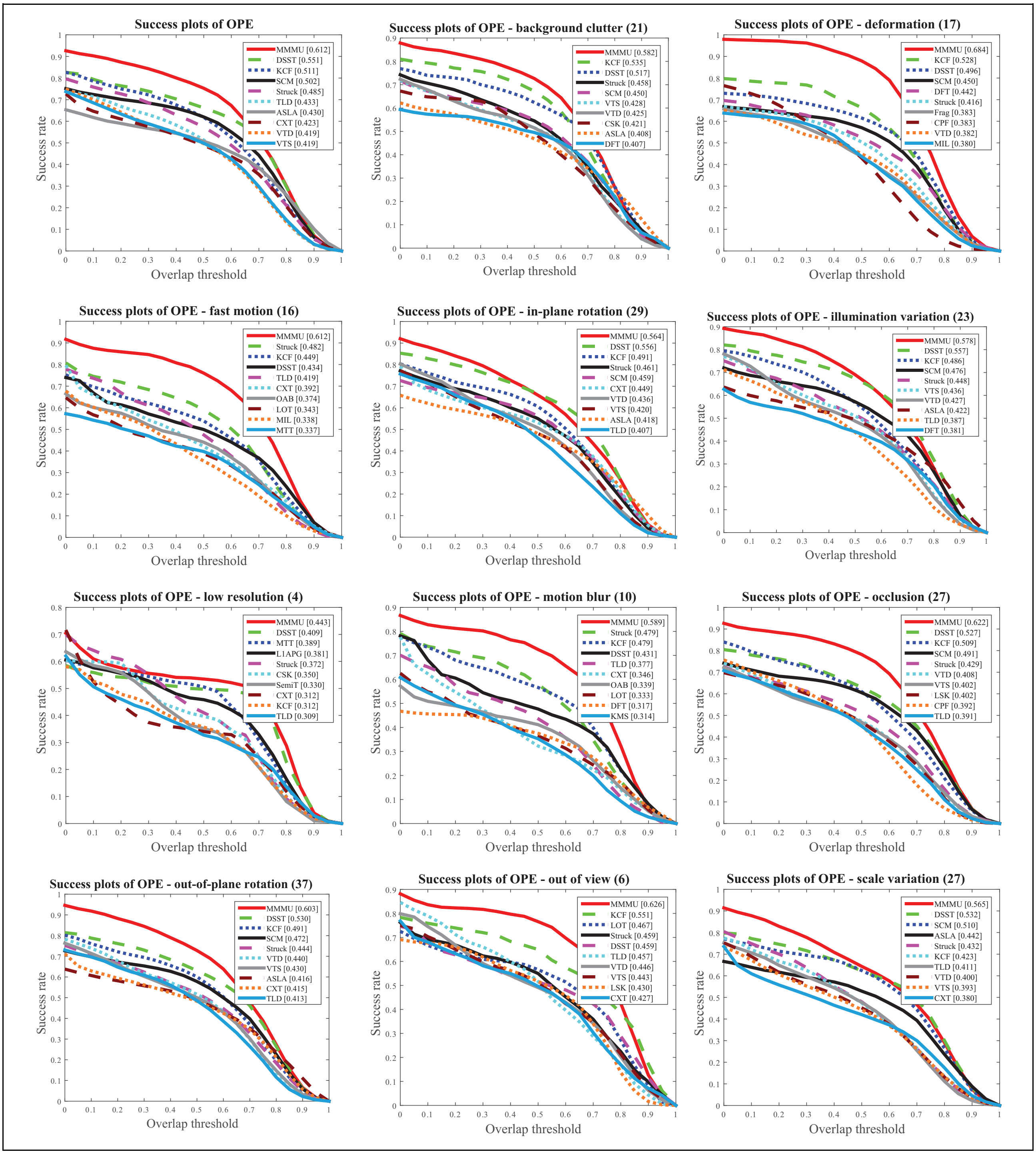


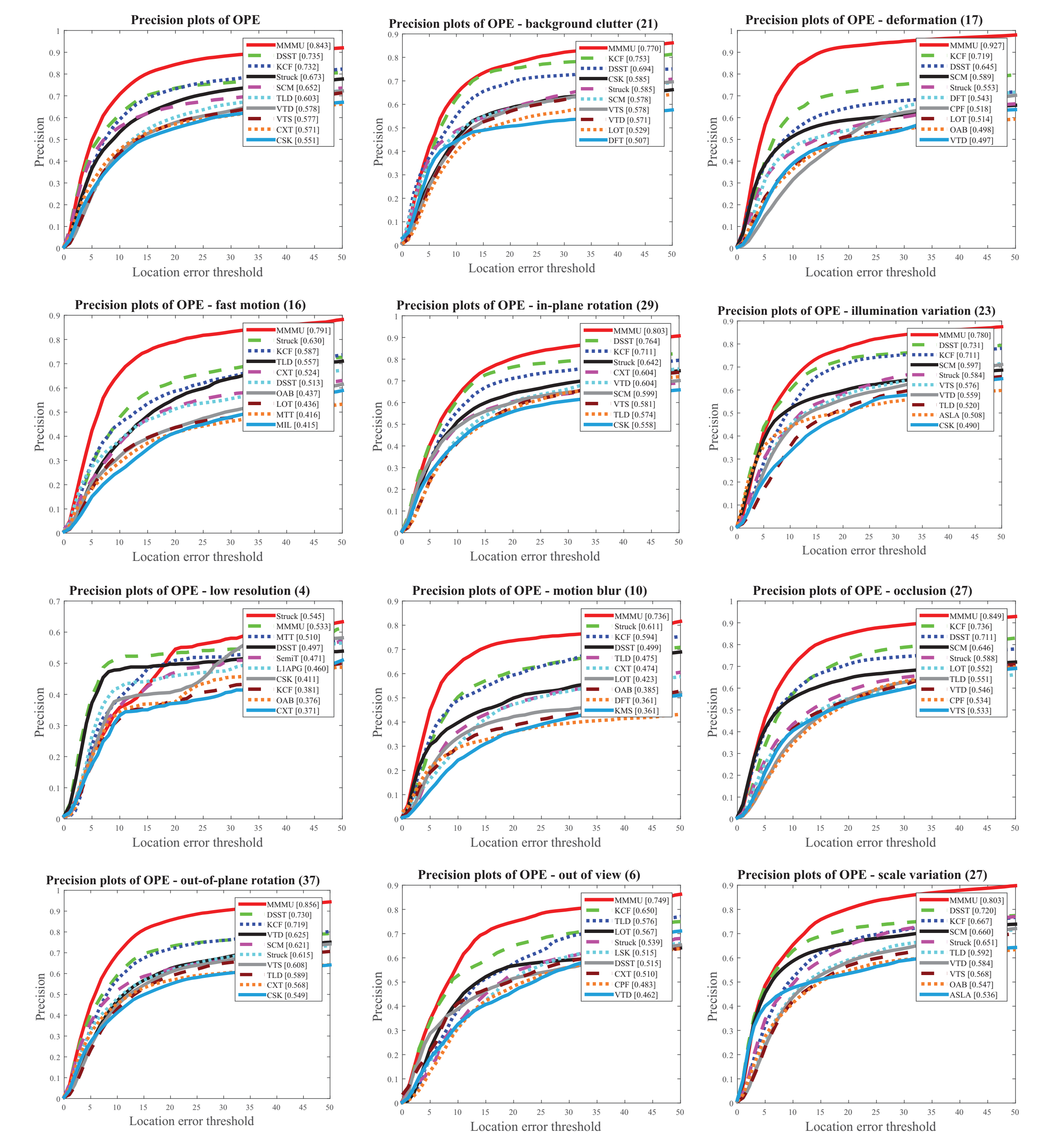
Figure 11. Success plots of OPE for MMMU and top 10 classic trackers in OTB50. In particular, we present success plots for 11 attributes. OPE: one-pass evaluation; MMMU: motion model and model updater.

$\langle w, \phi(t,\ \widehat{s}_j)\rangle$ to be higher than others. Assuming that the minimum margin of this difference is $\Delta(\ \widehat{s}_j, s)$. At this point, we define the loss term as

$$L(w) = \sum_j \left[\max_{s \neq \widehat{s}_j} \Delta(\widehat{s}_j, s) - w \cdot \delta\phi_j(t,s) \right]_+ \tag{9}$$

where $\delta\phi_j(t,s) = \phi(t,\ \widehat{s}_j) - \phi(t,s)$, and we define $\Delta(\ \widehat{s}_j, s)$ to represent the structural loss which can rescale the margin of each sample by the bounding box overlap ratio

$$\Delta(\widehat{s}_j, s) = 1 - \frac{\text{Area}(\widehat{s}_j \cap s)}{\text{Area}(\widehat{s}_j \cup s)} \tag{10}$$


Precision plots of OPE

Precision plots of OPE - background clutter (21)

Precision plots of OPE - deformation (17)

Precision plots of OPE - fast motion (16)

Precision plots of OPE - in-plane rotation (29)

Precision plots of OPE - illumination variation (23)

Precision plots of OPE - low resolution (4)

Precision plots of OPE - motion blur (10)

Precision plots of OPE - occlusion (27)

Precision plots of OPE - out-of-plane rotation (37)

Precision plots of OPE - out of view (6)

Precision plots of OPE - scale variation (27)

Figure 12. Precision plots of OPE for MMMU and top 10 classic trackers in OTB50. In particular, we present precision plots for 11 attributes. OPE: one-pass evaluation; MMMU: motion model and model updater.

Therefore, the optimal state $\widehat{s_j}$ is learned by the single objective function

$$\min_{w} \frac{1}{2}||w||_2^2 + \mathrm{CL}(w) \tag{11}$$

where C is scalar parameter and $||w||_2^2$ is the regularized term.

Experiment

Data Sets and evaluation methodology

We validate our methods on OTB50, which covers common challenges, such as illumination variation, scale variation (SV), OCC, DEF, motion blur, FM, in-plane rotation (IPR), out-plane rotation, out-of-view, background clutters,

Table 1. Compare with occ_aware tracker in success and precision plots of OPE.[a]

	Success plots of OPE		Precision plots of OPE	
	MMMU	occ_aware	MMMU	occ_aware
IV	*0.578*	0.566	0.78	*0.812*
OPR	*0.603*	0.591	*0.877*	0.856
SV	*0.565*	0.563	0.803	*0.81*
OCC	*0.622*	0.59	0.849	*0.855*
DEF	*0.684*	0.628	*0.927*	0.922
MB	*0.589*	0.538	0.736	*0.739*
FM	*0.612*	0.507	*0.791*	0.691
IPR	0.564	*0.584*	0.803	*0.851*
OV	*0.626*	0.524	*0.749*	0.67
BC	*0.582*	0.579	0.77	*0.832*
LR	0.433	*0.525*	0.533	*0.661*

OPE: one-pass evaluation; MMMU: motion model and model updater; IV: illumination variation; OPR: out-plane rotation; SV: scale variation; OCC: occlusion; DEF: deformation; MB: motion blur; FM: fast motion; IPR: in-plane rotation; OV: out-of-view; BC: background clutter; LR: low resolution.

[a]The best results are denoted in italics.

and low resolution (LR).[15] To better evaluate and analyze our methods, we mainly use this evaluation: one-pass evaluation (OPE).

Components test

Comparison of the CT with *improved motion model.* As a classic tracking method, CT[13] exhibits poor performance in tracking performance because of simple feature representation and simple motion model. Therefore, CT is suitable for verifying the effectiveness of our improved motion model. To evaluate the impact of improved motion model, we compare our model with the standard CT on OTB50[15] and get OPE to verify the performance. The results shown in Figure 6 confirm that our method significantly outperforms the baseline tracker CT in OPE protocol, and the score has increased by about 50%. And we only reduced by 1.836 frames/s in MATLAB R2014b on a PC with Intel i5 CPU 2.6 GHz.

In addition, we compare our tracker with seven tracking algorithms on OTB50, and these trackers are proposed almost the same period with CT. They are CSK,[38] SCM,[39] Struck,[25] ASLA,[40] TLD,[41] MIL,[42] and CT. The results in Figure 7 also show that our method almost get the best performance.

Comparison of the CF2 with improved model updater. In order to verify the validity of the improved model update, we employ CF2 tracker,[14] which exploits rich feature hierarchies of Convolutional Neural Networks (CNNs) as the experimental object. As a result of the richer feature, CF2 has perfect tracking performance, but also because it uses CNNs to make CF2 tracker relatively poor. Therefore, we optimize CF2 with improved model updater and name the new tracker CF2 with Improved Model Updater (IMU). In addition, after repeated tests from a large-scale data set, we define $\varepsilon = 3$ and $\delta = 15$.

We compare the speed and accuracy of the two trackers. As we known, CF2 is an excellent performance tracker, but runs slowly. We experimented on the Graphics Processing Unit (GPU) server, and their respective running speed is shown in Figure 8. Experiments show that CF2 with improved model updater in the running speed has indeed been significantly improved. And even more gratifying is that in addition to speed up 1/3, its tracking effect has also been improved to some extent. As shown in Figure 9, we can see that the improved tracker has improved on 8 attributes (there are 11 attributes in total). Experiments show that our approach not only reduces the number of classifier updates but also avoids the classifier incorrectly updating.

Basic tracker test

Our basic tracker MMMU will be tested in MATLAB on a 2.6-GHz Intel Core i5 CPU with 8-GB memory by

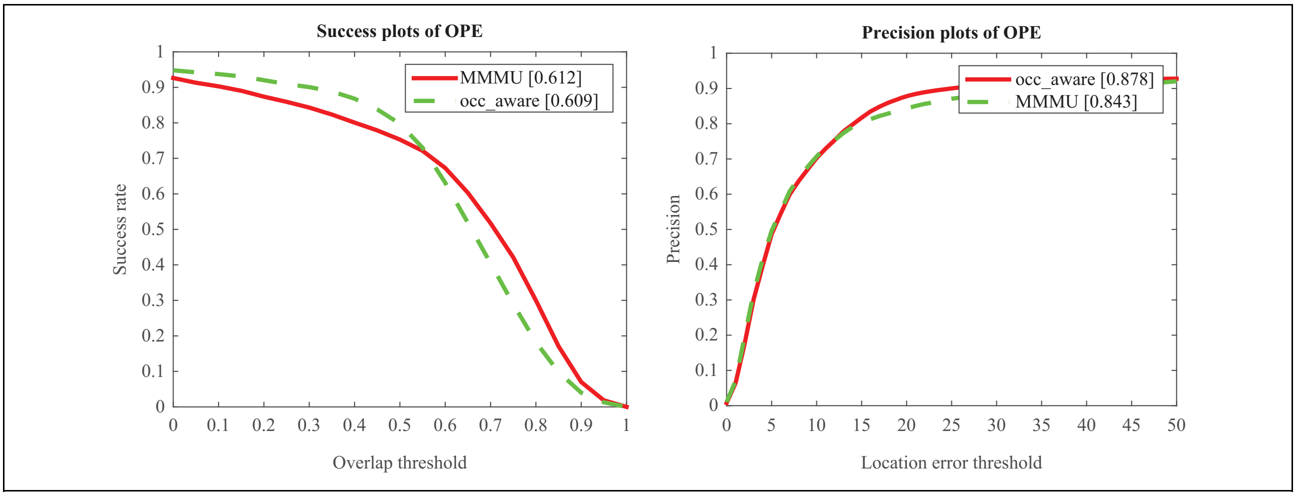


Figure 13. Success & precision plots of OPE for MMMU tracker and occ aware tracker in OTB 50.

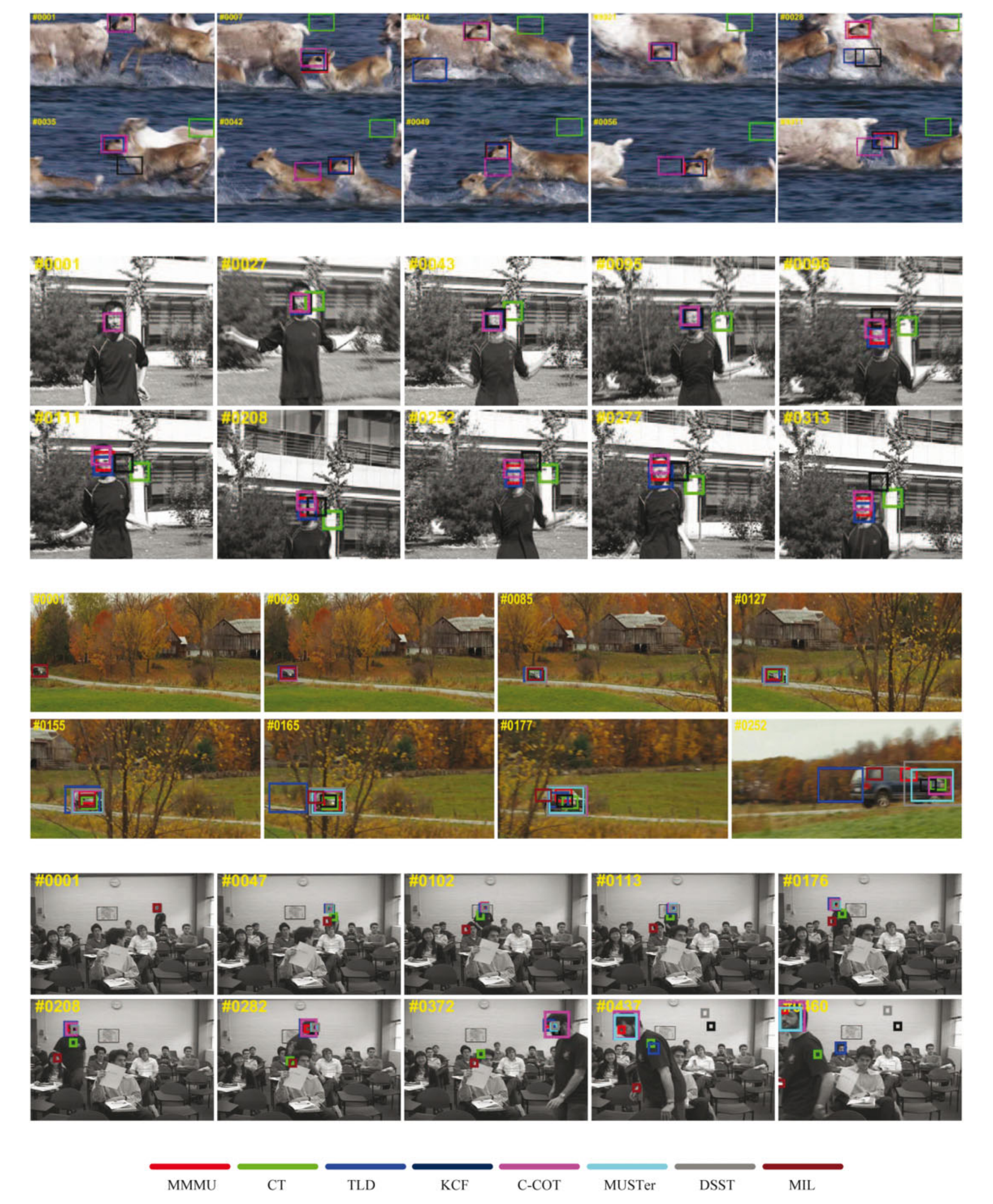


Figure 14. Tracking frames results on several typical examples. (a) *Deer*, (b) *Jumping*, (c) *CarScale*, and (d) *Freeman3*.

quantitative comparison and qualitative comparison. For quantitative comparison, we have chosen some representative trackers, such as Struck,[25] ASLA,[40] SCM,[39] TLD,[43] CXT,[44] VTD,[45] VTS,[46] LSK,[47] CSK,[48] DFT,[49] MTT,[50] OAB,[51] MIL,[42] CPF,[52] KCF,[53] and occ aware.[54] For qualitative comparison, we choose seven trackers, including CT,[13] TLD,[43] KCF,[53] MIL,[42] C-COT,[55] MUSTer,[56] and DSST.[57] All tests were performed on TB50.[15]

Quantitative comparison. As shown in Figure 11, Our tracker is ahead of all other tracking methods on all attributes in OPE success plots, especially in this DEF attribute, MMMU leads the best performance of KCF approximately 29.545%. The results from Figure 12 show that MMMU is significantly ahead of the opponents in 10 attributes except LR in OPE precision plots and just reduced by 2.201%. Therefore, as shown in Figures 11(a) and 12(a), our MMMU has shown a greater advantage both in the success and precision plots of OPE protocol.

In particular, we compare our MMMU to occ_aware tracker[54] which is a superior performance tracking algorithm. As shown in Figure 13, the two methods have their own advantages in the OPE protocol: in the precision plots of Figure 13(a), occ_aware is better than MMMU; in the success plots of Figure 13(b), MMMU has better performance. As shown in Table 1, our MMMU is superior to occ_aware in nine attributes of the success plots and four attributes of the precision plots. Therefore, MMMU is slightly dominant than occ_aware in OPE protocol.

However, in terms of running speed, MMMU occupy an absolute advantage, as shown in Figure 10; MMMU is 44 times faster than occ_aware in the same platform.

Qualitative comparison. FM is a typical problem in tracking research, such as *Deer* sequence in Figure 14(a), CT loses its target since frame #0007, TLD drifts completely since frame #0014, KCF incorrectly locates the target since frame #0035, and MIL also loses the target after the next several frames. Because of the rapid movement, the target will be far away from current position, but the range of current frame samples selection is fixed (generally rectangular or circular areas centered on the target). These samples, especially nontarget samples, may not represent the next frame target and its surrounding appearance; therefore, the samples of the current frame selected by this method are not suitable for training the classifier. This is the main reason why all other four trackers fail. Our method does not fix the sample selection region; hence, MMMU in this video series has good robustness. The same situation appears in *Jumping* sequence shown in Figure 14(b), CT and KCF lose their targets since frame #0027 and frame #0096, respectively.

In OTB50, there are many moving car videos that involve SVs. We choose one of the typical examples for analysis. CarScale video is challenging for scale variation and cluttered background, as shown in Figure 14(c).[15] From frames #0001 to #0085, each tracker has a good performance because the scale of the car changes very little. After frame #86, due to the SV, CT, TLD, and MIL have an unstable track. When a tree appears in front of the camera in frame #0155, OCC directly leads to CT and TLD failure. In frame #242, DSST partly drift. Our MMMU is still excellent.

Face tracking is a common tracking problem, SV, IPR, and out-of-plane rotation are usually the characteristics of this problem. From the results in Figure 14(d), *Freeman3* sequence, we find that MIL and CT have bad track from frame #47. In frame #102, when the target turns around, MIL tracks an erroneous target. In frame #437, KCF loses its target because of SV and IPR. Finally, TLD and MIL also retracks to the target, but MMMU maintains the correct trace all the time.

Conclusion

In this article, we propose efficient methods for conventional visual tracking in MMMU. Our method more comprehensively considers the visual-spatial attention factors in the appearance template, such as color, distance, intensity, and texture. Through cooperation between salient region detection and image segmentation, we get an effective motion model which has the right balance between target processing and scene analysis. Inspired by the biological memory system, we further develop an effective online model updater using fast image similarity to measure the rationality of the estimated target and current scene in the time dimension for reducing the frequency of the model update and the probability of classifier error updates. The experimental results prove that the MMMU performs favorably against most trackers in terms of efficiency and accuracy.

Authors' note

This paper was presented in part at the CCF Chinese Conference on Computer Vision, Tianjin, 2017. This paper was recommended by the program committee.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the National Key Technology R&D Program under grant 2015BAH52F00 and in part by the National Natural Science Foundation of China under grant 61304262.

ORCID iD

Wanli Xue http://orcid.org/0000-0002-6031-9334
Chao Xu http://orcid.org/0000-0002-6398-0398
Chengwei Zhang http://orcid.org/0000-0002-9157-6050

References

1. Zhao L, Gao X, Tao D, et al. Learning a tracking and estimation integrated graphical model for human pose tracking. *IEEE Trans Neural Netw Learn Syst* 2015; 26(12): 3176–3186.
2. Tian C, Gao X, Wei W, et al. Visual tracking based on the adaptive color attention tuned sparse generative object model. *IEEE Trans Image Process* 2015; 24(12): 5236–5248.
3. Zhao L, Gao X, Tao D, et al. Tracking human pose using max-margin Markov models. *IEEE Trans Image Process* 2015; 24(12): 5274–5287.

4. Lin L, Lu Y, Li C, et al. Detection-free multiobject tracking by reconfigurable inference with bundle representations. *IEEE Trans Cybern* 2016; 46(11): 2447–2458.
5. Xue W, Xu C and Feng Z. Robust visual tracking via multiscale spatio-temporal context learning. *IEEE Trans Circuits Syst Video Technol* 2017; PP(99): 1–1. DOI: 10.1109/TCSVT.2017.2720749.
6. Liu T, Xu C, Meng Z, et al. Improvement on tracking based on motion model and model updater. In: *CCF Chinese conference on computer vision* (eds Yang J, Hu Q, Cheng M, et al.), Singapore: Springer, 2017, pp. 639–650.
7. Yang H, Shao L, Zheng F, et al. Recent advances and trends in visual tracking: a review. *Neurocomputing* 2011; 74(18): 3823–3831.
8. Wang N, Shi J, Yeung DY, et al. Understanding and diagnosing visual tracking systems. In: *Proceedings of the IEEE international conference on computer vision*, Washington, DC, USA: IEEE Computer Society, 2015, pp. 3101–3109.
9. Achanta R, Shaji A, Smith K, et al. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans Pattern Anal Mach Intell* 2012; 34(11): 2274–2282.
10. Achanta R, Hemami S, Estrada F, et al. Frequency-tuned salient region detection. In: *Computer vision and pattern recognition, 2009 CVPR 2009 IEEE conference on*. IEEE, 2009, pp. 1597–1604.
11. Mıhçak MK and Venkatesan R. New iterative geometric methods for robust perceptual image hashing. In: *ACM workshop on digital rights management*. Springer, 2002, pp. 13–21.
12. Song H, Zheng Y and Zhang K. Robust visual tracking via self-similarity learning. *Electron Lett* 2016; 53(1): 20–22.
13. Zhang K, Zhang L and Yang MH. Real-time compressive tracking. In: *European conference on computer vision* (eds A Fitzgibbon, et al.), Springer, 2012, pp. 864–877.
14. Ma C, Huang JB, Yang X, et al. Hierarchical convolutional features for visual tracking. In: *Proceedings of the IEEE international conference on computer vision*, Washington, DC, USA: IEEE Computer Society, 2015, pp. 3074–3082.
15. Wu Y, Lim J and Yang MH. Online object tracking: a benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Washington, DC, USA: IEEE Computer Society, 2013, pp. 2411–2418.
16. Hong X, Chang H, Shan S, et al. Sigma set based implicit online learning for object tracking. *IEEE Signal Process Lett* 2010; 17(9): 807–810.
17. Xu C, Tao W, Meng Z, et al. Robust visual tracking via online multiple instance learning with Fisher information. *Pattern Recognit* 2015; 48(12): 3917–3926.
18. Dong X, Shen J, Yu D, et al. Occlusion-aware real-time object tracking. *IEEE Trans Multimedia* 2017; 19(4): 763–771.
19. Ma B, Huang L, Shen J, et al. Visual tracking under motion blur. *IEEE Trans Image Process* 2016; 25(12): 5867–5876.
20. Ma B, Hu H, Shen J, et al. Generalized pooling for robust object tracking. *IEEE Trans Image Process* 2016; 25(9): 4199–4208.
21. Ma B, Huang L, Shen J, et al. Discriminative tracking using tensor pooling. *IEEE Trans Cybern* 2016; 46(11): 2411–2422.
22. Ma B, Shen J, Liu Y, et al. Visual tracking using strong classifier and structural local sparse descriptors. *IEEE Trans Multimedia* 2015; 17(10): 1818–1828.
23. Song H, Zheng Y and Zhang K. Efficient algorithm for piecewise-smooth model with approximately explicit solutions. *Electron Lett* 2017; 53(4): 233–235.
24. Wang D, Lu H, Xiao Z, et al. Inverse sparse tracker with a locally weighted distance metric. *IEEE Trans Image Process* 2015; 24(9): 2646–2657.
25. Hare S, Saffari A and Torr PH. Struck: structured output tracking with kernels. In: *2011 international conference on computer vision*, Washington, DC, USA: IEEE Computer Society, 2011, pp. 263–270.
26. Choi IH, Pak JM, Ahn CK, et al. Arbitration algorithm of fir filter and optical flow based on ANFIS for visual object tracking. *Measurement* 2015; 75: 338–353.
27. Zuo W, Wu X, Lin L, et al. Learning support correlation filters for visual tracking. *arXiv preprint arXiv:160106032* 2016.
28. Wang D, Lu H and Bo C. Fast and robust object tracking via probability continuous outlier model. *IEEE Trans Image Process* 2015; 24(12): 5166–5176.
29. Liu Q, Ma X, Ou W, et al. Visual object tracking with online sample selection via lasso regularization. *Signal Image Video Process* 2017; 11(5): 881–888.
30. Song H, Wang G and Zhang K. Multiple change detection for multispectral remote sensing images via joint sparse representation. *Opt Eng* 2014; 53(12): 123103–123103.
31. Song H, Huang B, Liu Q, et al. Improving the spatial resolution of landsat TM/ETM+ through fusion with SPOT5 images via learning-based super-resolution. *IEEE Trans Geosci Remote Sens* 2015; 53(3): 1195–1204.
32. Song H, Wang G and Zhang K. Hyperspectral image denoising via low-rank matrix recovery. *Remote Sens Lett* 2014; 5(10): 872–881.
33. Kang T, Mo Y, Pae D, et al. Robust visual tracking framework in the presence of blurring by arbitrating appearance- and feature-based detection. *Measurement* 2017; 95: 50–69.
34. Zhang K, Zhang L and Yang MH. Fast compressive tracking. *IEEE Trans Pattern Anal Mach Intell* 2014; 36(10): 2002–2015.
35. Krawetz N. Looks like it. *The Hacker Factor Blog* 2011. http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html.
36. Bailer C, Pagani A and Stricker D. A superior tracking approach: building a strong tracker through fusion. In: *European conference on computer vision* (eds D Fleet, et al.), 2014, pp. 170–185.
37. Pak JM, Ahn CK, Mo YH, et al. Maximum likelihood FIR filter for visual object tracking. *Neurocomputing* 2016; 216: 543–553.
38. Danelljan M, Shahbaz Khan F, Felsberg M, et al. Adaptive color attributes for real-time visual tracking. In: *Proceedings of the IEEE conference on computer vision and pattern*

recognition, Washington, DC, USA: IEEE Computer Society, 2014, pp. 1090–1097.

39. Zhong W, Lu H and Yang MH. Robust object tracking via sparsity-based collaborative model. In: *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*, Washington, DC, USA: IEEE Computer Society, 2012, pp. 1838–1845.
40. Jia X, Lu H and Yang MH. Visual tracking via adaptive structural local sparse appearance model. In: *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*, Washington, DC, USA: IEEE Computer Society, 2012, pp. 1822–1829.
41. Kalal Z, Mikolajczyk K and Matas J. Tracking-learning-detection. *IEEE Trans Pattern Anal Mach Intell* 2012; 34(7): 1409–1422.
42. Babenko B, Yang MH and Belongie S. Visual tracking with online multiple instance learning. In: *Computer vision and pattern recognition, 2009 CVPR 2009 IEEE conference on*. IEEE, 2009, pp. 983–990.
43. Kalal Z, Matas J and Mikolajczyk K. Pn learning: bootstrapping binary classifiers by structural constraints. In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 49–56.
44. Dinh TB, Vo N and Medioni G. Context tracker: exploring supporters and distracters in unconstrained environments. In: *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, Washington, DC, USA: IEEE Computer Society, 2011, pp. 1177–1184.
45. Kwon J and Lee KM. Visual tracking decomposition. In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 1269–1276.
46. Kwon J and Lee K. Tracking by sampling trackers. In: *2011 international conference on computer vision*, Washington, DC, USA: IEEE Computer Society, 2011, pp. 1195–1202.
47. Liu B, Huang J, Yang L, et al. Robust tracking using local sparse appearance model and k-selection. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*. IEEE, 2011, pp. 1313–1320.
48. Henriques JF, Caseiro R, Martins P, et al. Exploiting the circulant structure of tracking-by-detection with kernels. In: *European conference on computer vision*, Berlin, Heidelberg: Springer, 2012, pp. 702–715.
49. Sevilla-Lara L and Learned-Miller E. Distribution fields for tracking. In: *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*. IEEE, pp. 1910–1917.
50. Zhang T, Ghanem B, Liu S, et al. Robust visual tracking via multi-task sparse learning. In: *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*, Washington, DC, USA: IEEE Computer Society, 2012, pp. 2042–2049.
51. Grabner H, Grabner M and Bischof H. Real-time tracking via on-line boosting. In: *BMVC*, 2006, vol. 1. pp. 47–56.
52. Pérez P, Hue C, Vermaak J, et al. Color-based probabilistic tracking. In: *European conference on computer vision*, Berlin Heidelberg: Springer, 2002, pp. 661–675.
53. Henriques JF, Caseiro R, Martins P, et al. High-speed tracking with kernelized correlation filters. *IEEE Trans Pattern Anal Mach Intell* 2015; 37(3): 583–596.
54. Sun C, Wang D and Lu H. Occlusion-aware fragment-based tracking with spatial-temporal consistency. *IEEE Trans Image Process* 2016; 25(8): 3814–3825.
55. Danelljan M, Robinson A, Khan FS, et al. Beyond correlation filters: learning continuous convolution operators for visual tracking. In: *European conference on computer vision*. Cham: Springer, 2011, pp. 472–488.
56. Hong Z, Chen Z, Wang C, et al. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 749–758.
57. Danelljan M, Häger G, Khan F, et al. Accurate scale estimation for robust visual tracking. In: *British machine vision conference* (eds M Valstar, A French and T Pridmore), Nottingham, UK, 1–5 September 2014. BMVA Press.