Robust Tracking and Redetection: Collaboratively Modeling the Target and Its Context

Chang Liu, Peng Liu, Wei Zhao, and Xianglong Tang

***Abstract*—Robust object tracking and redetection require stably predicting the trajectory of the target object and recovering from tracking failure by quickly redetecting it when it is lost during long-term tracking. The locations of the target and the background are calculated relative to the region occupied by the object. The effect of tracking can be enhanced by isolating the target and the background, modeling and tracking them, respectively, and integrating their tracking results. In this study, we propose an approach that builds motion models for the target and its context. Tracking results from a target tracker and a context tracker are integrated through linear fusion to predict the position of the target. A kernelized correlation filter tracker is used to track the target in the predicted position. When the target is lost, it can be quickly recovered by searching in the given field of view using a target model built and updated through observation models that are constructed prior to the loss of the target. Our approach is not sensitive to the segmentation of the target and the context. The motion models and observation models of the target and the context work together in the tracking process, whereas the target model alone is involved in redetection. Experiments to test our proposed approach, which simultaneously models the target and its context, showed that it can effectively enhance the robustness of long-term tracking.**

***Index Terms*—Collaborative modeling, fusion of tracking results, Re-detection, target tracking.**

I. Introduction

VISUAL tracking has progressed considerably in the last 30 years because of its wide variety of applications in surveillance, navigation, the military, and so on. It involves predicting variation in the position of a target in image sequences with a given initial state. Although significant advances have been made in target tracking [1], [2], it remains challenging to smoothly track a target in a long video because of a number of factors, such as illumination variation, scale variation, occlusion, deformation and fast motion [3], [4].

Robustness, adaptivity, and real-time processing are factors to consider in designing a robust tracking system [5]. In [6], Naiyan Wang *et al*. divided a tracking system into multiple constituent parts, including a motion model, a feature extractor, an observation model and a model updater. Robustness is influenced by the feature extractor, which describes an appearance model [7]. Most prevalent trackers, ranging from the simple normalized cross-correlation (NCC) tracker [8] to the complex deep learning trackers [9], [10], suffer from a contradiction between precision and speed, whereby high-speed trackers are prone to errors and high-precision trackers are generally inefficient.

(a) (b)

Fig. 1. Example image of bound effect. The red box represents the target, whereas the green box shows the size of the samples in KCF. (a) Original image. (b) Periodicity in correlation filter.

Currently available trackers can be classified into generative or discriminative trackers according to the method used for online learning [14]. Correlation filter-based tracking [11] has recently attracted a considerable amount of research attention, especially following the introduction of the kernelized correlation filter (KCF) tracker [12], which performs well at high speed. Long-term tracking is another interesting issue. It is an expanded version of ordinary tracking with significantly enhanced requirements of robustness and adaptivity. Reference [25] points out that the long-term tracking problem can be approached either from the perspective of tracking or detection.

The most commonly used KCF tracker is regarded as representative to highlight drawbacks in the methods reported abrove. The KCF classifier is trained using circular samples to detect the target, and its tracking precision is enhanced at a small time cost. In spite of the benefit of circular samples in terms of computational efficiency, they cause boundary effects [13] (see Fig. 1). As show in Fig. 1, the KCF filter collects a large number of samples in the target region and its surroundings. Samples are commonly taken from the original image (a); however, to use the fast Fourier transform, the surroundings of the target are artificially synthesized, as in (b), which involves converting the boundaries of the samples into error. The samples located far from the sampling center feature a significant number of inaccurate contents; while this causes some errors to be learned

Manuscript received July 7, 2017; accepted September 15, 2017. Date of publication October 6, 2017; date of current version March 15, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61671175. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Yap-Peng Tan. *(Corresponding author: Wei Zhao.)*

The authors are with the Department of Computer Science, Harbin Institute of Technology, Harbin 150001, China (e-mail: 15b903045@hit.edu.cn; pengliu@hit.edu.cn; zhaowei@hit.edu.cn; tangxl@hit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2017.2760633

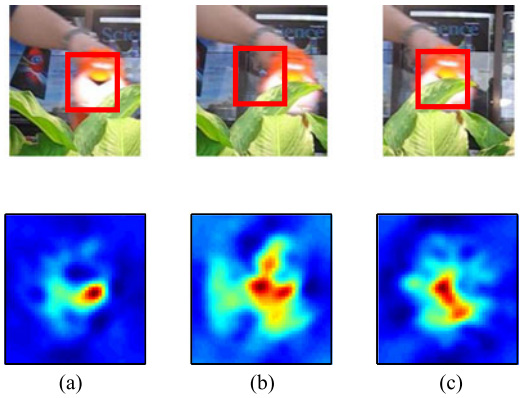

Fig. 2. An example of false detection in the video Tiger2. (a) The KCF detector in frame 105. (b) The KCF detector in frame 107. (c) The KCF detector in frame 107 with a corrected sampling center. The figure at the top shows the sampling region and that at the bottom the KCF response. The KCF detects the target where the response is maximum, and the result of detection is shown in the red box.

in training, the KCF tracker fails when the target is far from the sampling center. This phenomenon is common in high-speed object tracking.

In KCF, every sample is 2.5 times the target in terms of size, which means that the background has an effect on tracking. The KCF tracker detects the target in a search region of the same size as the samples, and locates the target where the response is maximum. It is a double-edged sword, whereby the potential relationship between the target and the context helps detect the former but enhances the likelihood that something in the background is mistakenly detected as the target. In Fig. 2, KCF fails in frame 107 because inaccurate samples blur the response; but in (c), if the sampling center is corrected, the detection result improves. The images in Fig. 2 indicate that the tracking precision of KCF depends on where the target is in the searched region. Many trackers or detectors work well in search regions of limited size. Nevertheless, the unstable motion of the changeable target causes them to fail, especially in cases involving fast motion. Although most trackers are reliable and effective in limited regions, they are prone to failure as the regions expand. For example, we tested the popular KCF with different region sizes in OTB100. The precision score was 69.2% when the size of the region was 2.5 times the target, and these scores were 67.4%, 62.5%, and 56.5% with regions of sizes three, 3.5, and four times the target, respectively. The KCF tracker performed unfavorably when regions were expanded. Hence, it is not the classifier, but an inaccurate motion model, that leads to failure.

In long-term tracking, there are many challenges that need to be addressed simultaneously. It is difficult for an ordinary tracker to perform well. Many researches have focused on short-term tracking, with no recourse when the target disappears. For example, once the KCF detector recognizes an incorrect object as target, it cannot recover. Tracking failure is common in long-term tracking. A minute disturbance in tracking can result in failure, such as a change of scene or an adjustment to the tracking parameters; failure is inevitable when the target is fully occluded. To ensure reliable long-term tracking performance, it is necessary in re-detection to search for the target in the entire image when the target reappears. However, most trackers cannot track or re-detect the target in the entire image.

It is necessary to use an expanded region to track the target in case of fast motion or long-term occlusions. In this study, we propose a method–Target and Background-underlined Tracking with a Kernelized Correlation Filter (TBKCF)–that emphasizes the independent functions of the target and its context in tracking. The proposed method aims to generalize a number of basic trackers in expanded regions. Because of their different properties, the target and the context play different roles in motion estimation. The target model records the general features of the target and provides valuable reference for tracking, so that it can operate in short- and long-term motion prediction because features are vital clues even when the target is lost. However, the results of the prediction of the target model can be confused in a large search region because there may exist objects causing disturbance with features identical to those of target. The context can be used to estimate the temporary motion of the target because it supplies valuable clues when the target is unstable or suffers from irregular deformation. Therefore, the target and context are tracked respectively, and the short-term motion model is a compromise between the two, whereas the long-term motion model is dominated by the target model. When the target is tracked in a large region, the time needed is longer than most trackers can afford. Consequently, the NCC tracker with FFT is implemented to efficiently track the target and the context. However, the NCC tracker is prone to drifting to similar objects in the given region. Accordingly, information concerning the context is integrated into the model to discriminate among similar objects. Furthermore, the KCF tracker is used for accurate localization and verification of the tracking results. In the proposed method, the target and background are tracked, respectively, and integrated to predict the motion of the former. Moreover, the target can be re-detected when lost.

The main contributions of this paper can be summarized as follows:

1) We propose a short-term model that tracks the target in expanded regions to deal with fast motion. The short-term model runs efficiently with fast NCC, and collaboratively models the target and its context to overcome the drift problem of the NCC tracker.
2) We explore a novel context model where the context is hollow, and improve the original NCC to track context.
3) We introduce a method to fuse the context tracker and the target tracker, such that reliable local detection regions are selected.
4) We establish a general long-term motion model that focuses on the target's identity in re-detection.

The rest of this paper is organized as follows: Section II introduces related work in the area, Section III presents the problem formulation, Section IV describes detailed algorithms for different parts of the tracking, Section V describes and analyses our experiments to test the proposed method, and Section VI contains our conclusions.

II. Related Work

A number of visual tracking methods have been proposed in recent research. In the following, we briefly introduce the generative and discriminative models, review the development of correlation tracking, reveal the main ideas underlying KCF, and the advantages of our approach.

A. Generative Models

These methods focus on learning a perfect appearance model to represent the target and matching to find the most similar of the candidates [15] [40]. NCC [8] and KLT [17] were the original trackers. They are easy and efficient, but prone to drift as the area of the search region increases. LOT [18] segments the target into superpixels for better adaptation in terms of the objects appearance. IVT [19] maintains an extended appearance model that learns a low-dimensional subspace representation. Since L1T [20] sparsely represents candidates using L1-norm minimization, sparse representation has contributed considerably to object tracking [21]. B. Ma proposed a multi-task reverse sparse representation model [42] to highlight motion blur in visual tracking. The authors of [43] presented a generalized feature pooling method by modeling the statistical distribution of sparse codes, and those of [44] argued that pooling tensors can preserve more structural information concerning the target appearance than pooling vectors. MTT [22] improves tracking performance through multi-task learning. Zhou *et al*. [23] proposed similarity fusion for template matching.

Generative models can measure how similar a given object is to a template. However, the learned appearance and motion models are insufficient for locating the target in unknown environments because background information is ignored. Disturbing the objects surrounding the target leads to drift.

B. Discriminative Models

In these methods, object tracking is formulated as a classification or regression problem and the object is located by detection [41]. Usually, positive and negative samples are selected to train and update the classifier. MIL [24] involves learning a discriminative classifier from bags of samples. TLD [25] divides long-term tracking into tracking, learning, and detection. Struck [26] introduced a structured output learning method that can train a structured SVM. CT [27] uses compressive sensing to compress image features. The authors of [16] utilized structural local sparse descriptors and trained a number of weak classifiers to obtain a boosting-based strong classifier. The authors of [45] trained a nonlinear classifier with a sparse dictionary to adapt to changing appearances.

Discriminative trackers learn the difference between the target and the background, and hence are resistant to background clutters. However a perfect classifier cannot be trained, errors accumulate with updates, and eventually necessitate error detection.

C. Correlation Tracking

In correlation filter-based tracking, correlation peaks are produced in the target while low responses are obtained in the background. Target models are learned with context, which helps suppress errors due to inaccurate bounding boxes. The training requirement had rendered the use of correlation filters for online tracking inappropriate until the proposal of the minimum-output sum of squared error [28], which uses FFT to expedite training. In CSK [29], circular samples are used to interpret the MOOSE, and are trained through ridge regression, which was improved upon in KCF. CN [30] introduces color attributes, whereas DSST [31] uses a feature pyramid and a 3D correlation filter to estimate scale. RPAC [32] introduces a part-based tracking strategy to deal with occlusion and trains a correlation filter for every part. SCT [33] trains the part-based model as a whole, LCT [34] enables re-detection with an online random fern classifier. SRDCF [13] suppresses bound effects by using a spatially regularization. It is improved with CNN features [35] and attained impressive performance in VOT2015 challenge [36].

The benefit of correlation filters is their excellent performance at high speed. The above correlation filter-based trackers mostly improve performance by using better features. However, the effect of the motion model is seldom considered.

D. Kernelized Correlation Filters

Henriques *et al*. [12], [29] proposed considering object tracking a kernelized ridge regression problem. It involves training classifier $y = f(\mathbf{x})$ with set of samples $\mathbf{X} = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]^{\mathrm{T}}$ and corresponding labels $\mathbf{y} = [y_1 \ \cdots \ y_n]^{\mathrm{T}}$.

Samples are constructed circularly, which means that $\mathbf{x}_1 = \mathbf{x}$ is the feature vector of the target and its context, whereas $\mathbf{x}_i$, $i = 2, \ldots, n$ is generated circularly from $\mathbf{x}$. The ridge regression can be trained by:

$$\hat{\boldsymbol{\alpha}} = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda} \tag{1}$$

where $\boldsymbol{\alpha}$ is the parameter of f, $\hat{\mathbf{y}}$ denotes the Fourier transform of $\mathbf{y}$, $\hat{\mathbf{k}}^{\mathbf{xx}}$ is the kernel correlation of $\mathbf{x}$, and λ is a regularization parameter to control overfitting. In detection, a new sample $\mathbf{z}$ is detected by :

$$\hat{\mathbf{y}} = \hat{\mathbf{k}}^{\mathbf{xz}} \cdot \hat{\boldsymbol{\alpha}} \tag{2}$$

where $\hat{\mathbf{k}}^{\mathbf{xz}}$ is the kernel correlation of $\mathbf{x}$ and $\mathbf{z}$. The target is located where the value of $\mathbf{y}$ is maximum.

With the kernel trick, potential relationships within the target, and between the target and the background, are used for training and detection. KCF is involved in our methods as a baseline.

E. Advantages of Our Approach

The proposed method aims to mine clearer relationships among the target, the background, and the whole. The motion models, including the long-term and the short-term models, are proposed in this paper. The KCF, NCC, and the context model are used in our approach, while their merits are exploited and their shortcomings are overcome. The target and its context are interactive in the region occupied by the target. They are tracked independently, and fused into an ensemble using the motion model. Both the target tracker and the context tracker

help the tracker predict short-term motion, whereas the target tracker predicts long-term motion to recover from failure or target loss.

The KCF [12] is a popular baseline method with desirable performances, such as the high speed and the excellent tracking precision. However, the KCF is sensitive to the fast motion and the target loss. In our method, we use a short-term model and a long-term model to overcome the limitations. Also, the KCF approaches used in our method are updated in two ways to adapt the short-term model and the long-term model, respectively. The NCC [8] is a generative tracker with many drawbacks. For example, the NCC tracker is prone to drifting to similar object. But we extend the NCC to an efficient one that can track targets with different shapes in large regions densely and can be calculated with the FFT. We use the NCC to obtain probable candidates instead of the final tracking results. The motion improved KCF is then conducted to select the best candidate as the target.

The STC [37] also proposes a context model. However, the target is considered as the part of the context, and the context is trained to obtain the relationships between the target and the background. The STC uses the context to locate the target directly. The STC may be confused to discriminate the target from its context, although the context is helpful to locate the target, when the target is partially occluded. So we select the hollow context without the target to stress the independent effect of the background to help discriminate target from the similar objects. Furthermore, the context model will be unreliable when the target moves fast. Hence we pay more attention on the target model in that case.

As for the re-detection strategy for trackers, the TLD [25] firstly stresses the long-term tracking issue and uses a random forest to re-detect the target. The LCT [34] introduces a long-term model for the KCF. Our long-term model is different from that of the LCT. On the one hand, the LCT improves the KCF with a random ferns classifier to re-detect the target. The classifier will be inaccurate when similar objects appear surrounding the target. In our method, we use the generative NCC together with the KCF to re-detect the target. We focus the target and can mining all probable candidates in the whole image. On the other hand, in LCT, the KNN classifier is used to select reliable tracked results to construct the random ferns classifier and a new KCF tracker similar to DSST is used to select the target from the candidate targets. The candidate target is not precise when the target changes. But our method obtains candidate regions and uses a reliably updated KCF to search the target in the regions, so a precise output is gained.

III. Problem Formulation

Object tracking involves finding the most likely position of the target in frame t with knowledge from old frames:

$$\hat{\mathbf{x}}^{(t)} = \underset{\mathbf{x}\in\mathbf{I}^{(t)}}{\operatorname{argmax}} P\left(\mathbf{x} \mid \mathbf{I}^{(t)}, \boldsymbol{\theta}^{(t-1)}\right) \tag{3}$$

where $\hat{\mathbf{x}}^{(t)}$ denotes the predicted target position in frame t, $\mathbf{I}^{(t)}$ is the image of frame t, $\boldsymbol{\theta}^{(t-1)}$ represents the model parameters, and $\mathbf{x}$ is a candidate target.

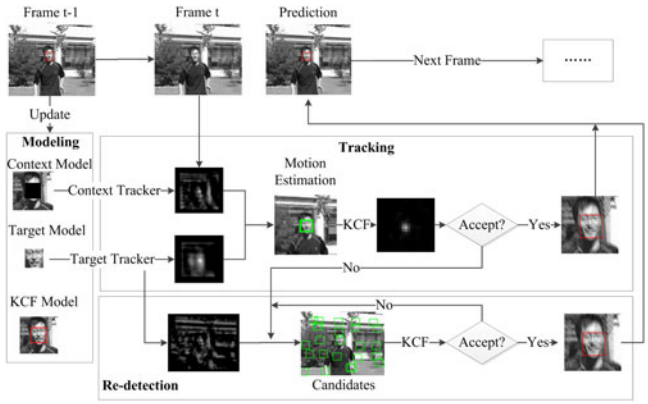

Fig. 3. Tracking Framework. The tracking is decomposed into modeling, tracking and re-detection. Models are trained and updated in modeling module. In tracking module, context tracker and target tracker jointly conduct a motion prediction and the target is precisely located by KCF tracker. Re-detection module starts when the target is detected as lost, target tracker finds proper candidates in the whole image. The candidates are validated by KCF and the best one is accepted as the target if it satisfies the predetermined conditions, otherwise the re-detection is continued in next frame.

The proposed tracking system is decomposed into three modules: modeling, tacking and re-detection (see Fig. 3). In the modeling module, features are extracted, processed, and composed into vectors to represent the target, the context, and the KCF samples, respectively. In other words, our framework researches the target using three models: M_T, M_C, and M_K represent the target model, the context model, and the KCF model, with parameters $\boldsymbol{\theta}_T$, $\boldsymbol{\theta}_C$, and $\boldsymbol{\theta}_K$, respectively.

In the tracking module, in order to adapt to the variety of backgrounds, context is locally tracked, and the result is used to predict the targets motion:

$$\hat{\mathbf{x}}_m^{(t)} = f_m\left(\hat{\mathbf{y}}_C^{(t)}, \hat{\mathbf{y}}_T^{(t)}\right) \tag{4}$$

where $\hat{\mathbf{x}}_m^{(t)}$ is the estimated target position after motion in a limited search region, $\hat{\mathbf{y}}_C^{(t)}$ is the predicted confidence map of the context tracker, $\hat{\mathbf{y}}_T^{(t)}$ is the predicted confidence map of the target tracker, and f_m is the tracking fusion function.

The rough prediction result is thought to be the new sampling center, so that a more appropriate sampling center can yield better KCF performance:

$$\hat{\mathbf{x}}_K^{(t)} = \underset{\mathbf{x}\in\mathbf{I}^{(t)}, SC(\mathbf{x})=\hat{\mathbf{x}}_m^{(t)}}{\operatorname{argmax}} \hat{\mathbf{y}}_K^{(t)} \tag{5}$$

where $\hat{\mathbf{y}}_K^{(t)}$ is the KCF response, $\hat{\mathbf{x}}_K^{(t)}$ is the predicted result of the KCF tracker, and $SC(\mathbf{x}) = \hat{\mathbf{x}}_m^{(t)}$ means that the sampling center of candidate $\mathbf{x}$ is $\hat{\mathbf{x}}_m^{(t)}$.

If the target is lost, the entire image is searched to determine where the target reappears. Candidates are proposed in the re-detection module. Their KCF responses are checked, and the candidate with the maximum response is considered the target:

$$\hat{\mathbf{x}}_R^{(t)} = \underset{\mathbf{x}\in f_R(\hat{\mathbf{y}}_T^{(t)})}{\operatorname{argmax}} \hat{\mathbf{y}}_K^{(t)}(\mathbf{x}) \tag{6}$$

where f_R is the candidate extract function, and $\hat{\mathbf{y}}_T^{(t)}$ is the target's confidence map in the entire image, which is calculated using the values of parameters from before the target is lost. $\hat{\mathbf{x}}_R^{(t)}$ is the predicted result of re-detection, and $\hat{\mathbf{y}}_K^{(t)}(\mathbf{x})$ is the KCF response in $\mathbf{x}$ when the samples center is in $\mathbf{x}$.

IV. Modeling, Tracking, and Re-Detection

In our method, three models are trained and updated with the given predicted data in the modeling module. The proposed tracker can locate the target by making a prediction concerning the upcoming frame in the tracking module. It is executed in the re-detection model, and begins recovery when tracking failure occurs. The KCF tracker in our tracking system is used for precise location and verification of the tracking result.

A. Modeling

The proposed method tracks the target and the context. Their tracking results are then fused for motion estimation of the target.

1) Target Model: A rectangle is commonly used to represent the region occupied by the target; hence the target is regarded as a matrix $\mathbf{T}$. Here, raw pixels are used to set-up the template vectors with zero mean:

$$M_T : \boldsymbol{\theta}_T = \mathbf{T} - \bar{\mathbf{T}} \tag{7}$$

where $\bar{\mathbf{T}}$ is the mean matrix of $\mathbf{T}$. We define $\bar{T}$ as the average value of all values in the matrix, $\bar{T} = \frac{1}{N_T}\sum_{i,j}\mathbf{T}_{i,j}$, where N_T is the number of elements in $\mathbf{T}$. $\bar{\mathbf{T}}$ is identical in size to $\mathbf{T}$, and the values of $\bar{\mathbf{T}}$ are $\bar{T}$: $\bar{\mathbf{T}} = [\bar{\mathbf{T}}_{i,j} = \bar{T}]$.

The target model describes how the candidates are similar to the target and generates them in re-detection. The target model needs to be updated to adapt to changes in the appearance of the target. However, the target model should strictly represent the target. The target is well tracked when $\hat{\mathbf{y}}_{\mathbf{K}}^{(t)}(\hat{\mathbf{x}}^{(t)}) \geq T_a$, namely the tracking result is reliable. Thus, the target model is updated slowly when a candidate is confirmed as target in case errors are learned during update:

$$\hat{\boldsymbol{\theta}}_T^{(t)} = \begin{cases} (1-\lambda_T)\hat{\boldsymbol{\theta}}_T^{(t-1)} + \lambda_T \boldsymbol{\theta}_T^{(t)}, & \text{if reliable} \\ \hat{\boldsymbol{\theta}}_T^{(t-1)}, & \text{Otherwise} \end{cases} \tag{8}$$

where λ_T is the update rate, T_a is a threshold to confirm whether the predicted candidate $\hat{\mathbf{x}}^{(t)}$ is the target, and $\hat{\mathbf{y}}_{\mathbf{K}}^{(t)}(\hat{\mathbf{x}}^{(t)})$ is the KCF response of the target.

2) Context Model: To discover the variation in the background and its effect on the target, we focus on temporary context, which is the most relevant part of the background with respect to the target.

Context is represented as a matrix of zero-mean raw pixels:

$$M_C : \boldsymbol{\theta}_C = \mathbf{C} - \bar{\mathbf{C}} \tag{9}$$

where $\mathbf{C}$ is a hollow matrix, which means the target region is not relevant to the context. In the NCC tracker, if $\boldsymbol{\theta}_C$ has zero mean, the hollow part can be filled with zeros. A visual interpretation of the hollow matrix is given in Fig. 4. Using matrix $\mathbf{D}$ as both target and context, and $\mathbf{D}_T$ as another matrix with the same size as $\mathbf{D}$ surrounding the target with zeros, $\mathbf{C} = \mathbf{D} - \mathbf{D}_T$, and $\bar{\mathbf{C}}$ can be calculated as:

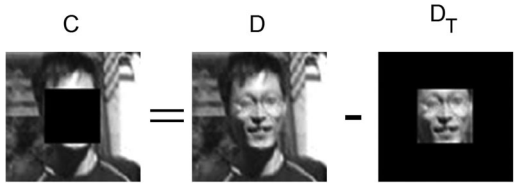

Fig. 4. An example of $\mathbf{C}$, $\mathbf{D}$, and $\mathbf{D}_T$.

$$\bar{\mathbf{C}} = \frac{N_D\bar{\mathbf{D}} - N_T\bar{\mathbf{T}}}{N_D - N_T} \tag{10}$$

where N_D is the number of elements in $\mathbf{D}$.

To capture real-time variation of the context, the context model is refreshed frame by frame.

3) KCF Model: The KCF model M_K is as explained in Section II Part D: $\boldsymbol{\theta}_K = (\boldsymbol{\alpha}, \mathbf{x}_K)$. In the proposed method, KCF parameters are trained frame by frame. In order to adapt to the target variations, the short-term KCF model is updated frame by frame. In the long-term tracking model, occlusion is taken into consideration. Re-detection is conducted when the target is lost, and tracking is recovered when the target is re-detected. The KCF model is utilized to check whether the tracking result is reliable and whether the target has been lost. The long-term KCF model is updated when the tracking results are reliable to make sure that the re-detection results are as well. As a result, the occurrence of errors re-detection can be reduced. The model is hence updated in two ways: $\hat{\boldsymbol{\theta}}_{K1}$ is continuously updated for smooth tracking, and $\hat{\boldsymbol{\theta}}_{K2}$ is strictly updated when the target is well tracked: $\hat{\mathbf{y}}_{\mathbf{K}}^{(t)}(\hat{\mathbf{x}}^{(t)}) \geq T_a$. With these different updates, two KCF detectors can be obtained:

$$\hat{\boldsymbol{\theta}}_{K1}^{(t)} = (1-\lambda_{K1})\hat{\boldsymbol{\theta}}_{K1}^{(t-1)} + \lambda_{K1}\boldsymbol{\theta}_{K1}^{(t)} \tag{11}$$

$$\hat{\boldsymbol{\theta}}_{K2}^{(t)} = \begin{cases} (1-\lambda_{K2})\hat{\boldsymbol{\theta}}_{K2}^{(t-1)} + \lambda_{K2}\boldsymbol{\theta}_{K2}^{(t)}, & \text{if reliable} \\ \hat{\boldsymbol{\theta}}_{K2}^{(t-1)}, & \text{Otherwise} \end{cases} \tag{12}$$

where λ_{K1} and λ_{K2} are update rates. The first detector is executed after construction of the short-term model to find the accurate position of the target, whereas the second detector is executed after construction of the long-term model to check the candidates and find the most appropriate target position.

B. Tracking

Usually the target is single-tracked to predict its position in the next frame. However, tracking is prone to drift if the tracker only focuses on the target. The background is found to be effective in the tracking process, especially when the target changes quickly. Considering that the target and the background can evolve independently, we track the target and the context in this order, and explore their relationships to integrate them in order to predict the motion of the target.

1) Target Tracker: The target tracker functions throughout the tracking process, including short- and long-term motion

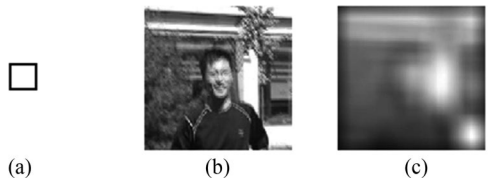

Fig. 5. An example of filtering (a) template $\mathbf{m}_T$, and (b) search region $\mathbf{R}$. (c) A diagram of the filter result.

model. As a generative tracker, the similarities between all candidates and the target should be calculated.

The normalized cross-correlation similarity between two matrices $\mathbf{x}$ and $\mathbf{z}$ of identical size is defined as follows:

$$\gamma(\mathbf{x},\mathbf{z}) = \frac{\sum_{i,j}(x_{i,j}-\bar{\mathbf{x}})(z_{i,j}-\bar{\mathbf{z}})}{\sqrt{\sum_{i,j}(x_{i,j}-\bar{\mathbf{x}})^2 \sum_{i,j}(z_{i,j}-\bar{\mathbf{z}})^2}} \tag{13}$$

where $x_{i,j}$ is the value at position (i,j) of matrix $\mathbf{x}$, and $\gamma(\mathbf{x},\mathbf{z})$ is in the range $[-1, 1]$. A larger value of $\gamma(\mathbf{x},\mathbf{z})$ means greater relevance between $\mathbf{x}$ and $\mathbf{z}$.

When the mean of $\mathbf{x}$ is zero, (13) can be simplified as:

$$\gamma(\mathbf{x},\mathbf{z}) = \frac{\sum_{i,j} x_{i,j}\cdot z_{i,j}}{\sqrt{\sum_{i,j} x_{i,j}^2}\sqrt{\sum_{i,j} z_{i,j}^2 - \frac{1}{N}(\sum_{i,j} z_{i,j})^2}} \tag{14}$$

where $\mathbf{x}$ has a shape template $\mathbf{m}$ that is a matrix containing ones, indicating pixels belonging to the object, and zeros, indicating pixels belonging to non-objects. N is the number of ones in $\mathbf{m}$.

To track an object in region $\mathbf{R}$ with template $\mathbf{x}$, the normalized cross-correlation similarities should be calculated for $\mathbf{x}$ and each $\mathbf{z}$ sampled in the position (u,v) of $\mathbf{R}$. Setting $\sigma_{\mathbf{x}} = \sqrt{\sum_{i,j}\mathbf{x}_{i,j}^2}$, defining $fi(\mathbf{A},\mathbf{B})$ as a filter operation on $\mathbf{B}$ with template $\mathbf{B}$, we can obtain $\sum_{i,j} x_{i,j}\cdot z_{i,j} = fi_{u,v}(\mathbf{x},\mathbf{R})$, $\sum_{i,j} z_{i,j}^2 = fi_{u,v}(\mathbf{m},\mathbf{R}\cdot\mathbf{R})$, and $\sum_{i,j} z_{i,j} = fi_{u,v}(\mathbf{m},\mathbf{R})$. The normalized cross-correlation coefficient can be obtained:

$$\gamma_{u,v}(\mathbf{x},\mathbf{R}) = \frac{fi_{u,v}(\mathbf{x},\mathbf{R})}{\sigma_{\mathbf{x}}\sqrt{fi_{u,v}(\mathbf{m},\mathbf{R}\cdot\mathbf{R}) - \frac{1}{N}fi_{u,v}^2(\mathbf{m},\mathbf{R})}} \tag{15}$$

In sum,

$$\gamma(\mathbf{x},\mathbf{R}) = \frac{fi(\mathbf{x},\mathbf{R})}{\sigma_{\mathbf{x}}\sqrt{fi(\mathbf{m},\mathbf{R}\cdot\mathbf{R}) - \frac{1}{N}fi^2(\mathbf{m},\mathbf{R})}} \tag{16}$$

where the fraction denotes element-wise division.

Three filter operations can be efficiently solved by using FFT in $O(N_R log N_R)$ time, where N_R is the number of elements in $\mathbf{R}$.

To track the target in region $\mathbf{R}$ with model $\hat{\boldsymbol{\theta}}_T$, the predicted confidence map of the target tracker $\hat{\mathbf{y}}_T$ can be calculated by:

$$\gamma(\hat{\boldsymbol{\theta}}_T,\mathbf{R}) = \frac{fi(\hat{\boldsymbol{\theta}}_T,\mathbf{R})}{\sigma_{\hat{\boldsymbol{\theta}}_T}\sqrt{fi(\mathbf{m}_T,\mathbf{R}\cdot\mathbf{R}) - \frac{1}{N_T}fi^2(\mathbf{m}_T,\mathbf{R})}} \tag{17}$$

An example of the filter operation $fi(\mathbf{m}_T,\mathbf{R})$ is shown in Fig. 5.

2) Context Tracker: The target is commonly assumed to move in a general mode, such as uniform speed, constant acceleration, and so on. This mode works well when the background

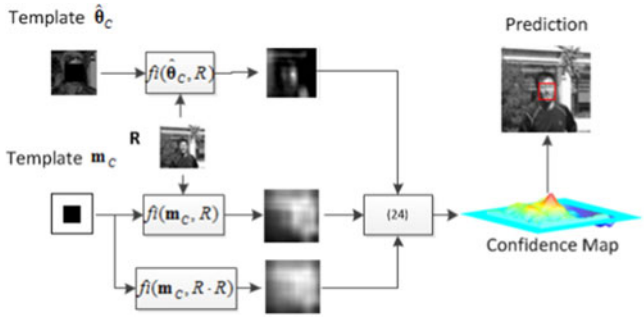

Fig. 6. The computation of the context tracker. $\gamma(\boldsymbol{\theta}_C,\mathbf{R})$ is shown as the confidence map, and the tracked context in the next frame is where the maximal value is in the confidence map.

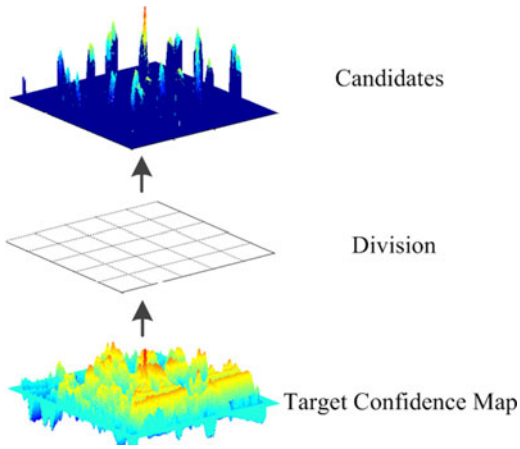

Fig. 7. A demonstration of the selection of candidates. Several exclusive candidates with local maximum values are selected after the division of the target confidence map.

is still. However, in practice, the background in a video is unstable, and can be affected by many factors, such as camera motion. It is thus necessary to model the motion of the background to adapt to background variation.

The context is tracked to help predict the short-term motion model for better target tracking. To track the context in region $\mathbf{R}$ with model $\hat{\boldsymbol{\theta}}_C$, by defining fi as filter operation, the deduction is similar to that of the target tracker (see Fig. 6), and the predicted confidence map of the context tracker $\hat{\mathbf{y}}_C$ is obtained by:

$$\gamma(\hat{\boldsymbol{\theta}}_C,\mathbf{R}) = \frac{fi(\hat{\boldsymbol{\theta}}_C,\mathbf{R})}{\sigma_{\hat{\boldsymbol{\theta}}_C}\sqrt{fi(\mathbf{m}_C,\mathbf{R}\cdot\mathbf{R}) - \frac{1}{N_C}fi^2(\mathbf{m}_C,\mathbf{R})}} \tag{18}$$

where $\mathbf{m}_C$ is identical to $\mathbf{D}$; and the values of elements in the context are 1, whereas those in the target are 0, $N_C = N_D - N_T$. It can be seen that the hollow context can be quickly tracked by using three filters with FFT.

3) Location: Using $\mathbf{R}^{(t)}$ as the context search region, the target tracker can make a pertinent prediction $\hat{\mathbf{y}}_T^{(t)}$ in (17), but is prone to failure as the search region increases in size. The context tracker that models the variation in the background has a positive reference value $\hat{\mathbf{y}}_C^{(t)}$ in (18). To predict the target with context, a motion model representing how the target is relative

TABLE I
THE AVERAGE VELOCITIES IN THE 15 VIDEOS ARE SHOWN, WHERE $\mathbf{v}_{truth}$ IS THE TARGET'S TRUE VELOCITY AND $\mathbf{v}_{prediction}$ THE TARGET'S PREDICTED VELOCITY BY OUR MODEL

velocity(pix/frame)	BlurBody	BlurOwl	Couple	Deer	Jumping	Tiger2	KiteSurf	Freeman1
$\mathbf{v}_{truth}$	20.20	13.35	7.37	10.79	2.61	11.32	2.69	1.97
$\mathbf{v}_{prediction}$	19.79	13.08	7.14	10.45	2.66	10.75	2.82	2.06
$\mathbf{v}_{relative}$	3.29	2.41	4.12	7.82	0.96	11.25	1.80	1.07
dv	2.88	2.38	2.61	2.91	1.38	3.83	3.56	1.93
	Freeman4	Human5	Human6	Human7	Singer1	Twinnings	Crowds	mean
$\mathbf{v}_{truth}$	1.66	1.67	3.38	6.92	1.47	1.23	1.69	5.89
$\mathbf{v}_{prediction}$	1.76	1.71	3.89	6.85	0.76	0.50	1.61	5.72
$\mathbf{v}_{relative}$	1.49	0.73	3.79	1.15	2.25	1.11	2.59	3.06
dv	2.01	0.32	2.57	2.65	2.61	1.95	2.10	2.38

$\mathbf{v}_{relative}$ is the target's velocity relative to the context, and dv is the change in the target's velocity relative to the context in continuous frames.

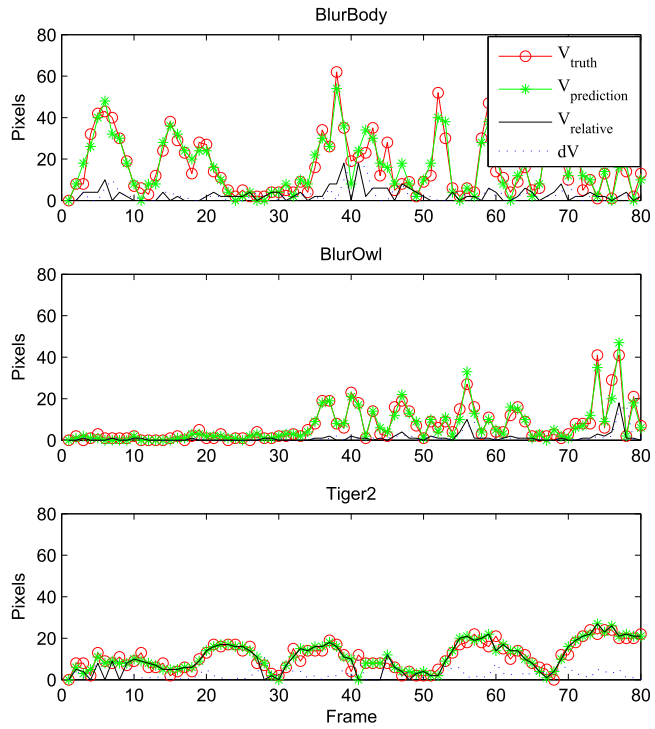

Fig. 8. Four real-time velocities in videos BlurBody, BlurOwl, and Tiger2 in frames 1 to 100.

to the context should be built. Under circumstances of fast motion, it is incorrect to assume that the target exhibits uniform linear motion relative to the image in successive frames because unpredictable variations arising in the background may make the targets motion unstable. Hence, many trackers cannot track quickly moving objects. However, it is reasonable to assume that the target moves smoothly relative to the local context in the object region. When the context undergoes displacement and the target is in uniform linear motion relative to the context in continuous frames, if the temporary velocity of the object in the last frame is $\mathbf{v}^{(t-1)}$, the confidence map of the target at position $\mathbf{x}$ is identical to the confidence map of the context at position $\mathbf{x} - \mathbf{v}^{(t-1)}\Delta t$.

The target model and the context model are two clues used to infer the motion of the target. Moreover, using T as the target model and C as the context model, we can observe that $T \cup C = R$ and $T \cap C = \phi$. The fusion results of the target and the context trackers are based on Bayesian inference. The likelihood of patch R as the position of the target can be expressed as:

$$\begin{aligned} f_m(\mathbf{x}) &= P_{\mathbf{x}\in\mathbf{R}}(\mathbf{x}|o) = \frac{P(\mathbf{x},o)}{P(o)} \\ &= \frac{1}{P(o)}(P(\mathbf{x},o,T) + P(\mathbf{x},o,C)) \\ &= \frac{P(\mathbf{x})}{P(o)}(P(o|\mathbf{x},T)P(T|\mathbf{x}) + P(o|\mathbf{x},C)P(C|\mathbf{x})) \end{aligned} \tag{19}$$

$P(o)$ is the normalization parameter. The prior probability is $P(\mathbf{x}) = \alpha e^{-\frac{|\mathbf{x}-\hat{\mathbf{x}}^{(t-1)}|^2}{\sigma^2}}$, which means that the target has a higher probability of appearing where the position is closer to the position of the target in the last frame. The prior confidence map $\hat{\mathbf{y}}_d(\mathbf{x}) = P(\mathbf{x})$ can be obtained. The posterior probabilities are the observation models of the target and the context, $P(T|\mathbf{x}) = \hat{\mathbf{y}}_T(\mathbf{x})$ and $P(T|\mathbf{x}) = \hat{\mathbf{y}}_C(\mathbf{x} - \mathbf{v}\Delta t)$. According to research on STC [37], $P(o|\mathbf{x},T)$ is the model's reliability from the view of the target model, and can be solved by hypothesizing that the likelihood $f_m(\mathbf{x})$ is subject to a special distribution. For simplicity, we set $P(o|\mathbf{x},T) = \max \hat{\mathbf{y}}_T$, since the maximum likelihood of the target tracker indicates the certainty degree. Similarly, $P(o|\mathbf{x},C) = \max \hat{\mathbf{y}}_C$. Consequently, the target model and the context model can be integrated by:

$$\hat{\mathbf{y}}_m^{(t)} = \alpha\hat{\mathbf{y}}_d(\lambda_m\hat{\mathbf{y}}_T^{(t)} + (1-\lambda_m)\hat{\mathbf{y}}_C^{(t)}(-\mathbf{v}^{(t-1)}\Delta t)) \tag{20}$$

where α is the normalization parameter, $\Delta t = 1$, and $\lambda_m = \frac{P(o|\mathbf{x},T)}{P(o|\mathbf{x},T)+P(o|\mathbf{x},C)}$. $\hat{\mathbf{y}}_T^{(t)}$ and $\hat{\mathbf{y}}_C^{(t)}$ are calculated using (17) and (18), respectively. $\hat{\mathbf{y}}_T^{(t)}$ and $\hat{\mathbf{y}}_C^{(t)}$ are in the range (0, 1], $\hat{\mathbf{y}}_m^{(t)}$ is the ensemble confidence map, and $\mathbf{v}^{(t)}$ is calculated as $\mathbf{v}^{(t)} = \frac{\hat{\mathbf{x}}^{(t)} - \hat{\mathbf{x}}_C^{(t)}}{\Delta t}$, where $\hat{\mathbf{x}}_C^{(t)}$ is calculated as $\hat{\mathbf{x}}_C^{(t)} = \operatorname{argmax} \hat{\mathbf{y}}_C^{(t)}$.

The predicted target position after motion is obtained as:

$$\hat{\mathbf{x}}_m^{(t)} = \operatorname{argmax} \hat{\mathbf{y}}_m^{(t)} \tag{21}$$

Samples used in the prediction of the new position of the KCF detector can increase resistance to background variation. In the KCF tracker, the new sample $\mathbf{z}_K^{(t)}$ is sampled in $\hat{\mathbf{x}}_m^{(t)}$ instead of

the initial $\hat{\mathbf{x}}^{(t-1)}$; here, the KCF detector with $\hat{\boldsymbol{\theta}}_{K1}^{(t-1)}$ is executed:

$$\hat{\mathbf{y}}_K^{(t)} = \hat{\mathbf{k}}^{\mathbf{x}_K^{(t)}\mathbf{z}_K^{(t)}} \cdot \hat{\boldsymbol{\alpha}}^{(t)} \tag{22}$$

Due to $\hat{\mathbf{x}}_m^{(t)}$, the performance of KCF can be improved, since its sensitivity to the target position in the search region is reduced, and it exploits the motion models. If the maximum response in $\hat{\mathbf{y}}_K^{(t)}$ surpasses threshold T_b, the output of the KCF tracker is accepted as the final prediction:

$$\hat{\mathbf{x}}^{(t)} = \operatorname{argmax} \hat{\mathbf{y}}_K^{(t)} \text{ if } \max \hat{\mathbf{y}}_K^{(t)} \geq T_b \tag{23}$$

If $\max \hat{\mathbf{y}}_K^{(t)} < T_b$, the target is considered lost because it is beyond region $\mathbf{R}^{(t)}$, and should be re-detected.

C. Re-Detection

Re-detection involves recovering the tracker when it fails. It is a temporary motion model for long-term tracking. When the candidates generated by the short-term motion model are not satisfactory, more candidates should be mined for in the extensive background.

Re-detection is conducted when $\max \hat{\mathbf{y}}_K^{(t)} < T_b$ in (22). The target tracker thus seeks candidates in the entire image and the responses $\hat{\mathbf{y}}_T^{(t)} = \gamma(\hat{\boldsymbol{\theta}}_T, \mathbf{I}^{(t)})$ where $\hat{\boldsymbol{\theta}}_T$ represents the values of the parameters before the target is lost and $\hat{\mathbf{y}}_T^{(t)}$ the positions where it might reappear.

Since it is not certain whether the tracker fails or the target generates significant deformation, false re-detection may lead to discovery failure. To ensure reliability and consider efficiency, the new candidates should have the following properties:

a) Similarity with the target template. The target template is a relatively reliable identifier, such that regions that are more similar to the template are more likely to be the target.
b) Sufficient in number, but not too many. Although the target template is strictly updated, it may not be accurate, and disturbers may exist in the background. This leads to a situation where the region with the greatest similarity may contain a similar object instead of the target. Hence, a sufficient number of candidates can increase the probability of successful re-detection. Considering that too many candidates reduce efficiency, the number should be set properly.
c) Far from one another. In order that the target can be detected with a limited number of candidates, the candidates should be far from one another because a candidate can represent nearby regions. It is easy to observe that regions close to one another, or ones that overlap, have similar similarities to the target template.

The proposed method segments confidence map $\hat{\mathbf{y}}_T^{(t)}$ into many regions and selects the one with the maximum local response as a candidate in each region. We simply divide the image into $n_r \times n_m$ even rectangles (see Fig 7):

$$f_R = \left\{ \mathbf{x} \mid x = \underset{x \in R(i,j)}{\operatorname{argmax}} \hat{\mathbf{y}}_T^{(t)}, 1 \leq i \leq n_r, 1 \leq j \leq m_r \right\} \tag{24}$$

where $R(i,j)$ is one of the $n_r \times n_m$ rectangles.

The re-detection should solve the inherent defects of the generative method. Hence, the KCF detector with $\hat{\boldsymbol{\theta}}_{K2}^{(t-1)}$ is used to check each candidate to determine whether is the target, as in (6). Discriminative detection on candidates obtained from generative methods can simultaneously exploit the advantages of both methods. The predicted position is not accepted unless the response reaches threshold T_r, and re-detection continues in next frame until the target is re-detected.

V. Experimental Results and Analysis

To evaluate the proposed method, two groups of experiments were conducted; one involving performance and characters analysis of the proposed method, and the other consisting of a comparative analysis with state-of-the-art methods. In the former, we analyzed the effects of the context tracker and the target tracker in short-term tracking, and investigated how the re-detection module works in long-term tracking. In the latter, the proposed Target and Background-underlined Tracking with Kernelized Correlation Filter (TBKCF) and the tracker without re-detection, called S-TBKCF, were compared with 12 trackers in OTB100 [4]. The 12 trackers were KCF [12], PoT [43], TPT [44], TLD [25], Struck [26], STC [37], CT [27], IVT [19], L1APG [38], LOT [18], ASLA [39], and DSST [31]. To assess the performance of TBKCF, three criteria were used: the overlap success (OS) rate at an overlap threshold of 0.5, the distance precision (DP) at a threshold of 20 pixels, and the center location error (CLE). A large value of OS or DP, or a small CLE, meant a more accurate result. We report the overall performance for one-pass evaluation (OPE). Our tracker ran at a speed of FPS $= 9.3$, and the short-term model was more efficient at a speed of FPS $=$ 10.6, where the environments were simulated using MATLAB R2012 on a 3.3 GHZ Intel Core i3 PC with 4 GB RAM. The videos were selected from a recent online tracking benchmark [3], [4]. The parameters of TBKCF were as follows: $\lambda_{k1} = 0.02$, $\lambda_{k2} = 0.02$, $\lambda_T = 0.08$, $T_a = 0.35$, $T_b = 0.25$, and $T_r = 0.33$. The search region size of our short-term model was 4.5 times the target size. The KCF tracker used HOG features with 31 bins, a detection region 2.5 times the target, and a Gaussian kernel. The context template was 2.5 times the target size, and the context tracker searched a region four times the target size. Among the parameters, T_a and T_b are two main parameters that make up the effect of the proposed long-term model, detailed discussions and experimental results on the two parameters are in the subsection "Long-term motion model evaluation".

A. Experiments on TBKCF Components

The two main contributions of TBKCF are the short-term tracking method, where the target and the context are tracked, respectively, and integrated to predict the target motion, and long-term re-detection. For short-term tracking, experiments

TABLE II
THE OS, DP, AND CLE OF THE S-TBKCF, T-TBKCF, AND KCF IN 15 VIDEOS FOR SHORT-TERM TRACKING

	OS(%)			DP(%)			CLE(pixels)		
	S-TBKCF	T-TBKCF	KCF	S-TBKCF	T-TBKCF	KCF	S-TBKCF	T-TBKCF	KCF
BlurBody	98.8	98.8	58.7	99.4	99.4	58.4	7.2	7.4	64.1
BlurOwl	98.9	99.8	22.8	99.1	100.0	22.8	5.3	4.3	183.4
Couple	72.9	72.9	24.3	100.0	100.0	25.7	4.8	5.8	47.2
Deer	100.0	100.0	81.7	100.0	100.0	81.7	4.6	4.9	21.3
Jumping	99.4	98.7	28.1	100.0	100.0	34.2	4.1	4.8	26.0
Tiger2	82.2	81.4	36.4	79.2	80.8	35.6	13.7	16.8	47.4
KiteSurf	96.4	44.1	31.0	100.0	46.4	33.3	3.3	59.6	40.8
Freeman1	23.3	16.6	16.0	95.7	39.9	40.2	8.3	99.1	94.6
Freeman4	21.9	18.4	18.4	96.8	83.0	53.4	7.6	13.2	26.9
Human5	34.2	34.2	23.6	99.6	99.6	26.5	6.9	7.4	175.5
Human6	22.6	22.6	22.5	40.8	40.4	29.0	92.2	37.8	107.7
Human7	38.4	34.4	40.8	100.0	99.2	47.2	6.9	9.0	48.2
Singer1	29.6	29.6	27.6	100.0	100.0	84.3	9.3	12.4	12.6
Twinnings	64.0	63.8	54.2	100.0	100.0	90.7	4.7	3.9	6.3
Crowds	92.5	95.4	99.7	100.0	100.0	100.0	5.0	6.4	3.1
Average	65.0	60.7	39.0	94.0	85.9	50.9	12.3	19.5	60.3

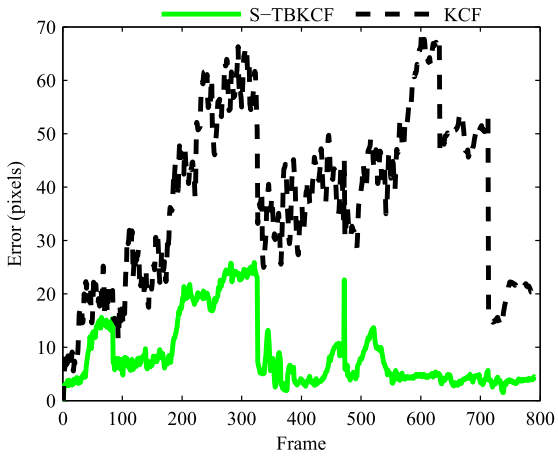

Fig. 9. The average error curves of real-time motion prediction results of the short-term S-TBKCF and the motion predicted by the original KCF in 15 videos.

were carried out using TBKCF without the re-detection module, and this was called S-TBKCF. In addition, short-term TBKCF without the context model, called T-TBKCF, was also tested. Specifically, the motion of the target relative to the context, the integration of the target and the context, the effect of the short-term model, and robustness against the division of the target and the background were evaluated in 15 videos, where the KCF exhibited fast motion. For long-term tracking, the experimental results of 10 videos were used where full occlusion or target loss occurred.

1) The Motion of the Target Relative to the Context: In our short-term motion model, we hypothesized that the target was in uniform linear motion relative to its context in continuous frames. To confirm the reliability of the hypothesis and the short-term motion model, four relevant velocities are listed in Table I. The target's velocity relative to the context was $\mathbf{v}_{\text{relative}}^{(t)} = \mathbf{v}^{(t)} = (\hat{\mathbf{x}}^{(t)} - \hat{\mathbf{x}}_C^{(t)})/\Delta t$, where $\hat{\mathbf{x}}_C^{(t)} = \operatorname{argmax} \hat{\mathbf{y}}_C^{(t)}$ was calculated using (20) and $\hat{\mathbf{x}}^{(t)}$ using (23). The change in the target's velocity relative to the context in continuous frames was $dv^{(t)} = v^{(t)} - v^{(t-1)}$. The target's true velocity in the video was $\mathbf{v}_{\text{truth}}^{(t)} = (\mathbf{x}_{\text{truth}}^{(t)} - \mathbf{x}_{\text{truth}}^{(t-1)})/\Delta t$, where $\mathbf{x}_{\text{truth}}^{(t)}$ was the true target position in the video. The target's predicted velocity by our model in the video was $\mathbf{v}_{\text{prediction}}^{(t)} = \mathbf{v}_T^{(t)} = (\hat{\mathbf{x}}^{(t)} - \hat{\mathbf{x}}^{(t-1)})/\Delta t$.

In 15 videos, the dv values were relatively small, and we obtained a mean of 2.38 pixels. This indicates that the target's velocity relative to the context was stable, despite the fact that it would change drastically. In addition, the values of $\mathbf{v}_{\text{prediction}}$ were similar to those of $\mathbf{v}_{\text{truth}}$, which proved that our short-term motion model was helpful in predicting the location of the target.

Moreover, real-time data for three videos featuring fast motion are shown in Fig 8. The three videos were shot at different attitudes. Severe view changes and a slow target were characteristic of the Blurbody video, and changing views and a stationary target were shown in the BlurOwl video; A static background and a quickly moving target were shown in Tiger2. In Blurbody and BlurOwl, we obtained small values of $\mathbf{v}_{\text{relative}}$, and in Tiger2, the values of $\mathbf{v}_{\text{relative}}$ were similar those of $\mathbf{v}_{\text{truth}}$. Real-time $\mathbf{v}_{\text{relative}}$ plots revealed that our motion model can reflect the true motion of the target relative to the context, which confirmed our motion model once again.

2) The Integration of the Target and the Context: Table II shows the OS, the DP, and the CLE of the S-TBKCF, T-TBKCF, and KCF in 15 videos. From Table II, we see that the T-TBKCF performed well, and using the context model, the S-TBKCF improved its precision. For example, in the videos BlurBody, Deer, and Human5, the CLE values were degraded due to the context model, although T-TBKCF obtained the same OS and DP with S-TBKCF. On occasions (KiteSurf, Freeman1, and Freeman4) when there were similar objects or severe background clutter in the surroundings of the target, the context model was able to discriminate the target effectively. Thus, with reasonable integration of the target tracker and the context tracker, our method can provide a reliable motion model with affordable error.

TABLE III
THE PERFORMANCE OF THE TBKCF, S-TBKCF, AND KCF IN 10 REPRESENTATIVE VIDEOS FOR LONG-TERM TRACKING

	OS(%)			DP(%)			CLE(pixels)		
	TBKCF	S-TBKCF	KCF	TBKCF	S-TBKCF	KCF	TBKCF	S-TBKCF	KCF
Box	84.0	33.5	35.8	89.1	39.4	41.5	11.6	107.4	89.1
Lemming	86.2	43.0	43.1	95.4	47.8	48.7	7.2	77.9	78.0
Shaking	92.9	1.4	1.4	94.8	1.6	1.9	8.0	115.1	113.2
Skating2-1	44.6	19.0	27.9	60.5	37.6	38.3	23.5	45.5	30.8
Jogging-1	96.4	22.5	22.5	97.7	23.1	23.5	5.7	90.8	87.9
Jogging-2	94.5	16.0	16.0	94.8	16.3	16.3	5.0	136.7	144.0
Board	96.3	85.8	85.4	77.5	65.0	65.6	15.5	35.6	35.5
DragonBaby	55.8	31.9	30.1	62.0	40.7	33.6	35.0	81.4	50.4
Football	85.9	79.0	68.2	99.2	79.6	79.6	5.8	13.3	14.8
Girl	96.6	81.8	75.6	100.0	94.0	86.4	2.6	7.7	11.9
Average	83.3	41.4	40.6	87.1	44.5	43.5	12.0	71.1	65.6

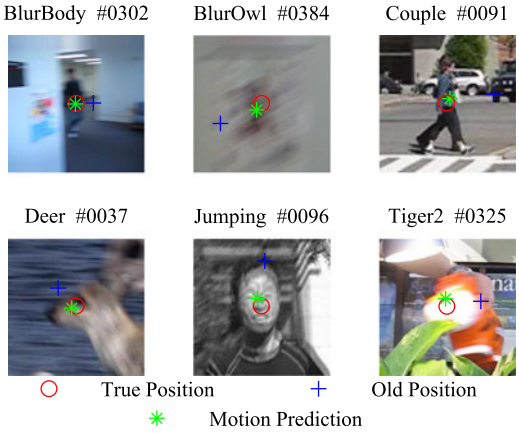

Fig. 10. Some fast motion cases in different videos. The true position is the target position in current frame. The old position is the target position in last frame. The motion prediction is the target position after motion predicted by the S-TBKCF.

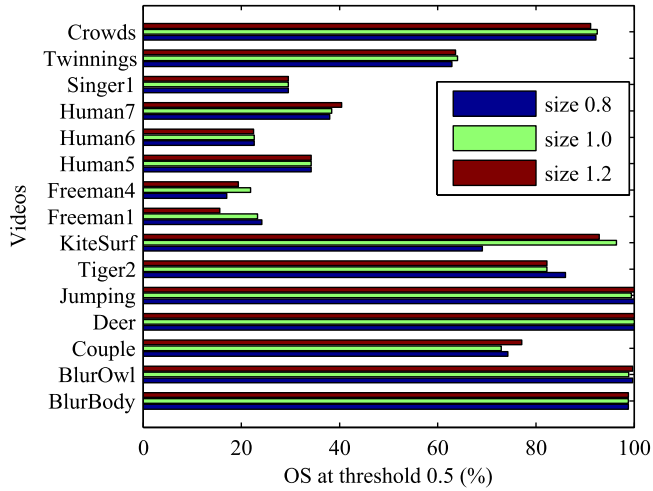

Fig. 11. The OS at threshold 0.5 of S-TBKCF with different sizes (0.8, 1.0, 1.2 times the original size) in 15 videos.

3) The Validity of the Short-Term Model: To confirm whether the novel model was effective at tracking, the S-TBKCF was compared with the original KCF tracker. Table II reveals that our short-term model can help KCF achieve outstanding performance, since the average OS was enhanced from 39% to 65% and the average DP from 50.9% to 94%. To observe the real-time effectiveness of our short-term model, the average real-time motion prediction errors were analyzed. Fig. 9 shows the average error curve $e_{\mathrm{TBKCF}}^{(t)} = ||\mathbf{x}_{\mathrm{truth}}^{(t)} - \hat{\mathbf{x}}_{m}^{(t)}||$ for real-time motion prediction for the 15 videos, where $\hat{\mathbf{x}}_{m}^{(t)}$ was as in (4) and $\mathbf{x}_{\mathrm{truth}}^{(t)}$ was the target's true position in frame t; the error curve $e_{\mathrm{KCF}}^{(t)} = ||\mathbf{x}_{\mathrm{truth}}^{(t)} - \hat{\mathbf{x}}_{\mathrm{KCF}}^{(t-1)}||$ of the motion is as hypothesized by the original KCF, where $\hat{\mathbf{x}}_{\mathrm{KCF}}^{(t)}$ is the predicted tracking result of the KCF tracker in frame t. The real-time average motion prediction errors of the S-TBKCF are smaller than those of the KCF, so our short-term model has a better motion prediction.

Fig. 10 shows some cases of fast motion. It can be seen that the target moves a great distance from the old position in last frame. Our predicted positions were close to the true position in current frame. Owing to the accurate short-term motion model, we achieved the impressive tracking performance at last. Without the short-term motion model, the KCF tracker would fail in these cases. Hence, an accurate short-term motion model is necessary and effective for long-term tracking.

4) Robustness of the Target and Background Against Segmentation: The target and the background were tracked respectively in TBKCF. To confirm whether the TBKCF needed accurate segmentation of the target and the background, it was tested with different disturbances in target size. Figs. 11 and 12 show the performance of the S-TBKCF with different target sizes in the 15 videos with respect to OS and DP. In most videos, the variations in target size had little effect on our tracker. However, in special videos, for example, in the video Freeman4, the tracker failed with a target size of 0.8 because background clutter was severe. This showed that more precise segmentation can yield better performance. The motion prediction error curves of the TBKCF with different target sizes in frames 1 to 80 of Couple are given in Fig. 13. The real-time plots indicate that the S-TBKCF predicted similar motion model results in the three situations in short-term tracking. To summarize, our tracker is robust against the segmentation of the target and the context,

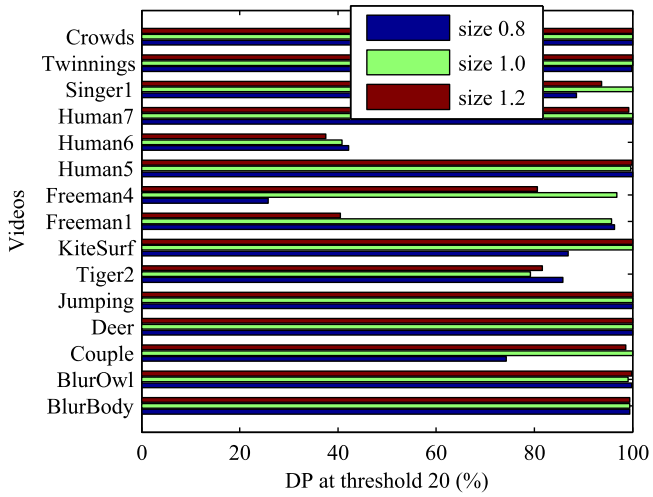

Fig. 12. The DP at threshold 20 of S-TBKCF with different sizes (0.8, 1.0, 1.2 times the original size) in 15 videos.

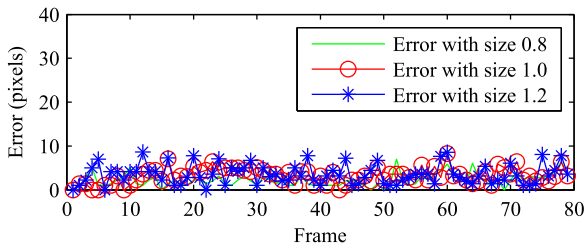

Fig. 13. The motion prediction error curves of TBKCF with different target sizes (0.8, 1.0, and 1.2 times the original size) in the video Couple (frames 1-80).

although more correct segmentation contributes to more reliable models.

5) Long-Term Motion Model Evaluation: In our TBKCF, the long-term motion model is executed when the target is lost. Table III gives the OS, the DP, and the CLE of the TBKCF, the S-TBKCF, and the KCF in 10 representative videos. When full occlusion or target loss occurred, the short-term S-TBKCF improved the KCF by little or not at all, such as when the OS increased from 40.6% to 41.4%. However, our long-term model contributed to excellent performance, since the OS was increased from 41.4% to 83.3%. To confirm how the long-term motion model works, the prediction error curves of the TBKCF and the KCF for Jogging-2 are shown in Fig. 14. In this video, the background was simple, and one of the main challenges was that the target was fully occluded in frame 46, and this occlusion lasted for many frames. Most trackers without a re-detection module failed to track the target. As we can see from the figures, using our TBKCF, the target was re-detected in frame 65, whereas the KCF tracker never recovered once the target was lost. Benefiting from the long-term motion model, the average tracking errors decreased drastically. As mentioned earlier, T_a and T_b are significant parameters for the long term model. In tracking, we must consider the target variations and error tracking results, simultaneously. The former should be included in the models, whereas the latter shouldn't be updated. In our approach, we use the maximum response value to determine whether the tracking results are reliable and whether the

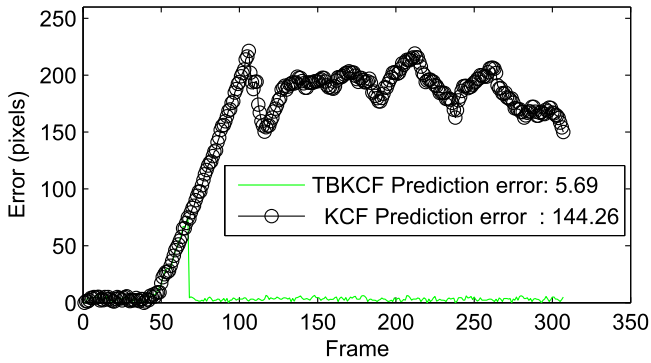

Fig. 14. The prediction error curves of TBKCF and KCF for video Jogging-2 (308 frames). The legend shows the average errors (in pixels).

TABLE IV
THE AVERAGE DISTANCE PRECISION SCORES IN 10 REPRESENTATIVE VIDEOS WITH DIFFERENT PARAMETERS

DP(%)		T_a		
		0.3	0.35	0.4
T_b	0.2	69	71.4	70
	0.25	82.8	83.8	74
	0.3	81.5	77.2	72.1

target is lost. However, both situations will make the maximum response value decrease. Namely, a greater value of T_a will reduce the error detection rate in long-term model, but the model may not adapt the variations of the target. Similarly, a greater value of T_b will find the loss of the target timely, but may regard the target variation as the target loss. Hence, we select the suitable T_a and T_b values with extensive tests to achieve compromises. Some experimental results are exhibited in Table IV. We have compared the effect of different T_a and T_b values, and the DP score peaks at $T_a = 0.35$, $T_b = 0.25$. Hence we adopt $T_a = 0.35$, $T_b = 0.25$ at last.

B. Experiments for Overall Performance, and Comparative Analysis

We report the overall performance for one-pass evaluation (OPE) in 100 videos [4]. Table V gives the average performance, including the area-under-the-curve (AUC) of the OS, the DP at a threshold of 20 pixels, the CLE and the speed. Our TBKCF yielded the best performance with an OSAUC of 53.6%, a DP of 79.4% and a CLE of 31.8 pixels. The short-term model S-TBKCF also performed well with an OSAUC of 50.5%, a DP of 74.6% and a CLE of 39.2. In the other trackers, the TPT [44] obtained a highest OSAUC of 49.4%, and the PoT [43] yielded a highest DP of 72.3% and a CLE of 35.4. As for the speed, KCT and STC are of high speed with 243 and 653 fps, respectively. But their tracking precision remains to be raised. The speed of our method is not the most fast but acceptable. Although tracking the target and context with large regions in the short-term model costs a lot of time, we gain rewards in tracking precision, as the DP score is increased from 69.2% to 74.6% using the short-term model. Also, the re-detection needs

TABLE V
THE AVERAGE PERFORMANCE IN OTB100

	TBKCF	S-TBKCF	KCF [12]	PoT [43]	TPT [44]	TLD [25]	Struck [26]
OS(%)	**53.6**	50.5	47.5	48.6	49.4	42.6	46.2
DP(%)	**79.4**	74.6	69.2	72.3	64.6	59.5	63.9
CLE(pixel)	**31.8**	39.2	45.0	35.4	50.0	59.9	46.9
Speed(FPS)	9.3	10.6	243.5	0.3	0.1	23.4	9.8
	STC [37]	CT [27]	IVT [19]	L1APG [38]	LOT [18]	ASLA [39]	DSST [31]
OS(%)	31.9	28.4	31.6	35.8	33.9	40.9	47.5
DP(%)	50.7	36.4	43.1	46.4	46.9	51.4	69.5
CLE(pixel)	90.5	79.8	88.0	82.9	65.3	67.9	47.8
Speed(FPS)	653.3	44.4	26.9	1.8	0.5	7.3	40.9

The best ones are shown in red and bold.

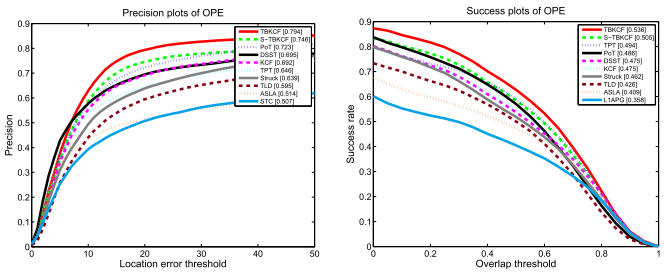

Fig. 15. The distance precision and overlap success plots of OPE in OTB100. The performance of the top 10 trackers is displayed. The legends show the average DP scores at a location error threshold of 20 and the OS AUC scores.

to track the target in the whole image, so an extra amount of computation is generated. But impressive accuracy in long-term tracking is achieved in return.

Fig. 15 illustrates the OS and the DP plots. The legends show the OSAUC and the DP at a threshold of 20 pixels. With KCF as the baseline of our tracker, the short-term model improved KCF by 5.4% in terms of DP score and 3% in terms of OSAUC. Moreover, the long-term model further improved the S-TBKCF by 4.8% in terms of DP score and 3.1% in terms of OSAUC.

The 100 videos in the benchmark were annotated with 11 attributes representing challenges in each video. The attributes were illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), out of view (OV), background clutter (BC), and low resolution (LR). Since only 9 videos are under low resolution, we focus on the other 10 attributes. The results of DP for the challenging attributes are shown in Fig. 16. Among the methods for comparison, TBKCF exhibited the best performance in terms of distance precision in fast motion (72.5%), background clutter (79.8%), motion blur (71%), deformation (73.5%), illumination variation (80.4%), in-plane rotation (78.1%), occlusion (74.9%), and scale variation (72.7%). In the videos featuring out-of-view, TBKCF yielded impressive performance with a DP score (59.7%) slightly lower than that of the PoT (63.9%). Since the videos may share different attributes, we compare the tracking results of our method in different attributes. As our method has the average DP score of 79.4% in the whole 100 videos, our method works well in background clutter and illumination variation, but has limitations in other attributes. Because the average DP scores in background clutter and illumination variation exceed the average scores in the whole benchmark. In most videos, the target deforms or scale verifies slowly, so our method can track the target well in terms of deformation and scale variations with update strategy. In cases of severe deformations and scale variations, our tracker may lose the target, but can re-detect the target when it recovers owing to our re-detection module. We can find the re-detection helps in the two cases as the DP scores of the methods from the S-TBKCF to TBKCF are from 68.3% to 73.5% in term of deformation and from 68.4% to 72.7% in term of scale variation.

Since we did not introduce any scale adaptation mechanism, the performance might have been influenced by scale variation. For this reason, we show the tracking performance in the 36 videos with no scale variation in TB100. As shown in Fig. 17, our TBKCF achieved outstanding performance with a DP score of 91.3%, which indicates that the TBKCF can track objects successfully when no scale variation occurs.

Moreover, to visually represent the performance of our tracker, a few images of five trackers (TBKCF, KCF [12], TLD [25], Struck [26], and DSST [31]) in eight videos are given in Fig. 18. There are four from each video, the first two showing short-term tracking results and the latter two the long-term tracking results. Our TBKCF performed well both in short-term and long-term tracking. In scenes with fast motion due to the background (BlurBody, BlurOwl), the target (Jumping), or both (Shaking), our TBKCF decomposed the motion into the motion of the background and that of the target relative to the background. Owing to the integration of the target and the local context, good performance was obtained. The KCF did well when the target moved smoothly, but drifted when fast motion occurred. DSST increased resistance to quick motion by optimizing the KCF with a multi-scale method, but still failed over time. TLD used optical flow for short-term tracking, but failed frequently even though it tracked well. Thus, the short-term tracking results were unsatisfactory. In short-term tracking, Struck worked as well as our TBKCF, but has a complex design, and no re-detection ability. In terms of occlusion (Box, Jogging-2, Tiger2), our TBKCF was able to re-detect the target successfully even after long-term occlusion. TLD has excellent re-detection, but false detection may occur in complex scenes or background clutters, as show in Fig. 18 (#350 in Tiger2 and #355 in Shaking). The short-term motion model of TBKCF takes

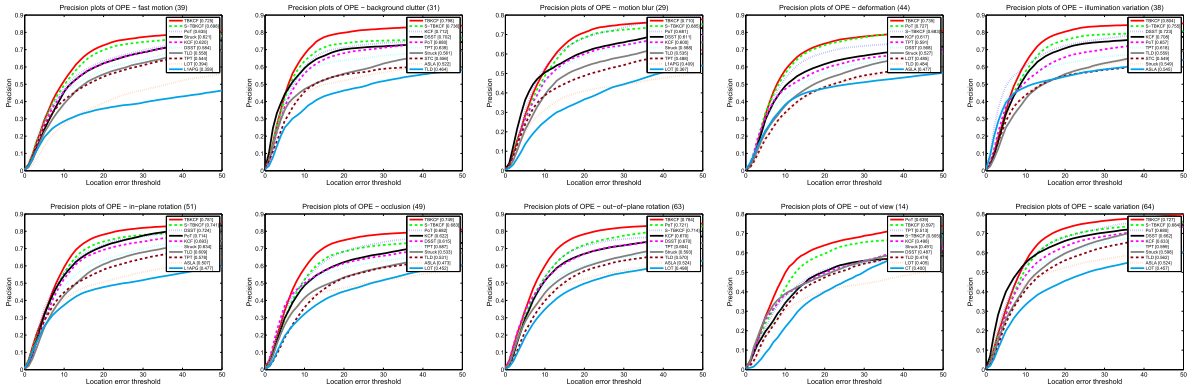

Fig. 16. Performance of 14 trackers in 10 challenging attributes. Only the distance precision plots of the top 10 trackers are presented. The legend contains the DP score at a threshold of 20 for each tracker in OPE.

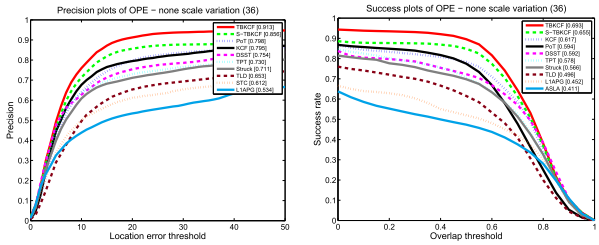

Fig. 17. The distance precision and overlap success plots of OPE in 36 videos with no scale variation. The performance of the top 10 trackers are displayed. The legends show the average DP scores at a location error threshold of 20 and the OS AUC scores.

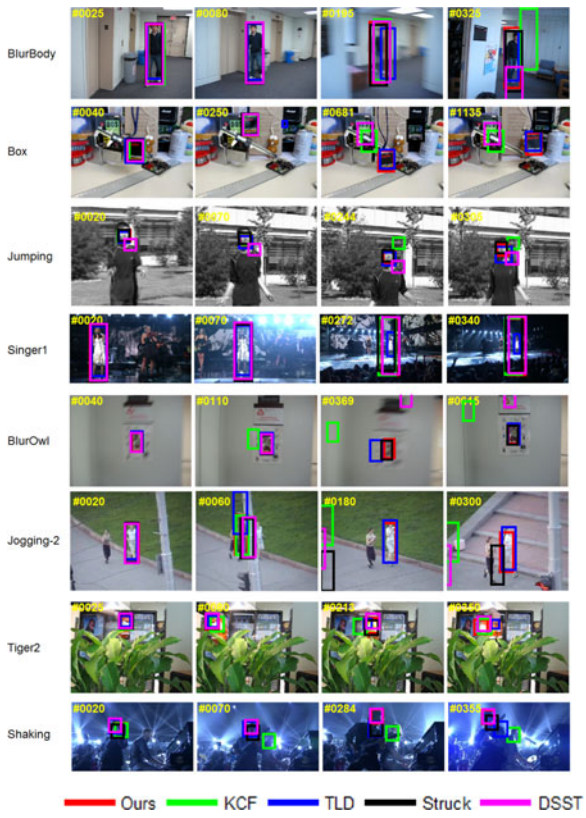

Fig. 18. Tracking pictures of TBKCF, KCF, TLD, Struck, and DSST on eight videos.

advantages of the context, and can provide relatively accurate target position (Singer1), or can re-detect the target when its scale recovers.

VI. Conclusion

The context of the target contains useful information for object tracking. By modeling the target and its context, respectively, we proposed a method that independently tracks them and then fuses the tracking results. The proposed method is robust against the segmentation of the target and its context. The movement of the target relative to its context is stable, even if the target moves quickly in the video. The parameters of the target and its context models are updated online. The target models update before loss of target was used in re-detection to predict the position of the target in videos, so that object tracking could recover from failure. Our method effectively improves robustness in long-term tracking.

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surveys*, vol. 38, no. 4, pp. 81–93, 2006.

[2] A. W. Smeulders *et al.*, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

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

[4] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[5] H. Yang, L. shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *Neurocomput.*, vol. 74, no. 18, pp. 3823–3831, 2011.

[6] N. Wang, J. shi, D. Y. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 3101–3109.

[7] X. Li *et al.*, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, pp. 478–488, 2013.

[8] U. D. Hanebeck, "Template matching using fast normalized cross correlation," *Proc. SPIE*, vol. 4387, pp. 95–102, 2001.

[9] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1834–1843, Apr. 2015.

[10] N. Wang and D. Y. Yeun, "Learning a deep compact image representation for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 809–817.

[11] Z. Chen, Z. Hong, and D. Tao, "An experimental survey on correlation filter-based tracking," *Comput. Sci.*, vol. 53, no. 6025, pp. 68–83, 2015.

[12] J. Henriques, C. Rui, 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, Mar. 2015.
[13] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4310–4318.
[14] Q. Liu, X. Zhao, and Z. Hou, "Survey of single-target visual tracking methods based on online learning," *IET Comput. Vis.*, vol. 8, no. 5, pp. 419–428, Oct. 2014.
[15] Y. Yuan, H. Yang, and Y. Fang, "Visual object tracking by structure complexity coefficients," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1125–1136, Aug. 2015.
[16] B. Ma, J. Shen, and Y. Liu, "Visual tracking using strong classifier and structural local sparse descriptors," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1818–1828, Oct. 2015.
[17] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *Int. J. Comput. Vis.*, vol. 56, no. 3, pp. 221–255, 2004.
[18] S. Oron, A. Bar-Hillel, L. Dan, and S. Avidan, "Locally orderless tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1940–1947.
[19] D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125–141, 2008.
[20] X. Mei, H. Ling, Y. Wu, and E. Blasch, "Minimum error bounded efficient ℓl 1 tracker with occlusion detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1257–1264.
[21] D. Lu, L. Li, and Q. Yan, "Survey: Target tracking algorithm based on sparse representation," in *Proc. Int. Symp. Comput. Intell. Des.*, 2008, vol. 2, pp. 195–199.
[22] N. Ahuja, S. Liu, B. Ghanem, and T. Zhang, "Robust visual tracking via multi-task sparse learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2042–2049.
[23] Y. Zhou, X. Bai, W. Liu, and L. Latecki, "Similarity fusion for visual tracking," *Int. J. Comput. Vis.*, vol. 118, no. 3, pp. 1–27, 2016.
[24] 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.
[25] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.
[26] S. Hare, A. Saffari, and P. Torr, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, Oct. 2016.
[27] K. Zhang, L. Zhang, and M. H. Yang, "Fast compressive tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 2002–2015, Oct. 2014.
[28] D. Bolme, J. Beveridge, B. Draper, and Y. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2544–2550.
[29] J. Henriques, C. Rui, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 702–715.
[30] M. Danelljan, F. Khan, M. Felsberg, and J. Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1090–1097.
[31] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–11.
[32] T. Liu, G. Wang, and Q. Yang, "Real-time part-based visual tracking via adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4902–4912.
[33] S. Liu, T. Zhang, and X. Cao, "Structural correlation filter for robust visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4312–4320.
[34] C. Ma, X. Yang, C. Zhang, and M. H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5388–5396.
[35] M. Danelljan, G. Hager, F. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. Int. J. Comput. Vis. Workshop*, 2015, pp. 621–629.
[36] M. Kristan *et al.*, "The visual object tracking VOT2015 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 564–586.
[37] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M. H. Yang, "Fast visual tracking via dense spatio-temporal context learning," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 127–141.
[38] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1830–1837.
[39] H. Lu, X. Jia, and M. H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1822–1829.
[40] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1269–1276.
[41] C. Leistner, H. Grabner, and H. Bischof, "Semi-supervised boosting using visual similarity learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[42] B. Ma, L. Huang, and J. Shen, "Visual tracking under motion blur," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5867–5876, Dec. 2016.
[43] B. Ma, H. Hu, and J. Shen, "Generalized pooling for robust object tracking," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4199–4208, Sep. 2016.
[44] B. Ma, L. Huang, and J. Shen, "Discriminative tracking using tensor pooling," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2411–2422, Nov. 2015.
[45] B. Ma, H. Hu, and J. Shen, "Linearization to nonlinear learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4400–4407.

Chang Liu received the Bachelor's degree in computer science and technology from Harbin Institute of Technology, Harbin, China, in 2014. He is currently working toward the Ph. D. degree in the College of Computer Science and Technology, Harbin Institute of Technology. His research interest focuses on computer vision and pattern recognition.

Peng Liu received the Ph.D. degree in microelectronics and solid state electronics from Harbin Institute of Technology, Harbin, China, in 2007. He is currently an Associate Professor in the College of Computer Science and Technology, Harbin Institute of Technology. His research interests include image processing, computer vision, and pattern recognition.

Wei Zhao received the Ph.D. degree in computer application technology from Harbin Institute of Technology, Harbin, China, in 2006. He is currently an Associate Professor in the College of Computer Science and Technology, Harbin Institute of Technology. Her research interests include computer vision and pattern recognition.

Xianglong Tang received the Ph.D. degree in computer application technology from Harbin Institute of Technology, Harbin, China, in 1995. He is currently a Professor in the College of Computer Science and Technology, Harbin Institute of Technology. His research interest focuses on pattern recognition.