

Locator-Checker-Scaler Object Tracking Using Spatially Ordered and Weighted Patch Descriptor

Han-UI Kim, *Student Member, IEEE*, and Chang-Su Kim, *Senior Member, IEEE*

Abstract—In this paper, we propose a simple yet effective object descriptor and a novel tracking algorithm to track a target object accurately. For the object description, we divide the bounding box of a target object into multiple patches and describe them with color and gradient histograms. Then, we determine the foreground weight of each patch to alleviate the impacts of background information in the bounding box. To this end, we perform random walk with restart (RWR) simulation. We then concatenate the weighted patch descriptors to yield the spatially ordered and weighted patch (SOWP) descriptor. For the object tracking, we incorporate the proposed SOWP descriptor into a novel tracking algorithm, which has three components: locator, checker, and scaler (LCS). The locator and the scaler estimate the center location and the size of a target, respectively. The checker determines whether it is safe to adjust the target scale in a current frame. These three components cooperate with one another to achieve robust tracking. Experimental results demonstrate that the proposed LCS tracker achieves excellent performance on recent benchmarks.

Index Terms—Visual tracking, object tracking, bounding box descriptor, discriminative tracker, and tracking with multiple estimators.

I. INTRODUCTION

GIVEN an initial state (or bounding box) of a target object in a frame, the objective of visual tracking is to estimate its states in subsequent frames. Visual trackers can be employed in a wide variety of image processing and computer vision applications, including surveillance systems, self-driving cars, and human-computer interface. Many efforts have been made for visual tracking, and recent advances in pattern recognition and machine learning have enabled the design and implementation of sophisticated tracking systems, called discriminative trackers (or tracking-by-detection systems).

In general, a discriminative tracker models the appearance of a target object using a classifier. It detects the target object in a series of frames sequentially, while updating the classifier using tracking results. Positive and negative samples,

corresponding to foreground and background image regions respectively, are employed to train the classifier. However, when training samples are assigned false labels or when sample descriptors cannot clearly distinguish between positive and negative samples, the classifier may be corrupted, degrading the tracking performance. To overcome these drawbacks, various algorithms have been proposed to reduce the impacts of the false labeling [1]–[3] and to describe positive and negative samples more distinguishably [4]–[6]. However, it is still challenging to develop a reliable object descriptor, since the bounding box of a target object often contains background features due to object deformation, occlusion, and size variation. Attempts have been made to alleviate the effects of undesirable background features [4], [7], but these approaches require prior knowledge or predefined parameters, the validity of which is hard to verify.

In this work, we propose a novel descriptor, called spatially ordered and weighted patch (SOWP), to represent the appearance of an object faithfully and suppress background information in a bounding box systematically. To construct the SOWP descriptor, we divide the bounding box of a target object into patches and describe each patch using color histograms and a gradient histogram. We then concatenate the patch descriptors within the bounding box to convey structural information of the object. Moreover, to reduce the effects of background, we scale each patch descriptor with a weight, which represents the importance of the patch in the object description. For the patch weight computation, we perform random walk with restart (RWR) simulation [8] twice: one for foreground clustering and the other for background clustering. Experimental results show that even a simple tracker based on the SOWP description provides competitive results in a recent tracking benchmark [9].

Another important issue in visual tracking is to adapt to target scale variations. Even though many trackers deal with scale variations [10]–[12], it still remains a challenging problem. When the appearance of a target is distorted by object deformation and occlusion, a tracker often estimates the scale of the target unreliably. Incorrect scale estimates, in turn, cause a model drift, eventually resulting in a tracking failure. Therefore, it is essential to develop robust scale estimation techniques.

In this work, for robust scale estimation, we present a novel tracking algorithm that consists of three components: locator-checker-scaler (LCS). The locator and the scaler estimate the state of a target object. Specifically, the locator estimates the center location of a target using a fixed scale, and then the scaler adjusts the target scale at the estimated center position.

Manuscript received August 12, 2016; revised February 13, 2017 and April 11, 2017; accepted May 5, 2017. Date of publication May 18, 2017; date of current version June 7, 2017. This work was supported in part by the National Research Foundation of Korea (NRF) through the Korea Government (MSIP) under Grant NRF-2015R1A2A1A10055037, and in part by the MSIP, Korea, through the ITRC support program supervised by the Institute for Information & communications Technology Promotion under Grant IITP-2017-2016-0-00464. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Nilanjan Ray. (*Corresponding author: Chang-Su Kim.*)

The authors are with the School of Electrical Engineering, Korea University, Seoul 136-701, South Korea (e-mail: hanulkim@mcl.korea.ac.kr; changsukim@korea.ac.kr).

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/TIP.2017.2706064

However, the scale estimation is unreliable in case of object deformation and occlusion. Thus, we adopt the checker to identify safe frames for the scale adjustment. By adapting scales only in safe frames, the proposed tracker achieves reliable tracking. Experiments show that the proposed tracker outperforms the recent trackers in [11] and [13]–[19] on the benchmark [9]. Moreover, the proposed tracker also provides excellent performance on other benchmarks [20], [21].

In summary, this work has the following contributions.

- We develop the simple yet effective SOWP descriptor. It employs the RWR-based adaptive weighting to suppress the background information in a bounding box.
- We propose the LCS tracker, which incorporate the locator, the checker, and the scaler to achieve robust tracking.
- The proposed algorithm yields outstanding performance on the recent benchmarks [9], [20], [21] and outperforms the trackers in [11], [13]–[17], and [19].

This work extends the conference paper in [22] as follows.

- We provide additional experiments to analyze the SOWP descriptor [22]. Specifically, we incorporate the descriptor into a more sophisticated tracker and analyze its efficacy by performing more extensive experiments.
- We propose the LCS algorithm to address target scale variations, and evaluate it extensively.

The remainder of this paper is organized as follows: Section II reviews related work. Section III explains the proposed SOWP descriptor, and Section IV describes the proposed LCS tracker. Section V discusses experimental results. Finally, Section VI draws conclusions.

II. RELATED WORK

This section briefly reviews related work in visual tracking, which is relevant to the proposed algorithm. For a comprehensive review of visual tracking, we refer the readers to recent surveys in [9], [20], [23], and [24].

A. Bounding Box Description

In visual tracking, a target object is located using a bounding box, which is described by a feature vector. This bounding box description influences the tracking performance significantly [25]. Various attempts have been made to design effective low-level features for tracking. For instance, Comaniciu *et al.* [4] adopted a color histogram in their mean-shift tracker. Some trackers [11], [13], [26] use histograms of oriented gradients (HOGs) [27] to encode local edge information. Also, Haar-like features [28] and local binary patterns [29] are employed to exploit texture information in visual tracking [1], [3], [30]. Furthermore, Li *et al.* [31] combined intensity histograms, local binary pattern histograms, gradient histograms, and Haar-like features, based on a random decision tree, for discriminative description. Chen *et al.* [6] utilized both intensity and gradient information. Similarly, many trackers [15], [16], [32] combine multiple diverse features to describe bounding boxes effectively.

The aforementioned descriptors, however, may become unreliable when a bounding box includes background information. Thus, researches have been carried out to reduce the

effects of background pixels. In [4] and [7] pixels near the center of a bounding box are emphasized with bigger weights since they are more likely to contain foreground information than boundary pixels are. However, these schemes do not take into account an object shape. They may yield erroneous results when a target object has a complex shape or is occluded. In this work, we also attempt to suppress background pixels, but, in contrast to [4] and [7], the proposed SOWP descriptor in Section III uses a weight adaptation method to consider a target shape.

Recently, instead of traditional hand-crafted features, several trackers have adopted neural networks to extract features from bounding boxes. Wang and Yeung [33] used a stacked denoising autoencoder [34] for the bounding box description. Wang *et al.* [35] proposed a domain adaptation method to adjust a pre-trained network for representing a specific target. Ma *et al.* [17] used a deep convolutional neural network [36] to generate a feature map for the correlation filtering [13]. Nam and Han [10] proposed the multi-domain convolutional neural network, which was pre-trained to obtain generic representation for visual tracking. The Nam and Han's tracker achieves outstanding results on the benchmark [9]. In [37] and [38] Siamese networks are used to measure the similarity between target and candidate boxes.

B. Discriminative Tracking

Trackers can be classified into either generative or discriminative ones in general [2], [39]. A generative tracker models the target appearance itself and then locates the target by searching for a candidate similar to the model. On the other hand, a discriminative tracker trains a classifier to separate the target from the background and then determines the target position yielding the best classification score. The proposed LCS tracker in Section IV is a discriminative one. Hence, we focus on the review of discriminative approaches only.

Since Avidan [40] formulated visual tracking as a classification problem, many discriminative trackers have been proposed based on various classification schemes, such as boosting variants [1], [2], [30], support vector machines (SVMs) [3], [15], [26], correlation filters [13], [16], [32], and neural networks [10], [33], [35], [37], [38]. Generally speaking, these classifiers are updated online with tracking results to adapt to target appearance changes. The online learning, however, may cause a drift problem. In other words, when tracking results are erroneous, the classifier is learned with false samples and the contaminated classifier, in turn, may provide further erroneous results. Therefore, false sample suppression is essential so as not to degrade the classification accuracy.

For the false sample suppression, Grabner *et al.* [1] proposed a semi-supervised approach, which trains a classifier with labeled samples in the first frame and unlabeled samples in subsequent frames to reduce adverse impacts of false labels. Babenko *et al.* [2] adopted the multiple instance learning, which uses a bag of multiple samples, instead of a single sample, to overcome the ambiguity in the foreground labeling. Hare *et al.* [3] employed the structured SVM [41], which accepts structural information between samples, instead of

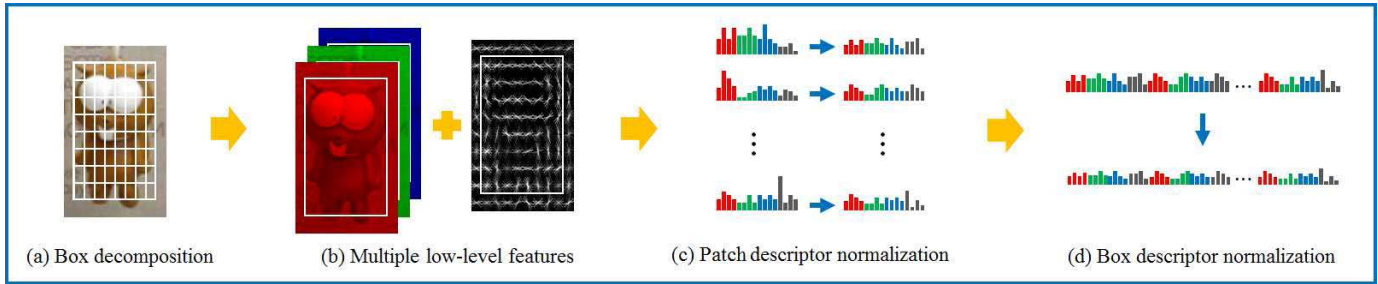


Fig. 1. Construction of an SOP descriptor: Given a bounding box, we decompose it into multiple patches in (a). Then, we construct patch descriptors by computing color and gradient histograms in (b). We normalize each histogram in the patch descriptors in (c). We then concatenate the patch descriptors and further normalize the box descriptor to yield the SOP descriptor in (d).

their binary labels, as input. In this paper, we also utilize the structured SVM to reduce the effects of false samples.

In discriminative tracking, it is also an important issue to adapt to target scale variations. Some trackers [12], [14], [37] simply generate candidates on various scales and then determine the best candidate to maximize the classification score. However, this simple scheme increases the search space (*i.e.*, the set of candidates) too much, which results in a higher computational complexity and also yields more erroneous results. To overcome this problem, Danelljan *et al.* [11] used two independent classifiers for the translation estimation and the scale estimation, respectively. Their tracker estimates the target center on a fixed scale and then adjusts the target scale around the estimated center. This two-step approach performs well in practice. Thus, we adopt a similar approach in this work, by employing the locator and the scaler. Nam and Han [10] used the bounding box regression technique [42] to refine a target scale.

An evaluation scheme can be employed to identify erroneous tracking results and correct them. Kalal *et al.* [39] developed a tracking system, composed of a tracker, a detector, and a learning system. The detector corrects the tracker if necessary, and the learning system estimates erroneous detection results to avoid such errors in the future. Zhang *et al.* [15] utilized trackers in previous frames to evaluate tracking results and restore a degraded tracker due to bad updates. Biresaw *et al.* [43] proposed the track-evaluate-correct framework, which determines an uncertain tracker based on an online performance measure and corrects inaccurate results with the assistance from neighboring trackers. Ma *et al.* [32] employed the correlation filter and the random fern classifier as a tracker and a detector, respectively. Their integrated system yields better results than the single correlation filter. These trackers [15], [32], [39], [43] are related to the proposed LCS tracker, in that they evaluate tracking results. However, unlike these trackers, the proposed tracker employs a delayed update scheme to suppress the drift problem.

In [44] and [45], multiple estimators are integrated to yield more accurate tracking results. Biresaw *et al.* [44] developed the tracker-level fusion algorithm, which conducts online tracking quality assessment to combine different trackers and improve the overall performance. Lee *et al.* [45] proposed the multihypothesis tracker, combining multiple component trackers using texture, color, and illumination-invariant

features, respectively. The proposed LCS tracker also uses multiple estimators. However, unlike [44], [45], the role of each estimator in the proposed tracker is clearly separated.

III. PROPOSED ALGORITHM: SOWP DESCRIPTOR

Let $\mathbf{x} = (\mathbf{c}, w, h)$ denote the bounding box (or state) of a target object, where \mathbf{c} , w , and h are the center coordinates, width, and height of the target, respectively. The contents in \mathbf{x} are encoded into a feature vector $\phi(\mathbf{x})$. As the study in [25] pointed out, the performance of a tracker is strongly dependent on this feature extraction or bounding box description. For visual tracking, it is essential to design an effective method for bounding box description.

We propose an accurate and reliable description method, called the SOWP descriptor. First, in Section III-A, we introduce the spatially ordered patch (SOP) descriptor, which exploits multiple low-level features and structural information in a bounding box. Then, in Section III-B, we develop an adaptive foreground weighting scheme, and apply the foreground weights to the SOP descriptor to yield the SOWP descriptor.

A. SOP Description

In the SOP description, a bounding box \mathbf{x} is regarded as an ordered set of non-overlapping patches, and each patch is described by low-level features. In other words, the SOP descriptor $\phi(\mathbf{x})$ for the bounding box \mathbf{x} is constructed by concatenating the feature vectors of all patches according to their spatial orders,

$$\phi(\mathbf{x}) = [\mathbf{f}_1^T, \dots, \mathbf{f}_N^T]^T \quad (1)$$

where \mathbf{f}_i is the feature vector of the i th patch and N is the number of patches in the bounding box.

Fig. 1 illustrates the process of the SOP description. In Fig. 1(a), we divide a bounding box into N patches, where $N = 8 \times 8$. In Fig. 1(b), we construct each patch descriptor \mathbf{f}_i by extracting multiple low-level features: three color histograms and a single gradient histogram. Then, in Fig. 1(c), we normalize the l_2 -norms of the four histograms, respectively, to equalize their contributions. Then, in Fig. 1(d), we concatenate the patch descriptors and further normalize the concatenated vector to set its l_2 -norm to 1. The resultant vector is the SOP descriptor.



Fig. 2. Examples, in which non-target objects belong to the same category as a target one does. Target objects are within yellow boxes.

Note that the concept of the SOP descriptor has been used in computer vision, since it conveys structural information in a bounding box effectively, by preserving the orders of patches within the box. For example, HOG, which represents a bounding box by concatenating the oriented gradient histograms of patches, is widely used in object detection [27], [42]. However, whereas the goal of object detection is to identify objects in a certain category, that of object tracking is to track a target object and distinguish it from non-targets. Object tracking is challenging, since the non-targets may belong to the same category as the target does. For instance, in Fig. 2, for successful tracking, the object descriptor should be able to distinguish the target sprinter (or dancer) from the other sprinters (or dancers), who have similar shapes or colors as the target. In such a case, the descriptor, composed of a single type of features, may be insufficient.

Therefore, we extract multiple low-level features from each patch to construct its feature vector \mathbf{f}_i in (1). As will be detailed in Section V-B, we evaluated various combinations of five low-level features: RGB color histogram, Lab color histogram, HSV color histogram, intensity histogram, and HOG [42]. We found that the combination of a 24-dimensional HSV color histogram (8-dimensional histogram for each color channel) and a 31-dimensional HOG [42] provides superb tracking performance. Consequently, we describe each patch with the 55-dimensional feature vector.

Also, in (1), we empirically set the number of patches within a bounding box to $N = 64$. Too many patches increase the complexity due to a high descriptor dimension. In contrast, a smaller number of patches cannot describe object structures faithfully. We test N from 36 to 100 and select $N = 64$ by considering both the tracking performance and the complexity, as will be described in detail in Section V-B.

B. SOWP Description

Ideally, the bounding box of a target object should contain foreground information only. However, as illustrated in Fig. 3, it often includes background information as well, which corrupts the descriptor and the classifier. Specifically, the background information may be used for training the classifier improperly, and then the wrongly trained classifier may detect a bounding box with more background pixels. This series of background expansion in a bounding box eventually results in a tracking failure in the video in Fig. 3. To overcome this problem, we assign foreground weights to patches in a



Fig. 3. Inclusion of background information within a bounding box due to a complex object shape. In this video, the background information causes a drift problem, causing the tracker to lose the target object eventually.

bounding box. The foreground weight of a patch represents the likelihood that the patch belongs to the foreground.

Fig. 4 summarizes the process of the SOWP description. First, we construct a graph by employing patches inside and around the bounding box as nodes. Second, we perform RWR simulation twice on the graph to compute the foreground and background stationary distributions, π_i^F and π_i^B , respectively. Third, we compare these two distributions to obtain the foreground weights. Finally, by modulating the magnitudes of the patch descriptors in the SOP descriptor using the foreground weights, we obtain the SOWP descriptor. Let us describe each step in detail subsequently.

We employ the RWR simulation to compute foreground weights. In RWR [8], [46], [47], a random walker traverses a graph $G = (V, E)$, where $V = \{v_1, \dots, v_M\}$ is the node set, M is the number of nodes, and E is the edge set. Suppose that edge $e_{ij} \in E$ connects nodes v_i and v_j . Then, with the transition probability a_{ij} , the walker at v_j traverses to v_i . The transition matrix \mathbf{A} records these transition probabilities, *i.e.*, the (i, j) th element of \mathbf{A} is a_{ij} . Note that $a_{ij} = 0$ if there is no edge from v_j to v_i . Unlike the ordinary random walk, RWR enforces the walker to return to specified nodes, according to a restart distribution $\mathbf{r} = [r_1, \dots, r_M]^T$, with a restart probability ϵ . Let $\mathbf{p}^{(k)} = [p_1^{(k)}, \dots, p_M^{(k)}]^T$ denote the probability distribution of the walker at the k th iteration. Then, we have the recursion

$$\mathbf{p}^{(k)} = (1 - \epsilon)\mathbf{A}\mathbf{p}^{(k-1)} + \epsilon\mathbf{r}. \quad (2)$$

It can be shown [47] that, as k approaches infinity, $\mathbf{p}^{(k)}$ converges to the stationary distribution $\boldsymbol{\pi} = \lim_{t \rightarrow \infty} \mathbf{p}^{(k)}$, regardless of an initial distribution $\mathbf{p}^{(0)}$. The stationary distribution $\boldsymbol{\pi}$ satisfies

$$\boldsymbol{\pi} = (1 - \epsilon)\mathbf{A}\boldsymbol{\pi} + \epsilon\mathbf{r}, \quad (3)$$

and it can be obtained by applying (2) recursively. The stationary distribution $\boldsymbol{\pi}$ represents the affinity of each node to the specified nodes in the restart distribution \mathbf{r} , since the random walker is compelled to return to the specified nodes at each iteration. This property can be exploited for data clustering. For example, RWR has been adopted in data mining [8], [46], interactive image segmentation [48], and saliency detection [49], [50].

In this work, we adopt RWR to achieve foreground clustering and background clustering in a bounding box. To this end, we first define the node set V , which consists of patches

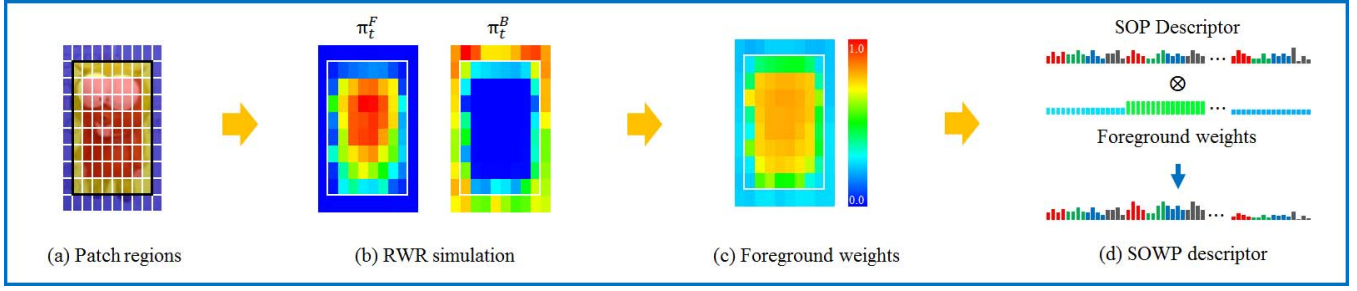


Fig. 4. Construction of an SOWP descriptor: We define the set of inner, boundary, and outer patches in (a), all of which become nodes in the graph. We compute the stationary distributions π_t^F and π_t^B , by carrying out the RWR simulation twice with different restart distributions, in (b). We obtain the foreground weight vector ρ_t , by comparing π_t^F with π_t^B , in (c). Finally, we apply the foreground weights to the SOP descriptor to yield the SOWP descriptor in (d).

within and around the bounding box. For instance, in Fig. 4(a), the thick black rectangle depicts the bounding box, which is composed of $N = 8 \times 8$ patches. These inside patches are classified into two categories: *boundary* patches and *inner* patches. A boundary patch shares the boundary with the bounding box, while an inner patch does not. In Fig. 4(a), there are 28 boundary patches and 36 inner patches, which are depicted in red and yellow, respectively. Additionally, we consider *outer* patches which are outside the bounding box but share the boundary. In Fig. 4(a), they are depicted in blue. Let Ω_t^{in} , Ω_t^{bnd} , and Ω_t^{out} denote the sets of inner, boundary, and outer patches of a bounding box at the t th frame, respectively. All these patches become nodes in V .

Next, we define the edge set E . If nodes (*i.e.* patches) v_i and v_j are 8-neighbors, they are connected by edge e_{ij} , which is assigned a weight, w_{ij} , according to the feature vectors \mathbf{f}_i and \mathbf{f}_j of the patches. Specifically,

$$w_{ij} = \exp(-\gamma \|\mathbf{f}_i - \mathbf{f}_j\|^2) \quad (4)$$

where γ is a scaling parameter. Then, we define the transition matrix \mathbf{A} by normalizing the edge weights as

$$a_{ij} = \frac{w_{ij}}{\sum_i w_{ij}}. \quad (5)$$

As mentioned previously, a_{ij} is the probability that the walker at v_j traverses to v_i . A high w_{ij} means that the two patches v_i and v_j are similar to each other, and a_{ij} is proportional to w_{ij} . Therefore, the random walker at a certain patch moves to a more similar patch with a higher probability.

We perform the RWR simulation on the graph $G = (V, E)$ twice using two different restart distributions \mathbf{r}_t^F and \mathbf{r}_t^B , which are associated with foreground and background regions, respectively. Then, as in (3), we obtain the foreground and background stationary distributions, π_t^F and π_t^B . Specifically,

$$\pi_t^F = (1 - \epsilon)\mathbf{A}\pi_t^F + \epsilon\mathbf{r}_t^F, \quad (6)$$

$$\pi_t^B = (1 - \epsilon)\mathbf{A}\pi_t^B + \epsilon\mathbf{r}_t^B. \quad (7)$$

These stationary distributions π_t^F and π_t^B represent probabilistic shapes of a target object and its background, by forming clusters around the restart distributions \mathbf{r}_t^F and \mathbf{r}_t^B , respectively.

Contrary to the patches near the center of the bounding box, the patches near the box boundary are likely to have

background features. In other words, many patches in Ω_t^{in} are foreground patches, while many ones in Ω_t^{out} belong to the background. Based on these assumptions, we determine the foreground and background restart distributions \mathbf{r}_t^F and \mathbf{r}_t^B by

$$\mathbf{r}_t^F(i) = \begin{cases} \kappa_t^F \times \pi_{t-1}^F(i) & \text{if } v_i \in \Omega_t^{\text{in}} \cup \Omega_t^{\text{bnd}} \\ 0 & \text{if } v_i \in \Omega_t^{\text{out}} \end{cases} \quad (8)$$

$$\mathbf{r}_t^B(i) = \begin{cases} 0 & \text{if } v_i \in \Omega_t^{\text{in}} \\ \kappa_t^B \times \pi_{t-1}^B(i) & \text{if } v_i \in \Omega_t^{\text{bnd}} \cup \Omega_t^{\text{out}} \end{cases} \quad (9)$$

where κ_t^F and κ_t^B are the normalizing parameters to make \mathbf{r}_t^F and \mathbf{r}_t^B probability distributions. Notice that we employ the stationary distributions at the previous frame to estimate the restart distributions at the current frame, since the shape of a target object changes smoothly between consecutive frames in a typical video. Fig. 4(b) illustrates the stationary distributions π_t^F and π_t^B , which represent the likelihoods that each patch belongs to the foreground and the background, respectively.

The i th patch is likely to contain foreground information, when it has a large $\pi_t^F(i)$ and a small $\pi_t^B(i)$. Therefore, we define the foreground weight $\rho_t(i)$ of the i th patch as

$$\rho_t(i) = \frac{1}{1 + \exp(-\alpha(\pi_t^F(i) - \pi_t^B(i)))} \quad (10)$$

where α controls the steepness of the logistic function. Fig. 4(c) shows the foreground weights, where red and blue colors depict large and small weights, respectively. We see that the patches, which are assigned relatively large weights, reveal the object shape.

Finally, as in Fig. 4(d), we use the foreground weights, when concatenating the patch descriptors, and obtain

$$\phi(\mathbf{x}_t) = [\rho_t(1)\mathbf{f}_1^T, \dots, \rho_t(N)\mathbf{f}_N^T]^T. \quad (11)$$

which is the SOWP descriptor for the bounding box \mathbf{x}_t . This is similar to the SOP descriptor in (1), but the patch descriptors are multiplied by the corresponding foreground weights.

IV. PROPOSED ALGORITHM: LCS TRACKER

We propose the LCS tracker, composed of locator, checker, and scaler. First, the locator finds the center position of a target in a current frame, based on its previous state. Second, the checker determines whether it is safe to adjust the target scale in the current frame. Third, the scaler estimates the size of

the target to adapt to scale variations. Let us describe each component in detail, and explain how they cooperate.

A. Locator

The locator is a classifier to identify the target center. For this purpose, we employ the structured SVM [41] that uses structural information between samples, instead of binary labels, for the training. Therefore, the structure SVM does not require a heuristic binary labeling process during the classifier update, and outperforms the binary SVM. Note that the Struct tracker [3] also adopts the structured SVM to alleviate adverse impacts of false labels and yields excellent performance in a recent tracking benchmark [9].

Let $\mathbf{x}_{t-1} = (\mathbf{c}_{t-1}, w_{t-1}, h_{t-1})$ be the target state at the previous frame $t - 1$, where \mathbf{c}_{t-1} , w_{t-1} , and h_{t-1} are the target center, width, and height, respectively. To estimate the target center \mathbf{c}_t at the current frame t , we first set a square search region, which is centered at \mathbf{c}_{t-1} and has a side length of $2\sqrt{w_{t-1}h_{t-1}}$. Then, we sample candidate states within the search region using the sliding window method. At this stage, every candidate has the same size $w_{t-1} \times h_{t-1}$. To describe the contents in each candidate state \mathbf{x} , we encode it into the SOWP descriptor $\phi(\mathbf{x})$ via (11). We then determine the current state \mathbf{x}_t to yield the highest score,

$$\mathbf{x}_t = (\mathbf{c}_t, w_{t-1}, h_{t-1}) = \arg \max_{\mathbf{x}} \mathbf{u}_{t-1}^T \phi(\mathbf{x}) \quad (12)$$

where \mathbf{u}_{t-1} is the weight vector of the classifier (*i.e.* locator) updated at the previous frame $t - 1$.

After the state estimation, we update the weight vector \mathbf{u}_t to adapt to the appearance change of the target. To this end, we sample bounding boxes on a polar grid around the current object location and extract their descriptors. The structured SVM constrains that the bounding box \mathbf{x}_t should yield a larger score than a nearby box \mathbf{x} by a margin, which decreases as the intersection over union overlap ratio, $\text{IoU}(\mathbf{x}_t, \mathbf{x})$, between the two boxes increases. Specifically, the objective function of the structured SVM is given by

$$\mathbf{u}^* = \arg \min_{\mathbf{u}} \lambda \|\mathbf{u}\|^2 + \sum_{\mathbf{x}} \max\{0, \Delta(\mathbf{x}_t, \mathbf{x}) - \mathbf{u}^T \delta(\mathbf{x}_t, \mathbf{x})\} \quad (13)$$

where $\Delta(\mathbf{x}_t, \mathbf{x}) = 1 - \text{IoU}(\mathbf{x}_t, \mathbf{x})$, $\delta(\mathbf{x}_t, \mathbf{x}) = \phi(\mathbf{x}_t) - \phi(\mathbf{x})$, and $\lambda = 0.0001$ is a regularization parameter. In (13), the regularization term $\|\mathbf{u}\|^2$ prevents the weight vector \mathbf{u}^* from having too large a magnitude. Also, the loss term

$$\begin{aligned} \Delta(\mathbf{x}_t, \mathbf{x}) - \mathbf{u}^T \delta(\mathbf{x}_t, \mathbf{x}) \\ = 1 - \text{IoU}(\mathbf{x}_t, \mathbf{x}) - (\mathbf{u}^T \phi(\mathbf{x}_t) - \mathbf{u}^T \phi(\mathbf{x})) \end{aligned} \quad (14)$$

means that, if there is a less overlap between \mathbf{x} and \mathbf{x}_t , the classification score of $\mathbf{u}^T \phi(\mathbf{x})$ should be much lower than the highest score $\mathbf{u}^T \phi(\mathbf{x}_t)$. In this way, the weight vector \mathbf{u}^* in (13) is optimized such that the score margin, $\mathbf{u}^{*T} \phi(\mathbf{x}_t) - \mathbf{u}^{*T} \phi(\mathbf{x})$, becomes bigger as the overlap $\text{IoU}(\mathbf{x}_t, \mathbf{x})$ gets smaller in general. For the optimization in (13), we employ the stochastic variance reduced gradient (SVRG) technique [51].

Then, to avoid the abrupt change between the consecutive weight vectors \mathbf{u}_{t-1} and \mathbf{u}_t , we update \mathbf{u}_t by mixing \mathbf{u}_{t-1} and \mathbf{u}^* . More specifically,

$$\mathbf{u}_t = (1 - \eta_u) \mathbf{u}_{t-1} + \eta_u \mathbf{u}^* \quad (15)$$

where $\eta_u = 0.1$ is a learning rate.

B. Checker

After the locator determines the target center \mathbf{c}_t , the scaler adjusts the width w_t and the height h_t . However, the scale estimation should be performed cautiously. When a target undergoes occlusion or abrupt illumination changes, its appearance is distorted, and the scale estimation becomes unreliable. Note that these incorrectly estimated scales contaminate the classifier during the training, and the contaminated classifier causes inaccurate tracking results in subsequent frames. Therefore, it is essential to identify *safe* frames for the scale estimation, in which a target appearance is not distorted. The checker is a binary classifier that is employed for this purpose.

Let \mathbf{x}_t be the estimated state of the locator in (12). Then, the checker computes a safety score c by

$$c = \mathbf{w}_{t-1}^T \phi(\mathbf{x}_t) \quad (16)$$

where \mathbf{w}_{t-1} is the weight vector of the checker at the previous frame $t - 1$. A positive score, $c > 0$, indicates that the current frame t is safe and the scaler can adjust the target scale. On the contrary, a negative score means that frame t is unsafe. For an unsafe frame, only the locator should be employed and the scaler should be bypassed. In such a case, the box size is unchanged from the previous frame, *i.e.* $w_t = w_{t-1}$ and $h_t = h_{t-1}$.

The checker encodes the appearance information of the target object in safe frames only into the weight vector \mathbf{w} . Since the target appearance may change gradually even in safe frames, we should update \mathbf{w} , as well as the weight vectors \mathbf{u} of the locator. However, the update of \mathbf{w} may also cause a drift problem. To avoid misclassifying unsafe frames into safe ones, we adopt a more conservative approach to the update of \mathbf{w} than to that of \mathbf{u} . Specifically, instead of updating \mathbf{w} every frame, we employ a delayed update scheme. Suppose that τ is the latest frame with a negative safety score, $c < 0$. Then, for the following safe frames, we do not update \mathbf{w} . Instead, we wait until another unsafe frame t is detected. We then use the information in the safe frames from $\tau + 1$ to $t - 1$ to update \mathbf{w}_t at frame t .

For updating \mathbf{w}_t , we collect training samples from frames $\tau + 1$ to $t - 1$ as follows. As positive samples, we gather the bounding boxes whose IoU overlap ratios with the target states are greater than 0.9. On the other hand, to collect various negative samples, we choose bounding boxes from the entire images. Specifically, we employ the object proposal technique in [52] and then select the proposals whose IoU overlap ratios with the target states are less than 0.3. Then, we optimize the objective function of a binary SVM,

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \sum_{\mathbf{x}} \max\{0, 1 - y_{\mathbf{x}} \mathbf{w}^T \phi(\mathbf{x})\} \quad (17)$$



Fig. 5. An example in which a target object experiences scale variations.

where $y_{\mathbf{x}} = 1$ if a sample \mathbf{x} is positive, and -1 otherwise. Thus, the loss term, $1 - y_{\mathbf{x}} \mathbf{w}^T \phi(\mathbf{x})$, penalizes a weight vector \mathbf{w} when the score $\mathbf{w}^T \phi(\mathbf{x})$ is smaller than 1 for a positive sample \mathbf{x} . On the contrary, it penalizes \mathbf{w} when the score is larger than -1 for a negative sample.

Then, we update \mathbf{w}_t via

$$\mathbf{w}_t = (1 - \eta_w) \mathbf{w}_t + \eta_w \mathbf{w}^* \quad (18)$$

where the learning rate η_w is fixed to 0.2. An exception is the case $t = \tau + 1$, in which there is no safe frame between frames τ and t . Then, we update \mathbf{w}_t using samples from the current frame t . However, since the current frame is declared as unsafe, we use a smaller learning rate $\eta_w = 0.1$ to reduce the impacts of unreliable information.

C. Scaler

Fig. 5 shows an example in which a target becomes smaller gradually. In such a case, the locator cannot provide accurate results, since it considers only candidates of the same size. In other words, the search space excludes the true state of a target regarding scale, thereby leading to a tracking error. The problem can be alleviated by considering candidates of various sizes. However, this approach may generate too many candidates, increase false positives, and reduce the tracking reliability. Hence, for scale estimation, we adopt the approach of modern trackers [11], [16], [18], [32], which decomposes the problem of target state estimation into the two subproblems of translation estimation and scale estimation. It first estimates the center of a target using a fixed scale and then determines the scale of the target at the estimated center location. Despite its simplicity, this approach enables robust scale estimation, by reducing the number of candidates efficiently.

Notice that the scaler becomes active only if the checker declares that the current frame t is safe. In such a case, given the target state $\mathbf{x}_t = (\mathbf{c}_t, w_{t-1}, h_{t-1})$ estimated by the locator, the scaler samples bounding boxes to refine the scale. The sampled boxes are $\mathbf{x} = (\mathbf{c}_t, s \cdot w_{t-1}, s \cdot h_{t-1})$ with scale factors $s \in \{0.90, 0.91, \dots, 1.10\}$. Then, the scaler updates \mathbf{x}_t to the optimal box, yielding the highest score,

$$\mathbf{x}_t = (\mathbf{c}_t, w_t, h_t) = \arg \max_{\mathbf{x}} \mathbf{v}_{t-1}^T \phi(\mathbf{x}) \quad (19)$$

where \mathbf{v}_{t-1} is the weight vector of the scaler.

To update the weight vector \mathbf{v}_t , we carry out a similar procedure to the locator update. Given the optimal estimate $\mathbf{x}_t = (\mathbf{c}_t, w_t, h_t)$ in (19), we extract bounding boxes $\mathbf{x} = (\mathbf{c}_t, s \cdot w_t, s \cdot h_t)$ and the corresponding descriptors $\phi(\mathbf{x})$ for $s \in \{0.50, 0.51, \dots, 1.50\}$. Then, we find the optimal \mathbf{v}^* by

$$\mathbf{v}^* = \arg \min_{\mathbf{v}} \lambda \|\mathbf{v}\|^2 + \sum_{\mathbf{x}} \max\{0, \Delta(\mathbf{x}_t, \mathbf{x}) - \mathbf{v}^T \delta(\mathbf{x}_t, \mathbf{x})\} \quad (20)$$

Algorithm 1 LCS Tracking

Input: $\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}, \mathbf{w}_{t-1}, \tau$

Output: $\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t, \mathbf{w}_t, \tau$

```

1:  $\mathbf{x}_t \leftarrow$  translation estimation by locator  $\triangleright$  (12)
2:  $c \leftarrow$  safety score computation by checker  $\triangleright$  (16)
3: if  $c > 0$  then
4:    $\mathbf{x}_t \leftarrow$  scale estimation by scaler  $\triangleright$  (19)
5:    $\mathbf{u}_t \leftarrow$  locator update  $\triangleright$  (15)
6:    $\mathbf{v}_t \leftarrow$  scaler update  $\triangleright$  (21)
7:    $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1}$ 
8: else
9:   if  $t = \tau + 1$  then
10:     $\mathbf{u}_t \leftarrow$  locator update  $\triangleright$  (15)
11:     $\mathbf{v}_t \leftarrow$  scaler update  $\triangleright$  (21)
12:     $\mathbf{w}_t \leftarrow$  checker update with  $\eta_w = 0.1$   $\triangleright$  (18)
13:   else
14:     $\mathbf{w}_t \leftarrow$  checker update with  $\eta_w = 0.2$   $\triangleright$  (18)
15:   end if
16:    $\tau \leftarrow t$ 
17: end if

```

and update \mathbf{v}_t by

$$\mathbf{v}_t = (1 - \eta_v) \mathbf{v}_t + \eta_v \mathbf{v}^*, \quad (21)$$

in the same way as (13) and (15), respectively. The scaler uses the same learning rate $\eta_v = 0.1$ as the locator does.

D. LCS Tracker

Algorithm 1 summarizes how the proposed LCS tracker employs and updates the locator, the checker, and the scaler to track a target object in frame t . Given a previous state $\mathbf{x}_{t-1} = (\mathbf{c}_{t-1}, w_{t-1}, h_{t-1})$, we first estimate the target center \mathbf{c}_t using the locator. We then compute a safety score c using the checker. When frame t is declared as safe, *i.e.* $c > 0$, we estimate the target scale (w_t, h_t) using the scaler. Then, we update the locator and the scaler to account for the target appearance variation. On the contrary, when frame t is unsafe, we maintain the target scale without the adjustment and update the checker with training samples, which are extracted from previous safe frames. However, if there is no previous safe frame, *i.e.* $t = \tau + 1$, we train the locator, the scaler, and the checker by employing samples in the current frame t .

V. EXPERIMENTAL RESULTS

We assess the proposed SOWP descriptor and LCS tracker comparatively to conventional trackers on a recent benchmark in [9]. Furthermore, we report the results on other benchmarks [20], [21] as well for more comprehensive evaluation.

A. Datasets

The object tracking benchmark (OTB) [9] consists of 100 video sequences and provides the ground truth bounding boxes. Each sequence is annotated with its challenging attributes, such as illumination variation, scale variation, occlusion, and background clutters. Precision and success rate plots

TABLE I

PR/SR SCORES OF THE SOP AND SOWP DESCRIPTORS USING VARIOUS LOW-LEVEL FEATURES. A BOLD-FACED NUMBER DENOTES THE HIGHEST SCORE IN EACH TEST. THE BEST PR/SR SCORES ARE ACHIEVED BY SOWP USING ‘HSV+HOG’ FEATURES

	SOP	SOWP
HOG	0.724 / 0.507	0.742 / 0.519
Intensity	0.684 / 0.491	0.735 / 0.512
RGB	0.692 / 0.499	0.748 / 0.528
Lab	0.719 / 0.501	0.728 / 0.518
HSV	0.734 / 0.518	0.745 / 0.519
Intensity + HOG	0.771 / 0.544	0.799 / 0.558
RGB + HOG	0.808 / 0.573	0.807 / 0.565
Lab + HOG	0.791 / 0.553	0.809 / 0.554
HSV + HOG	0.832 / 0.579	0.860 / 0.585

are used to evaluate a tracker. The precision is the proportion of frames, in which the distance between an estimated object location and the ground truth is smaller than a threshold. The success rate is the proportion of frames, in which the overlap ratio between an estimated bounding box and the ground truth is larger than a threshold. The precision at the distance threshold of 20 pixels is employed as the representative precision (PR) score, and the average success rate, which is the area under the success rate curve over all overlap thresholds, is used as the representative success rate (SR) score. Since a small perturbation of an initial bounding box can lead to quite a different tracking result, the one pass evaluation (OPE) may be insufficient. Thus, the spatial robustness evaluation (SRE) and the temporal robustness evaluation (TRE) are also carried out. SRE varies an initial box with shifting and scaling, and TRE tests a tracker by starting it at 20 different initial frames.

The Temple Color 128 (TC) benchmark [20] contains 128 video sequences and the results of 16 conventional trackers, which are combined with 10 color models, such as RGB, Lab, and HSV histograms. TC benchmark also employs precision and success rate plots to evaluate a tracker.

The VOT 2015 challenge [21] includes 60 video sequences and provides the results of 62 conventional trackers. It adopts two evaluation metrics. First, the accuracy is the overlap ratio between an estimated bounding box and the ground truth. Second, the robustness counts tracking failures in a video sequence, *i.e.*, the number of frames in which the overlap ratios are zero.

B. Descriptor Analysis

The SOWP descriptor has three controllable parameters, γ in (4), ϵ in (6) and (7), and α in (10). In this work, they are set to $\gamma = 10.0$, $\epsilon = 0.75$, and $\alpha = 35.0$ to yield the best tracking performance. Note that they are fixed in all experiments.

Let us use the locator in Section IV-A as a simple base tracker to analyze the effectiveness of the proposed SOWP descriptor. Hence, the base tracker estimates only the translation of a target object, without considering the scale variation. Table I analyzes the impacts of using multiple low-level features. In this test, OPE is performed on the OTB benchmark, and the average PR/SR scores are reported.

TABLE II

THE PR/SR SCORES AND THE PROCESSING SPEEDS OF THE BASE TRACKER ACCORDING TO THE NUMBER OF PATCHES IN A BOUNDING BOX. A PROCESSING SPEED IS MEASURED IN FRAMES PER SECOND (FPS)

# of patches	PR / SR	FPS
6×6	0.811 / 0.555	13.63
7×7	0.830 / 0.567	11.64
8×8	0.860 / 0.585	9.01
9×9	0.841 / 0.571	7.80
10×10	0.841 / 0.576	7.12

Each color histogram is 24-dimensional, the intensity histogram is 8-dimensional, and HOG [42] is 31-dimensional. We can make the following observations: First, the performance of the SOP descriptor is significantly improved by combining a color histogram and HOG, instead of using one of the features only. This is because color and gradient histograms convey different types of information and are complementary to each other. Second, the patch weighting further improves the tracking performance in most cases. Specifically, the SOWP descriptor outperforms the SOP descriptor for every combination of features, except for ‘RGB+HOG,’ by employing the adaptive patch weighting.

In Table I, the SOWP descriptor using ‘HSV+HOG’ features yields the best performance. It is hence employed in the remaining experiments unless otherwise specified. The base tracker (PR/SR = 0.860/0.585) performs better than the Struck tracker [3] (PR/SR = 0.638/0.461), which ranked 1st in the benchmark [9]. In other words, the base tracker improves the PR and SR scores by 34.8% and 26.9%, respectively, as compared with Struck. Both Struck and the base tracker consider translational motion only and use the structured SVM as their classifiers. However, whereas Struck describes a bound box with Haar-like features, the base tracker uses the SOWP descriptor. This test confirms the efficacy of the SOWP descriptor.

Table II lists the PR/SR scores of the base tracker according to the number N of patches in a bounding box. It also presents the processing speeds in frames per second (FPS). We implement the tracker in C++ without parallelization and perform experiments using a 2.40GHz CPU. Note that the best performance is obtained when a bounding box is divided into 8×8 patches. Fewer patches provide a lower performance since they cannot represent object structures faithfully. On the other hand, if a bounding box is divided into too many patches, individual patch descriptors, composed of histograms, become unreliable, and the tracking performance is also degraded. Also, too many patches reduce the processing speed because of a high dimension of the object descriptor. Therefore, we fix the number of patches to $N = 8 \times 8$.

Fig. 6 shows the evolution of foreground weights in (10), which reflect the shapes of target objects over time faithfully. Without the adaptive weighting, the tracking fails on the ‘Singer2’ sequence, as shown in Fig. 3. In contrast, in Fig. 6(c), the base tracker traces the singer successfully using foreground weights. It is worth pointing out that the

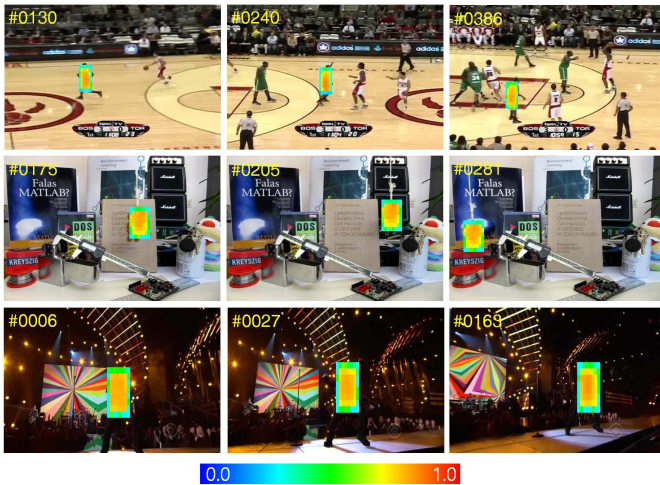


Fig. 6. The evolution of foreground weights through video sequences: From top to bottom, (a) ‘Basketball,’ (b) ‘Lemming,’ and (c) ‘Singer2.’

TABLE III
THE PR/SR SCORES AND THE PROCESSING SPEEDS OF
COMBINATIONS 1, 2, AND 3 ON THE OTB BENCHMARK.
BOTH SOP AND SOWP DESCRIPTORS
USE ‘HSV+HOG’ FEATURES

Tracker	Descriptor	PR / SR	FPS
Combination 1	SOP	0.832 / 0.579	9.18
	SOWP	0.860 / 0.585	9.01
Combination 2	SOP	0.805 / 0.605	8.38
	SOWP	0.814 / 0.607	8.32
Combination 3	SOP	0.840 / 0.627	0.87
	SOWP	0.858 / 0.639	0.84

foreground weights implicitly reshape rectangular bounding boxes to reflect the deformation of a target at different frames. Thus, those weights facilitate more reliable object description, especially when a target is occluded or changes its shape.

C. Tracking Component Analysis

Table III summarizes the PR/SR scores of the proposed tracker on the OTB benchmark, when we use different combinations of the locator, the checker, and the scaler. We use only the locator for ‘Combination 1,’ the locator and the scaler for ‘Combination 2,’ and all three components for ‘Combination 3.’ We see that the scaler improves the SR scores by enabling the tracker to adapt to target scale variations. However, the scale estimation without the checker lowers the PR score significantly. This is because, when target appearance is distorted due to challenging factors, including occlusion, deformation, and motion blur, the scaler fails to estimate the target size correctly and causes a tracking error. The checker is essential for robust scale estimation.

Also, Table III includes the performances when the SOWP descriptor is replaced by the SOP descriptor. We see that, for all three combinations, SOWP provides better PR/SR scores than SOP via the adaptive patch weighting. The computational complexity of the patch weighting is relatively low.

Table III compares the processing speeds as well. The computational load due the scaler is light, and the speed of Combination 2 is comparable to that of Combination 1.

On the other hand, the checker demands a heavy computational load and reduces the speed about tenfold, since it uses a lot of negative training samples obtained by the object proposal scheme [52]. A smaller number of samples can be used to reduce the complexity, but in such a case, the robustness of the checker is degraded and the overall performance is lowered. The complexity reduction of the checker, without lowering the performance, is a future research issue.

Fig. 7 shows the SR scores of the three combinations according to the various challenging factors [9]. Although the scaler improves the overall SR score, it decreases the SR scores especially in case of occlusion, deformation, and motion blur, which interfere with accurate scale estimation. However, when the scaler is integrated with the checker, the proposed LCS tracker achieves robust scale estimation. Therefore, the SR scores of Combination 3 are significantly higher than those of Combination 1 for all the challenging factors.

Fig. 8 compares tracking results of the three combinations. In the ‘Car1’ sequence, a car gets smaller gradually, and the scale estimation is necessary for its accurate tracking. Hence, Combinations 2 and 3 outperform Combination 1. However, Combination 2 misses a target in ‘Jogging2,’ where the target suffers from occlusion. It estimates the scale incorrectly at frame 52 due to the occlusion, and it cannot trace the target at frame 66 successfully. In contrast, Combination 3 does not adjust the target scale during the occlusion and restores the target after the occlusion. ‘Skating2.2’ is a challenging sequence with scale variation, occlusion, and deformation. Combination 1 fails to provide accurate results. Combination 2 yields erroneous scale estimation since the target is also occluded and deformed during the scale variation. In contrast, Combination 3 provides reliable tracking results.

Finally, we carry out a test to see if the checker can be replaced by the locator score. In this alternative approach, after the locator identifies a target center via (12), we compare the locator score $\mathbf{u}_{t-1}^T \phi(\mathbf{x}_t)$ with a threshold, instead of employing the checker to compute the safety score in (16). Then, we employ the scaler only if the locator score is larger than the threshold, which is set to 0.15 to yield the best performance. However, this replacement of the checker lowers the PR/SR scores to 0.826/0.611, which are significantly worse than the original scores 0.858/0.639 of the LCS tracker. Note that a structured SVM is trained to just predict a larger score for a target sample than the other samples with structured margins. Thus, locator scores are relative in the sense that even the best box can yield a very low score. The scores are useful to locate the best box among candidate boxes, but they are not suitable to determine whether the contents in the best box are distorted or not. Therefore, we train the checker, which is a binary SVM, to assign positive scores to target samples and negative scores to background samples. In this way, the safety score of the checker is non-relative, and it is more appropriate for finding irregularities in the bounding box than the locator score is.

D. Comparative Results on OTB Benchmark

We compare the proposed LCS tracker with conventional trackers: 31 trackers, whose results were reported in the

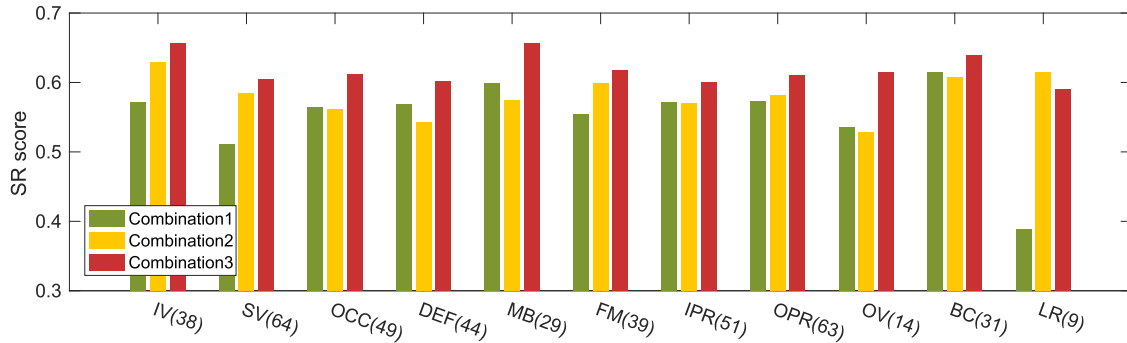


Fig. 7. The SR scores of the proposed LCS tracker on the OTB benchmark according to the 11 challenging factors [9]: IV (illumination variation), SV (scale variation), OCC (occlusion), DEF (deformation), MB (motion blur), FM (fast motion), IPR (in-plane-rotation), OPR (out-of-plane rotation), OV (out-of-view), BC (background clutters), and LR (low resolution). In this test, OPE is performed. Combination 1 uses only the locator, Combination 2 the locator and the scaler, and Combination 3 the locator, the scaler, and the checker.

TABLE IV
COMPARISON OF THE PR/SR SCORES OF THE PROPOSED LCS TRACKER AND RECENT STATE-OF-THE-ART TRACKERS IN THE OPE METHOD, ACCORDING TO THE 11 CHALLENGING FACTORS

	Struck [3]	KCF [13]	DSST [11]	MEEM [15]	MUSTer [16]	HCF [17]	SRDCF [18]	HDT [19]	MDNet [10]	Proposed
IV(38)	0.549 / 0.420	0.708 / 0.474	0.723 / 0.489	0.740 / 0.517	0.782 / 0.600	0.817 / 0.540	0.792 / 0.613	0.820 / 0.535	0.915 / 0.689	0.858 / 0.656
SV(64)	0.597 / 0.402	0.633 / 0.394	0.662 / 0.409	0.736 / 0.470	0.710 / 0.512	0.799 / 0.485	0.745 / 0.561	0.808 / 0.486	0.892 / 0.658	0.828 / 0.605
OCC(49)	0.532 / 0.391	0.622 / 0.438	0.615 / 0.426	0.741 / 0.504	0.734 / 0.554	0.767 / 0.525	0.735 / 0.559	0.774 / 0.528	0.857 / 0.646	0.817 / 0.612
DEF(44)	0.527 / 0.383	0.617 / 0.436	0.568 / 0.412	0.754 / 0.489	0.689 / 0.524	0.791 / 0.530	0.734 / 0.544	0.821 / 0.543	0.899 / 0.649	0.832 / 0.602
MB(29)	0.587 / 0.463	0.600 / 0.459	0.611 / 0.467	0.731 / 0.556	0.678 / 0.544	0.804 / 0.585	0.767 / 0.594	0.789 / 0.574	0.866 / 0.679	0.832 / 0.657
FM(39)	0.620 / 0.467	0.620 / 0.460	0.584 / 0.442	0.752 / 0.542	0.683 / 0.533	0.815 / 0.570	0.769 / 0.597	0.817 / 0.568	0.885 / 0.675	0.805 / 0.618
IPR(51)	0.633 / 0.451	0.693 / 0.465	0.724 / 0.485	0.794 / 0.529	0.773 / 0.551	0.854 / 0.559	0.745 / 0.544	0.844 / 0.555	0.910 / 0.655	0.843 / 0.601
OPR(63)	0.593 / 0.424	0.670 / 0.450	0.670 / 0.448	0.794 / 0.525	0.744 / 0.537	0.807 / 0.534	0.742 / 0.550	0.805 / 0.533	0.900 / 0.661	0.845 / 0.611
OV(14)	0.487 / 0.374	0.498 / 0.393	0.487 / 0.374	0.685 / 0.488	0.591 / 0.469	0.677 / 0.474	0.597 / 0.460	0.663 / 0.472	0.825 / 0.627	0.806 / 0.615
BC(31)	0.557 / 0.429	0.707 / 0.489	0.699 / 0.470	0.744 / 0.515	0.779 / 0.572	0.842 / 0.580	0.769 / 0.573	0.843 / 0.573	0.923 / 0.668	0.843 / 0.639
LR(9)	0.746 / 0.364	0.657 / 0.318	0.709 / 0.323	0.684 / 0.345	0.774 / 0.487	0.849 / 0.404	0.762 / 0.536	0.872 / 0.421	0.935 / 0.652	0.863 / 0.590
AVG(100)	0.638 / 0.461	0.692 / 0.475	0.695 / 0.475	0.781 / 0.530	0.774 / 0.577	0.837 / 0.562	0.789 / 0.598	0.848 / 0.564	0.909 / 0.678	0.858 / 0.639

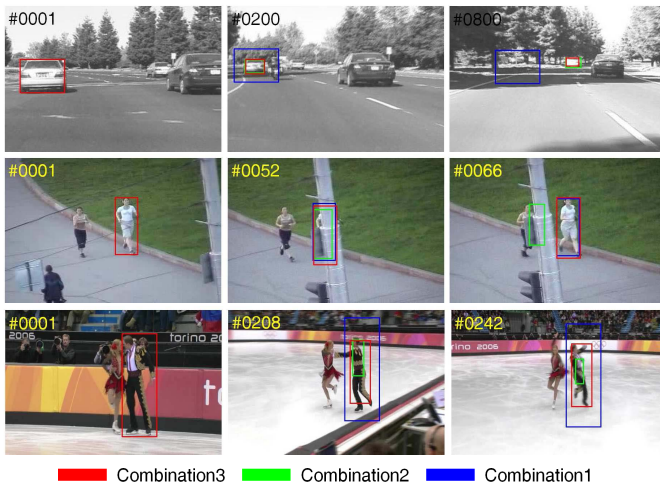


Fig. 8. Tracking results of Combinations 1, 2, and 3 on (a) Car1, (b) Jogging2, and (c) Skating2.2.

benchmark [9], and recent state-of-the-art trackers KCF [13], DSST [11], TGPR [14], MEEM [15], MUSTer [16], HCF [17], SRDCF [18], HDT [19], MDNet [10]. The source codes of these recent trackers are publicly available.

Fig. 9 plots the precision and success rate curves of the top 10 trackers in the OPE, SRE, and TRE methods. We see that, in OPE and SRE, the proposed tracker provides the second best PR and SR scores after MDNet [10]. In contrast,

in TRE, the proposed tracker outperforms MDNet, yielding the best performance among all tested trackers. These results can be interpreted as follows. For challenging sequences, by employing more discriminative features based on a convolutional neural network, MDNet causes less tracking failures than the proposed tracker. On the other hand, the proposed tracker tends to yield a higher overlap ratio between a tracking result and the ground truth in the normal tracking operation, by suppressing background information in the bounding box based on the RWR simulation. Note that TRE tests a tracker, by starting it at 20 initial frames, and then averages the performances over all 20 cases. Thus, it less penalizes tracking failures than OPE and SRE do.

Table IV reports the PR/SR scores of the trackers in the OPE method according to the 11 challenging factors. We observe that, in most cases, the proposed LCS tracker yields the second best PR/SR scores after MDNet. Note that MDNet, HDT, and HCF require heavy computational loads since they employ deep convolutional neural networks. On the contrary, the proposed tracker yields competitive results, by employing relatively simple features of color and gradient histograms. Table IV also confirms the effectiveness of the proposed SOWP descriptor. By comparing Table IV and Fig. 7, we can observe that even Combination 1 outperforms most state-of-the-art trackers despite its simplicity.

Fig. 10 shows tracking results of the top 5 trackers in Table IV, which are MDNet, HDT, HCF, SRDCF and the proposed LCS tracker. ‘Shaking’ and ‘Ironman’ contain severe

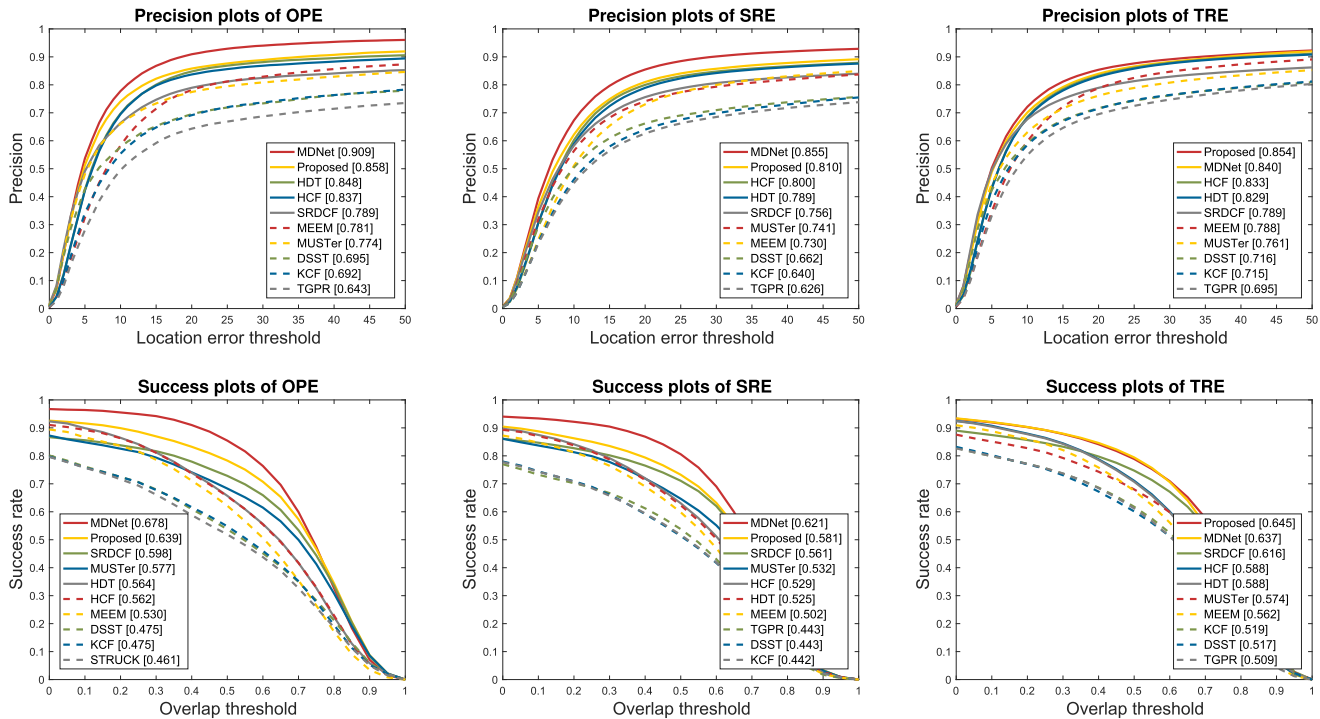


Fig. 9. Comparison of the precision and success rate plots on the OTB benchmark. The representative score for a precision plot measures the score at the threshold of 20 pixels, and the representative score for a success rate plot is the area under the curve over all overlap thresholds.

TABLE V

COMPARISON OF THE PR/SR SCORES ON THE TC BENCHMARK ACCORDING TO THE 11 CHALLENGING FACTORS. IN THIS TEST, THE OPE METHOD IS USED. NUMBERS IN PARENTHESES IN THE FIRST COLUMN REFER TO THE NUMBERS OF THE SEQUENCES WITH THE CORRESPONDING ATTRIBUTES. THE BEST RESULTS ARE BOLD FACED, AND THE SECOND BEST ONES ARE UNDERLINED

	LIAPG [53]	OAB [54]	MIL [2]	VTD [55]	Frag [56]	ASLA [57]	KCF [13]	Struck [3]	MEEM [15]	Proposed
IV(37)	0.472 / 0.366	0.507 / 0.397	0.445 / 0.351	0.506 / 0.404	0.406 / 0.342	0.534 / 0.421	0.558 / 0.409	0.607 / 0.448	<u>0.643</u> / <u>0.479</u>	0.750 / 0.580
SV(65)	0.485 / 0.375	0.522 / 0.367	0.514 / 0.370	0.493 / 0.371	0.447 / 0.348	0.585 / 0.416	0.544 / 0.365	0.613 / 0.414	<u>0.631</u> / <u>0.432</u>	0.726 / 0.532
OCC(62)	0.412 / 0.323	0.454 / 0.341	0.435 / 0.343	0.431 / 0.340	0.454 / 0.362	0.470 / 0.352	0.517 / 0.385	0.531 / 0.395	<u>0.647</u> / <u>0.468</u>	0.673 / 0.510
DEF(36)	0.444 / 0.293	0.499 / 0.359	0.578 / 0.402	0.513 / 0.365	0.523 / 0.378	0.560 / 0.401	0.654 / 0.453	0.661 / 0.462	<u>0.745</u> / <u>0.493</u>	0.867 / 0.610
MB(37)	0.368 / 0.286	0.441 / 0.334	0.487 / 0.356	0.442 / 0.329	0.517 / 0.397	0.393 / 0.291	0.513 / 0.374	0.531 / 0.403	<u>0.568</u> / <u>0.434</u>	0.629 / 0.464
FM(57)	0.343 / 0.300	0.434 / 0.373	0.462 / 0.382	0.416 / 0.357	0.471 / 0.392	0.394 / 0.327	0.503 / 0.408	0.546 / 0.435	<u>0.599</u> / <u>0.472</u>	0.650 / 0.505
IPR(57)	0.406 / 0.337	0.475 / 0.373	0.469 / 0.373	0.463 / 0.372	0.432 / 0.353	0.495 / 0.372	0.503 / 0.376	0.568 / 0.432	<u>0.638</u> / <u>0.464</u>	0.677 / 0.517
OPR(73)	0.424 / 0.346	0.503 / 0.390	0.481 / 0.389	0.475 / 0.375	0.458 / 0.375	0.501 / 0.391	0.554 / 0.415	0.594 / 0.443	<u>0.654</u> / <u>0.478</u>	0.735 / 0.546
OV(14)	0.299 / 0.262	0.349 / 0.281	0.457 / 0.355	0.333 / 0.289	0.558 / 0.428	0.300 / 0.230	0.419 / 0.334	<u>0.556</u> / 0.410	0.544 / <u>0.439</u>	0.559 / 0.447
BC(45)	0.499 / 0.354	0.581 / 0.396	0.529 / 0.363	0.534 / 0.375	0.539 / 0.389	0.602 / 0.420	0.622 / 0.416	0.634 / 0.444	<u>0.754</u> / <u>0.515</u>	0.775 / 0.557
LR(18)	0.724 / 0.458	0.532 / 0.265	0.674 / 0.348	0.561 / 0.317	0.632 / 0.392	0.691 / 0.397	0.459 / 0.262	0.729 / 0.461	<u>0.740</u> / <u>0.468</u>	0.753 / 0.505
AVG(129)	0.493 / 0.374	0.527 / 0.386	0.541 / 0.390	0.528 / 0.392	0.537 / 0.405	0.562 / 0.406	0.584 / 0.414	0.641 / 0.460	<u>0.705</u> / <u>0.496</u>	0.752 / 0.555

illumination variations and background clutters. Moreover, ‘Ironman’ also suffers from fast motion, motion blur, and rotation. In such cases, it is essential to construct a robust descriptor that can distinguish a target from the background. The proposed tracker deals with these challenging factors successfully using the SOWP descriptor. MDNet, HDT, and HCF also provide comparable results by employing deep convolutional networks to construct feature maps, which represent target contents effectively. In contrast, SRDCF fails to trace the targets. ‘Freeman1’ undergoes scale variations. HDT and HCF estimate target centers successfully but do not consider the varying scales. The other trackers perform the scale estimation effectively. ‘Human9’, ‘ClifBar’, and ‘Box’ experience illumination variations, motion blur, and occlusion, respectively. The targets also undergo scale variations. SRDCF fails to perform reliable scale estimation in these sequences. Although MDNet yields competitive results in ‘Human9’ and ‘ClifBar’, it misses

the target in ‘Box’ due to occlusion. In contrast, the proposed tracker tracks all three targets successfully. Finally, ‘Girl2’ and ‘Skating1’ are also challenging due to illumination variations, occlusion, deformation, motion blur, and background clutters. However, the proposed LCS tracker traces the targets in these challenging sequences accurately as well.

E. Comparative Results on TC Benchmark

We also test the proposed LCS tracker on the TC benchmark [20]. For comparison, we use the results of the 16 conventional trackers, which were reported in [20]. Fig. 11 shows the precision and success rate plots of the top ten trackers in the OPE method. The proposed tracker outperforms the second best tracker MEEM [15] considerably, achieving score margins of 0.047 in PR and 0.059 in SR. Table V reports the PR/SR scores according to the 11 challenging factor. It is remarkable

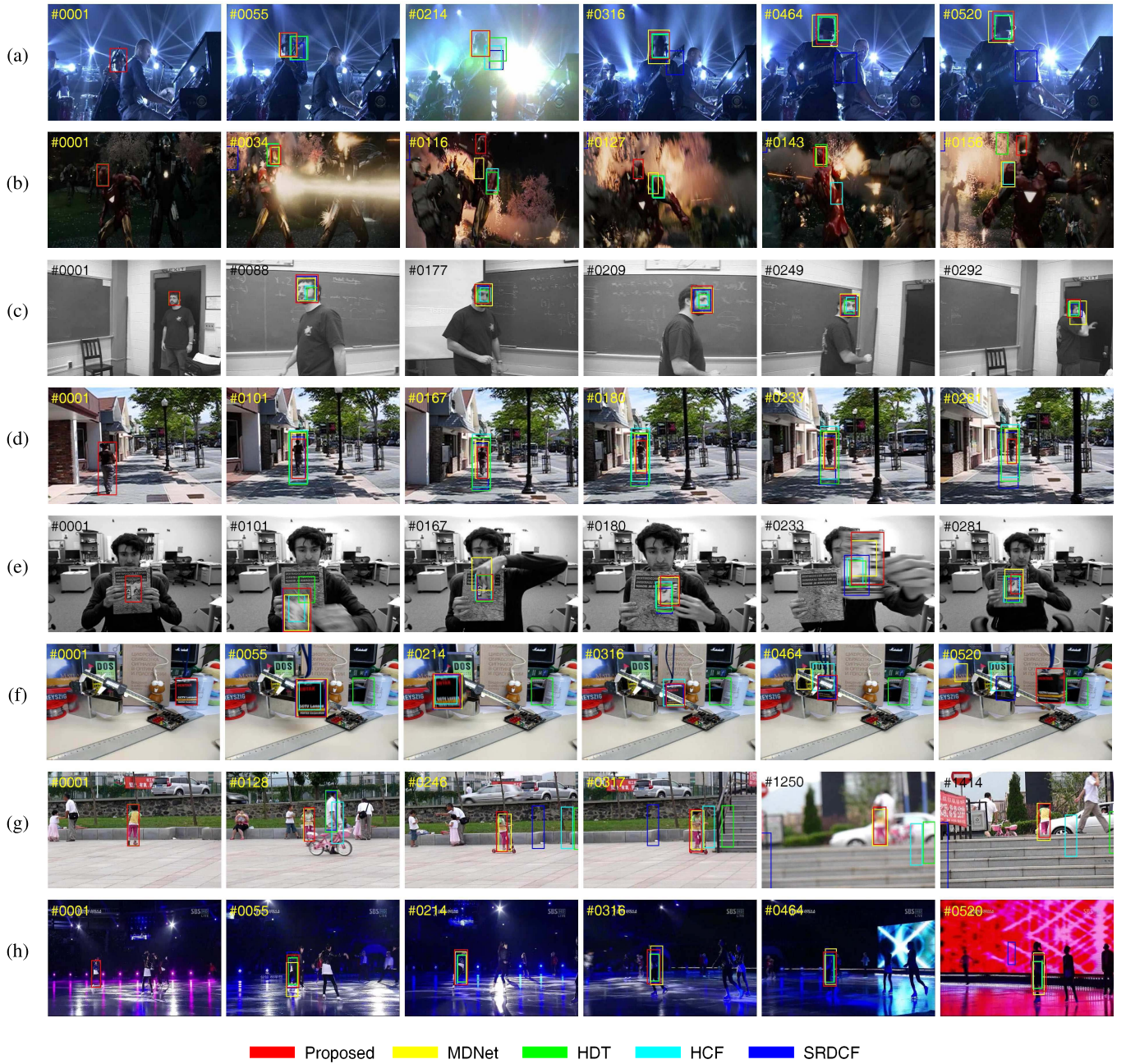


Fig. 10. Tracking results of the proposed LCS tracker and the conventional MDNet [10], HDT [19], HCF [17] and SRDCF [18] on the (a) Shaking, (b) Ironman, (c), Freeman1, (d), Human9, (e) ClifBar, (f) Box, (g) Girl2, and (h) Skating1 sequences.

TABLE VI
COMPARISON OF THE PROPOSED LCS TRACKER AND THE TOP 9 PUBLISHED TRACKERS IN THE VOT2015 CHALLENGE.
THE BEST RESULTS ARE BOLDFACED, AND THE SECOND BEST ONES ARE UNDERLINED

	S3Tracker [58]	RAJSSC [59]	Struck [3]	NSAMF [12]	SPST [60]	SRDCF [18]	EBT [61]	DeepSRDCF [62]	MDNet [10]	Proposed
Accuracy	0.515	0.566	0.471	0.530	0.547	0.559	0.473	0.564	0.603	<u>0.570</u>
Robustness	1.768	1.630	1.610	1.292	1.480	1.242	<u>1.021</u>	1.046	0.694	1.601

that the proposed tracker yields the best PR/SR scores for all the challenging factors.

F. Comparative Results on VOT2015 Benchmark

Finally, we assess the performance of the proposed LCS tracker on the VOT2015 challenge [21]. The proposed

tracker is compared with 62 recent trackers, whose results were reported in [21]. Table VI lists the accuracy and robustness scores of the proposed tracker and the top 9 published trackers in [21]. The proposed tracker provides the second best accuracy score and the seventh best robustness score.

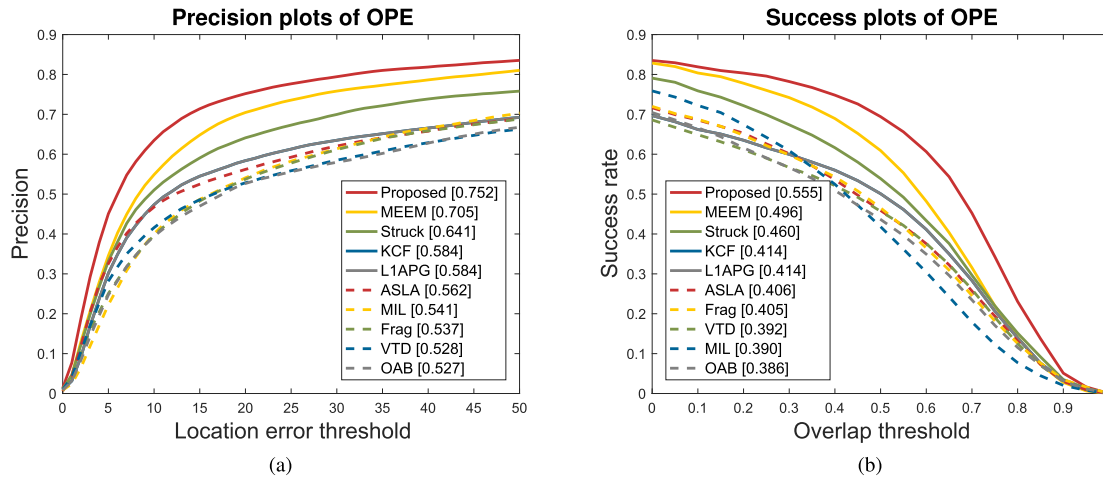


Fig. 11. Comparison of the precision and success rate plots on the TC benchmark. The representative score for a precision plot measures the score at the threshold of 20 pixels, and the representative score for a success rate plot is the area under the curve over all overlap thresholds.

VI. CONCLUSIONS

In this paper, we proposed the SOWP descriptor for robust visual tracking. We decomposed the bounding box of a target object into multiple patches and encoded them into color and gradient histograms. Then, we described the object appearance by concatenating the patch descriptors. In the concatenation, we assigned different weights to those patches according to their relevance to the target appearance, by performing the RWR simulation of the foreground and background walkers. We thus suppressed the background information in the bounding box efficiently. The SOWP descriptor was evaluated extensively, and it was shown that even a simple tracker using the SOWP descriptor provides competitive results on the benchmark [9].

Moreover, we also developed the LCS tracker, composed of the locator, the checker, and the scaler. The locator finds the center location of a target, and the scaler estimates the target size to account for scale variations. The checker identifies safe frames, in which target scales can be adjusted reliably. We evaluated the LCS tracker using the SOWP descriptor on the benchmarks [9], [20], [21]. It was demonstrated that the proposed algorithm yields promising performance. A limitation of the LCS tracker is the relatively high computational complexity of the checker. Although the checker is essential for robust scale estimation, it is a computational bottleneck in the LCS tracker. Its complexity reduction is one of the future research issues.

REFERENCES

- [1] H. Grabner, M. Grabner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. ECCV*, Oct. 2008, pp. 234–247.
- [2] 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.
- [3] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE ICCV*, Nov. 2011, pp. 263–270.
- [4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.
- [5] M. Danelljan, F. S. Khan, M. Felsberg, and J. van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE CVPR*, Jun. 2014, pp. 1090–1097.
- [6] D. Chen, Z. Yuan, Y. Wu, G. Zhang, and N. Zheng, "Constructing adaptive complex cells for robust visual tracking," in *Proc. IEEE ICCV*, Dec. 2013, pp. 1113–1120.
- [7] S. He, Q. Yang, R. Lau, J. Wang, and M.-H. Yang, "Visual tracking via locality sensitive histograms," in *Proc. IEEE CVPR*, Jun. 2013, pp. 2427–2434.
- [8] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proc. ACM SIGKDD*, Aug. 2004, pp. 653–658.
- [9] 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.
- [10] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE CVPR*, Jun. 2016, pp. 4293–4302.
- [11] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. BMVC*, Sep. 2014, pp. 1–11.
- [12] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proc. ECCV Workshops*, Sep. 2014, pp. 254–265.
- [13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.
- [14] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Proc. ECCV*, Sep. 2014, pp. 188–203.
- [15] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust tracking via multiple experts using entropy minimization," in *Proc. ECCV*, Sep. 2014, pp. 188–203.
- [16] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "Multi-store tracker (MUSTER): A cognitive psychology inspired approach to object tracking," in *Proc. IEEE CVPR*, Jun. 2015, pp. 749–758.
- [17] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE ICCV*, Dec. 2015, pp. 3074–3082.
- [18] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE ICCV*, Dec. 2015, pp. 4310–4318.
- [19] Y. Qi *et al.*, "Hedged deep tracking," in *Proc. IEEE CVPR*, Jun. 2016, pp. 4303–4311.
- [20] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: Algorithms and benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5630–5644, Dec. 2015.
- [21] M. Kristan *et al.*, "The visual object tracking VOT2015 challenge results," in *Proc. ICCV Workshops*, Dec. 2015, pp. 564–586.
- [22] H.-U. Kim, D.-Y. Lee, J.-Y. Sim, and C.-S. Kim, "SOWP: Spatially ordered and weighted patch descriptor for visual tracking," in *Proc. IEEE ICCV*, Dec. 2015, pp. 3011–3019.
- [23] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

- [24] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1261–1274, Mar. 2016.
- [25] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *Proc. IEEE ICCV*, Dec. 2015, pp. 3101–3109.
- [26] L. Zhang and L. van der Maaten, "Preserving structure in model-free tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 756–769, Apr. 2014.
- [27] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, Jun. 2005, pp. 886–893.
- [28] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, Aug. 2008.
- [29] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
- [30] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.
- [31] X. Li, C. Shen, A. Dick, and A. van den Hengel, "Learning compact binary codes for visual tracking," in *Proc. IEEE CVPR*, Jun. 2013, pp. 2419–2426.
- [32] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE CVPR*, Jun. 2015, pp. 5388–5396.
- [33] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. NIPS*, Dec. 2013, pp. 809–817.
- [34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
- [35] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video tracking using learned hierarchical features," *IEEE Trans. Image Process.*, vol. 24, no. 4, pp. 1424–1435, Apr. 2015.
- [36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, May 2015, pp. 1–14.
- [37] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE CVPR*, Jun. 2016, pp. 1420–1429.
- [38] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional Siamese networks for object tracking," in *Proc. ECCV Workshops*, Oct. 2016, pp. 1420–1429.
- [39] 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.
- [40] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.
- [41] I. Tschantzaris, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Sep. 2005.
- [42] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
- [43] T. A. Biresaw, A. Cavallaro, and C. S. Regazzoni, "Correlation-based self-correcting tracking," *Neurocomputing*, vol. 152, no. 1, pp. 345–358, Mar. 2015.
- [44] T. A. Biresaw, A. Cavallaro, and C. S. Regazzoni, "Tracker-level fusion for robust Bayesian visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 5, pp. 776–789, May 2015.
- [45] D.-Y. Lee, J.-Y. Sim, and C.-S. Kim, "Multihypothesis trajectory analysis for robust visual tracking," in *Proc. IEEE CVPR*, Jun. 2015, pp. 5088–5096.
- [46] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. IEEE ICDM*, Apr. 2006, pp. 613–622.
- [47] C. Lee, W.-D. Jang, J.-Y. Sim, and C.-S. Kim, "Multiple random walkers and their application to image cosegmentation," in *Proc. IEEE CVPR*, Jun. 2015, pp. 3837–3845.
- [48] T. H. Kim, K. M. Lee, and S. U. Lee, "Generative image segmentation using random walks with restart," in *Proc. ECCV*, Oct. 2008, pp. 264–275.
- [49] J.-S. Kim, J.-Y. Sim, and C.-S. Kim, "Multiscale saliency detection using random walk with restart," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 198–210, Jun. 2013.
- [50] H. Kim, Y. Kim, J.-Y. Sim, and C.-S. Kim, "Spatiotemporal saliency detection for video sequences based on random walk with restart," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2552–2564, Aug. 2015.
- [51] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. NIPS*, Dec. 2013, pp. 315–323.
- [52] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.
- [53] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Proc. IEEE CVPR*, Jun. 2012, pp. 1830–1837.
- [54] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. BMVC*, Sep. 2006, pp. 47–56.
- [55] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. IEEE CVPR*, Jun. 2010, pp. 1269–1276.
- [56] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE CVPR*, Jun. 2006, pp. 798–805.
- [57] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE CVPR*, Jun. 2012, pp. 1822–1829.
- [58] J.-Y. Lee and W. Yu, "Visual tracking by partition-based histogram backprojection and maximum support criteria," in *Proc. IEEE ROBOT*, Jun. 2011, pp. 2860–2865.
- [59] M. Zhang, J. Xing, J. Gao, X. Shi, Q. Wang, and W. Hu, "Joint scale-spatial correlation tracking with adaptive rotation estimation," in *Proc. ICCV Workshops*, 2015, pp. 595–603.
- [60] Y. Hua, K. Alahari, and C. Schmid, "Online object tracking with proposal selection," in *Proc. IEEE ICCV*, Dec. 2015, pp. 3092–3100.
- [61] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *Proc. IEEE CVPR*, Jun. 2016, pp. 943–951.
- [62] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. ICCV Workshops*, Dec. 2015, pp. 621–629.



Han-Ui Kim (S'14) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2014, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interests include computer vision and machine learning.



Chang-Su Kim (S'95–M'01–SM'05) received the Ph.D. degree in electrical engineering from Seoul National University with a Distinguished Dissertation Award in 2000. From 2000 to 2001, he was a Visiting Scholar with the Signal and Image Processing Institute, University of Southern California, Los Angeles. From 2001 to 2003, he coordinated the 3D Data Compression Group, National Research Laboratory, for 3D Visual Information Processing in SNU. From 2003 and 2005, he was an Assistant Professor with the Department of Information Engineering, Chinese University of Hong Kong. In 2005, he joined the School of Electrical Engineering, Korea University, where he is a Professor. His research topics include image processing and computer vision. He has authored over 240 technical papers in international journals and conferences. In 2009, he received the IEEE/IEEE Joint Award for Young IT Engineer of the Year. In 2014, he received the Best Paper Award from Journal of Visual Communication and Image Representation (JVCI). He served as an Editorial Board Member of JVCI and an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING. He is a Senior Area Editor of JVCI and an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA.