SIViP (2017) 11:1073–1080
DOI 10.1007/s11760-017-1059-x

ORIGINAL PAPER

Robust Struck tracker via color Haar-like feature and selective updating

Shaojie Jiang[1,2] · **Jifeng Ning**[1,2] · **Cheng Cai**[1] · **Yunsong Li**[2]

Received: 4 June 2016 / Revised: 2 January 2017 / Accepted: 16 January 2017 / Published online: 28 January 2017

Abstract Recently, Struck—a tracker based on structured support vector machine, received great attention as a consequence of its superior performance on many challenging scenes. In this work, we present an improved Struck tracker by using color Haar-like features and effective selective updating. First, we integrate color information into Haar-like features in a simple way, which models the spatial and color information simultaneously without increasing the computational complexity. Second, we make selective model updates according to the tracking status of the object. This prevents inferior patterns resulted by occlusions, abrupt appearance or illumination changes from being added to object model, which decreases the risk of model drift problem. The experimental results indicate that the proposed tracking algorithm outperforms the original Struck by a remarkable margin in precision and accuracy, and it is competitive with other state-of-the-art trackers on a tracking benchmark of 50 challenging sequences.

Keywords Object tracking · Structured support vector machine · Haar-like feature · Selective updating

1 Introduction

The goal of object tracking is to estimate the locations of a tracked object in every frame of a video sequence. With the growing interests in video surveillance, human computer interaction, vehicle navigation, to name a few, object tracking has become an important topic in the field of computer vision. Meanwhile, object tracking is still a challenging task due to the presence of occlusion, abrupt object motion, changing appearance, background clutter and so on. Readers are referred to [18,28,29,37] for more details.

Tracking algorithms can be generally classified into two categories: generative and discriminative. Generative tracking methods maintain an appearance model of the tracked object, which is expected to account for appearance changes of the target. In this kind of method, tracking process is formulated as comparing the similarity between candidates and the model. Here are some examples of such method: eigentracker [4] utilizes eigenspace representations for targets; kernel-based tracker [7] employs a feature histogram-based target representation with spatial masking; in ℓ_1 tracker [19], targets are sparsely represented; covariance tracker [24] represents an object window as the covariance matrix of features; incremental tracker [25] learns a low-dimensional subspace representation incrementally; Zhou et al. [39] proposed a tracking algorithm based on weighted subspace reconstruction error.

Discriminative trackers usually treat tracking task as a classification problem. Rather than building an exact appearance representation of the target, discriminative trackers focus on finding decision boundaries between the object and the background. In [6], an online feature ranking mechanism is applied to adaptively select the most discriminative features for tracking. In [2], online multiple instance learning is used to help achieving robust and stable tracking performance. Global mode seeking is applied in [30] to detect the object after total occlusion and reinitialize the local tracker. SVT [1] integrates the support vector machine classifier into an optic-flow-based tracker. In [31], a generative model encoding all the appearance variations is used to reacquire

✉ Jifeng Ning
jf_ning@sina.com

[1] College of Information Engineering, Northwest A&F University, Yangling 712100, China

[2] The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China

the object, and a discriminative classifier, implemented as an online support vector machine, is trained to focus on recent appearance changes. In [9], a tracking method utilizing one-class SVM that adopts histograms of oriented gradient and 2-bit binary patterns as features is proposed. In [38], a sparsity-based generative model and a sparsity-based discriminative classifier are used collaboratively.

Hare et al. [12] note that previous discriminative trackers have a problem that, the objective for the classifier and that for tracker are not consistently coupled during learning. To overcome this problem, they use a kernelized structured SVM algorithm (named as Struck) [12] with SMO optimization method [23], which shows superior performance in recent challenging benchmark testings [21,27]. Actually, Struck treats tracking as a regression problem more than classification problem, which is more suitable for tracking task compared with the binary classifiers in previous works.

Struck employs the widely used Haar-like feature [26] as its image representation method, which is a block-wise gradient feature designed for grayscale images. However, video sequences are mostly in color nowadays, and color information is some kind of important for tracking task. Although the Haar-like feature applied to grayscale images performs well, it cannot utilize the color information of color sequences, and this may cause tracking failure at times, as shown in Fig. 1a. So if we can introduce color compatibility into Haar-like features, the spatial and color information of the target can be modeled simultaneously, thus improving the performance of Struck tracker. In Chang's work on object detection [5], they extend Haar-like feature with color compatibility by extracting Haar-like features from each channel of color image. In this paper, we propose a novel method to incorporate color information into Haar-like feature without increasing the computational complexity.

For tracking algorithms, model updates are very important to keep the tracker recognizing target appearance variations. As to discriminative trackers, the key issue is to improve the sample collection part to make the online-trained classifier more robust [22,27,32,33]. The incremental updating methods used in MIL tracker [3] and CT tracker [36] are not very effective as they treat previous samples equally. Struck [12] dynamically adjusts the relative weights (the coefficients of support vectors) of all samples by applying SSVM to learn discriminative classifier. But when conditions such as occlusions or abrupt appearance changes happen, the samples extracted from current frame are unreliable. If the classifier is updated using these samples, it will respond more to the background in the following frames, and target drift may happen gradually. Therefore, if the tracker can selectively reject unreliable samples in the updating process, the classifier should perform better.

In this work, we present an improved Struck tracker via color Haar-like feature and selective updating. We will explain the advantages of the proposed tracking method against original Struck from the aspects of feature representation and model updating. The experimental results show that our method performs better than the original Struck in precision and robustness, meanwhile it is competitive with state-of-the-art trackers according to a tracking benchmark [27] of 50 challenging video sequences.

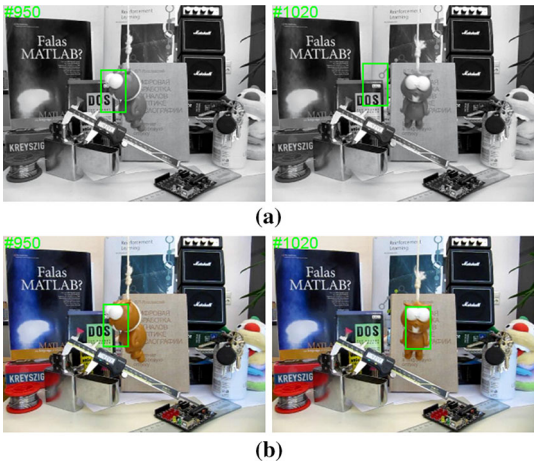

Fig. 1 Clear color distinction between the tracked doll and the background of *Lemming* sequence. **a** Tracking results of original Struck tracker, which fails at about 970th frame. Images are converted to grayscale before tracking. The background which confuses the tracker is actually very different from the target in color sequence. **b** Tracking results of the improved Struck tracker with our proposed color Haar-like feature, which will be introduced in Sect. 3.1

The rest of this paper is organized as follows: in Sect. 2, we introduce some backgrounds of our work; in Sect. 3 we describe our proposed methods in great details; in Sect. 4, we perform experiments to compare our work with original Struck [12] and other state-of-the-art algorithms; then we conclude in Sect. 5.

2 Background

In this section, we first explain the main procedure of the tracking-by-detection framework, then we introduce the traditional binary SVM and structured output SVM algorithm (Struck [12]) for tracking.

2.1 Tracking-by-detection

Object tracking task can be described as follows: with the target position given as a bounding box in the first frame of a video sequence, predict the positions of the object in the following frames. In recent years, the tracking-by-detection framework has been widely applied, since it is very suitable for tracking tasks.

Tracking-by-detection consists of two stages: learning and detection. With the object position known in the first frame, training samples are extracted (typically with binary labels) to learn a tracker. In next frame, the target position is detected by the tracker first, then learning process is applied as that in the first frame to update the tracker. These two stages are iteratively carried out this way in the following frames.

Typically, tracking-by-detection framework is used coupled with discriminative classifiers. As the two stages described above take turns to work during tracking, the classifier is updated in such an online manner, therefore tracking-by-detection algorithms are usually called *online* trackers [2,12,34].

2.2 Binary SVM tracking

Traditional SVM algorithms treat object tracking task as a binary classification problem. Let $\mathbf{p}_t \in \mathscr{P}$ be the 2D bounding box target position in the t-th frame of a video sequence: $\mathbf{f}_t \in \mathscr{F}$. With $\mathbf{p}_1$ given, features $\mathbf{x}_1^{\mathbf{p}} \in \mathscr{X}$ extracted from the first frame at positions $\mathbf{p} = \mathbf{p}_1 \circ \mathbf{y}$, where $\mathbf{y} \in \mathscr{Y}$ represents transformations (typically translations), are assigned binary labels $z = \pm 1$ according to $\mathbf{y}$, the notation $\circ$ is a transform operator. Then the sample pairs $(\mathbf{x}, z)$ are fed to the training process to learn a separating hyperplane $\mathbf{w}$, which separates positive samples from negatives. The classification function $h : \mathscr{X} \rightarrow \mathbb{R}$ is given as $h(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$, the label of a new feature vector $\mathbf{x}$ is determined by $\hat{z} = \mathrm{sign}(h(\mathbf{x}))$.

At frame $\mathbf{f}_t$, the target position of previous frame $\mathbf{f}_{t-1}$ is estimated as $\mathbf{p}_{t-1}$. Then the goal of tracking the target at this frame is to find a transformation $\mathbf{y}_t$. The best estimation of the transformation is found according to :

$$\mathbf{y}_t = \underset{\mathbf{y}\in\mathscr{Y}}{\mathrm{argmax}}\ h\left(\mathbf{x}_t^{\mathbf{p}_{t-1}\circ\mathbf{y}}\right), \tag{1}$$

and the best target position of frame $\mathbf{f}_t$ is $\mathbf{p}_t = \mathbf{p}_{t-1} \circ \mathbf{y}_t$. After $\mathbf{p}_t$ being estimated, features $\mathbf{x}_t^{\mathbf{p}}$ are extracted with binary labels. Then these new samples are used to update the classifier $\mathbf{w}$.

The main drawback of this process is that classifier is trained without spatial information of the samples, because of the projection from transformations $\mathbf{y}$ to binary labels z.

2.3 Struck: structured SVM tracker

To deal with the problem mentioned above, Struck [12] introduces structured output SVM (SSVM) into object tracking. The main procedure of SSVM for tracking is the same as binary SVM, in this part we mainly focus on primary differences.

SSVM discards the projection operation by using the transformations $\mathbf{y}$ directly as its structured labels. So the sample pair is now written as $(\mathbf{x}, \mathbf{y})$, and we denote the sample pair with joint kernel map as $\Phi(\mathbf{x}, \mathbf{y})$.

Similar as binary SVM, the separating hyperplane is learned through a convex quadratic programming function:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2a}$$

$$s.t.\ \forall i : \xi_i \geq 0 \tag{2b}$$

$$\forall i, \forall \mathbf{y} \neq \mathbf{y}_i : \langle \mathbf{w}, \delta\Phi_i(\mathbf{y})\rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \tag{2c}$$

where $\delta\Phi_i(\mathbf{y}) = \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y})$. The difference between this function and that of binary SVM lies in (2c). The corresponding constraint inequation in binary SVM is:

$$z_i \langle \mathbf{w}, \mathbf{x}\rangle \geq 1 - \xi_i . \tag{3}$$

Because of the discard of binary labels z, (3) is inappropriate for SSVM.

The $\Delta(\mathbf{y}_i, \mathbf{y})$ in (2c) denotes a loss function, which shows the dissimilarity between a transformation $\mathbf{y}$ and the best transformation $\mathbf{y}_i$. It is typically based on the bounding box overlap:

$$\Delta(\mathbf{y}_i, \mathbf{y}) = 1 - s^o_{\mathbf{p}_t}(\mathbf{y}_i, \mathbf{y}), \tag{4}$$

where

$$s^o_{\mathbf{p}_t}(\mathbf{y}_i, \mathbf{y}) = \frac{(\mathbf{p}_t \circ \mathbf{y}_i) \cap (\mathbf{p}_t \circ \mathbf{y})}{(\mathbf{p}_t \circ \mathbf{y}_i) \cup (\mathbf{p}_t \circ \mathbf{y})} \tag{5}$$

is the overlap function.

To solve the function (2a)–(2c) for $\mathbf{w}$, Struck [12] typically employs Lagrangian duality technique and SMO optimization step [23]. The solving process is beyond the scope of this work, we refer the readers to [12,23] for detailed descriptions.

3 The proposed tracking method

In this section, we introduce the improved Struck tracker from two aspects. First, we improve the Haar-like feature representation by incorporating multi-channel information into it (referred as Color Haar-like), which is originally designed to deal with grayscale images only. Second, we propose a selective update scheme (SUS) to prevent inferior model updates.

3.1 Object representation with color Haar-like feature

Haar-like feature representation is widely used by modern tracking algorithms [2,12,17], since it is simple and effective. But Haar-like features were commonly used to process

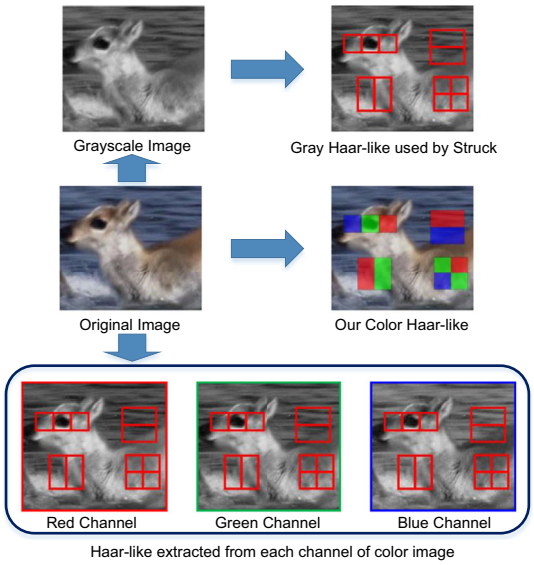

Fig. 2 Compare our color Haar-like with original gray Haar-like and existing color-based method [5]. Our method extracts each patch of a Haar-like block in a randomly selected channel, so every block is integrated with color information and the dimensionality is equal to gray Haar-like. While the existing method extracts Haar-like blocks from every channel of the image, respectively

grayscale images. When the sequence is in color, images need to be converted to gray scale, thus causing loss of color information.

For example in Fig. 1a, the color of the doll being tracked is very different from the background in color images. But when converted to grayscale images, the difference becomes inconspicuous, and this results in the tracker's failure eventually. Moreover, the dual optimization strategy and Gaussian kernel function used by Struck [12] require low-dimensional feature vectors, otherwise the tracking procedure will be very time-consuming. Here, we propose a method to incorporate multi-channel information into Haar-like feature elegantly without increasing its dimension. Note that the color Haar-like feature mentioned here was called multi-channel Haar-like feature in our preliminary work [20].

In our method, each patch of the Haar-like block is calculated in one of the three channels randomly. Our method utilizes the same six types of Haar-like feature arranged on a 4×4 grid at 2 scales, resulting a 192D feature vector. Although patches are selected from all three channels, the dimension of feature vector is the same as Struck's. Compared with the existing color-based Haar-like feature [5] as mentioned in Sect. 1, their method will result in a $192 \times 3 = 576D$ feature vector for each sample, which will be more time-consuming than ours. The illustration of the differences between their method and ours is shown in Fig. 2, and we will show that our method is more effective in the experiments section.

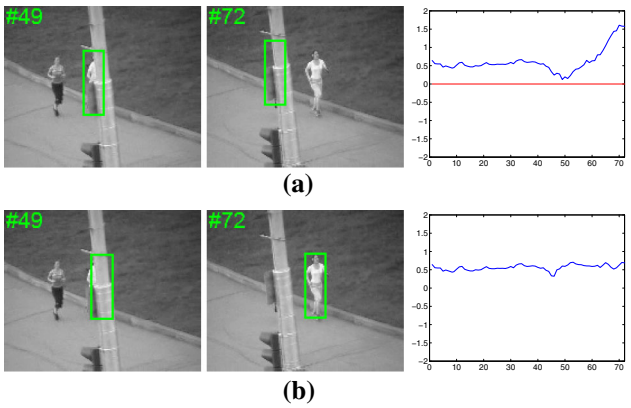

Fig. 3 **a** Two screen shots and the line chart of history scores of the *Jogging-2* sequence. It is shown that classification score changes slowly when the object is well tracked before frame 45, where the target is becoming occluded. Then the original Struck tracker lost the target and the classification scores become very unstable in the following frames. **b** By using our proposed selective update scheme, the improved Struck tracker managed to recapture the target after occlusion. Although the score decreases to some extent when occlusion happens, the overall curve is much more smooth than that of (**a**). **a** Original Struck tracker with occlusion, **b** improved Struck tracker with occlusion

By using color Haar-like feature, the improved Struck tracker tracks the doll successfully as shown in Fig. 1b. We will show more comparisons of our color Haar-like feature representation against original gray Haar-like in the experiments section.

3.2 Selective update scheme

In this part, we introduce the proposed selective update scheme, which is used to prevent the discriminative classifier from being contaminated by bad training samples.

We observed that when target appearance changes smoothly and slowly, the classification scores of the target will fluctuate mildly. Since the classifier keeps being updated to recognize the target well, when unexpected conditions (such as occlusions, abrupt target appearance changes and drastic illumination changes) happen, the classification scores will decline significantly. Figure 3a is an example of occlusion. As we can see, before occlusion happens at about 45th frame, the classification scores are relatively stable. But when the target is occluded, scores drop to 0 drastically. Since the target is occluded, the training samples extracted from this frame contain lots of noises. If the classifier is updated using these unreliable samples, the subsequent classification scores will become unreliable, then the tracker may get lost into backgrounds.

To deal with this problem, we propose a selective update scheme (SUS) to detect unexpected conditions right after they happen. Then the update process is stopped to protect the classifier from being contaminated.

Algorithm 1 Color Struck with Selective Update Scheme

Input: $\mathbf{p}_1$, video sequence $\mathbf{f}_t$ $(t = 1 \ldots N)$
Output: $\mathbf{p}_t$
$\mathbf{w}$ = Initialize($\mathbf{f}_1, \mathbf{p}_1$)
for i=2:N **do**
  s_i = GetScore($\mathbf{w}, \mathbf{f}_i, \mathbf{p}_{i-1}$)
  Calculate u_t
  if $u_t == 0$ **then**
    Update w by Struck with color Haar-Like feature
    $\mathbf{p}_i$ = GetPosition($s_i, \mathbf{f}_i$)
  else
    Skip update
    Remove s_i from history scores
    $\mathbf{p}_i = \mathbf{p}_{i-1}$
  end if
end for

We base our assumption on the observation that, classification score changes slowly with smooth target appearance variation. Therefore, in a narrow time window, the scores will be close to each other. Let u_t denote the detection state of unexpected conditions at frame $\mathbf{f}_t$. The best score of $\mathbf{f}_t$ is denoted as s_t, and k is the time window. Then we define:

$$u_t = \begin{cases} 1 & \text{if } s_t < \dfrac{a}{k} \displaystyle\sum_{i=t-k}^{t-1} s_i \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where a is a scaling factor. It means unexpected conditions are detected when $u_t = 1$, and the update process of frame $\mathbf{f}_t$ is skipped. As shown in Fig. 3b, our improved tracker manages to recapture the target after occlusion in the *Jogging-2* sequence, and the score chart becomes more smooth. The overall effect of our SUS scheme will be shown in the experiments section.

3.3 The proposed tracking algorithm

In order to clearly show the position where our color Haar-like feature and SUS scheme work, we give the overall tracking process in Algorithm 1. Note that Algorithm 1 focuses on the main process of our proposed algorithm, some details are omitted to keep the conciseness. To realize more concrete explanations of the update process and sampling strategy, please refer to [12].

4 Experiments and analysis

In this section, we evaluate our proposed tracker by comparing it with original Struck and some other state-of-the-art trackers on a benchmark dataset [27]. First we introduce the experimental setups, then we perform comparative experiments. We run our C++ implementation on a PC with Intel E5-2650 CPU (2.30 GHz) and 32 GB memory. The source codes of this paper will be released to public for repeatable research.

4.1 Experimental setup

Parameter settings For all sequences, we use fixed parameter values to perform robustness evaluation. The six different types of color Haar-like features are arranged on a 4×4 grid of 2 scales with each feature normalized to $[-1, 1]$. We set the time window $k = 10$, and the scaling factor $a = 0.6$ for SUS. The other parameters are remained the same as the default settings in Struck: σ is set to 0.2 for Gaussian kernel, penalty parameter $C = 100$; during tracking, search radius $r = 30$ pixels is used; in the updating process a polar grid of 5 radial and 16 angular divisions with $r = 60$ is used; budget size $B = 100$ for support vector maintenance.

Benchmark dataset We evaluate our tracker on a benchmark dataset of 50 video sequences [27]. The sequences are annotated with 11 attributes: 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 clutters (BC) and low resolution (LR). For robustness evaluation, we perform all three tests provided by the benchmark [27]: one-pass evaluation (OPE), temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). OPE initializes the tracker with the ground truth position in the first frame and then run it throughout the test sequence. TRE divides each sequence into 20 fragments and initializes the tracker with the ground truth position of each fragment, then the tracker runs to the end of the sequence. SRE initializes the trackers with 12 different disturbances of the bounding box in the first frame, including eight spatial shifts and four scale variations. Each of the three tests is evaluated through precision plot and success plot. Precision plot measures the average Euclidean distance between the center locations of tracked targets and the manually labeled ground truths, the score for the threshold = 20 pixels is used as the representative precision score. Success plot measures the overlap of tracked bounding box and the ground truth, it shows the ratios of successful frames at the thresholds varied from 0 to 1, and the score is calculated by the area under the curve (AUC).

4.2 Compare with original struck

The performance comparisons of our improved Struck tracker and original Struck [12] are listed in Table 1. For convenience, we denote Struck with color Haar-like as *C-Struck* and Struck with color Haar-Like and SUS as *CS-Struck*. *C2-Struck* uses the feature extraction method proposed in [5],

Table 1 Comparisons between original Struck and our improved Struck tracker

	OPE		TRE		SRE		FPS
	Precision	Success	Precision	Success	Precision	Success	
Struck	0.656	0.474	0.707	0.514	0.634	0.449	10.801
Struck2015	–	–	**0.785**	0.545	*0.707*	0.469	–
C-Struck	*0.713*	*0.498*	0.779	*0.553*	*0.707*	*0.488*	10.193
C2-Struck	0.683	0.487	0.759	0.547	0.666	0.467	3.075
CS-Struck	**0.747**	**0.520**	*0.782*	**0.555**	**0.717**	**0.494**	10.112

The results of *Struck* is given by [27]. In the recent version of Struck [11], they made some improvements and gave out the TRE and SRE scores on benchmark [27], here we refer it as *Struck2015*. *C-Struck* uses our proposed color Haar-like feature, *C2-Struck* uses the Haar-like feature proposed in [5]. *CS-Struck* uses the color Haar-like and SUS proposed in this paper simultaneously. The entries in bold indicate the best results, and the ones in italic indicate the second best

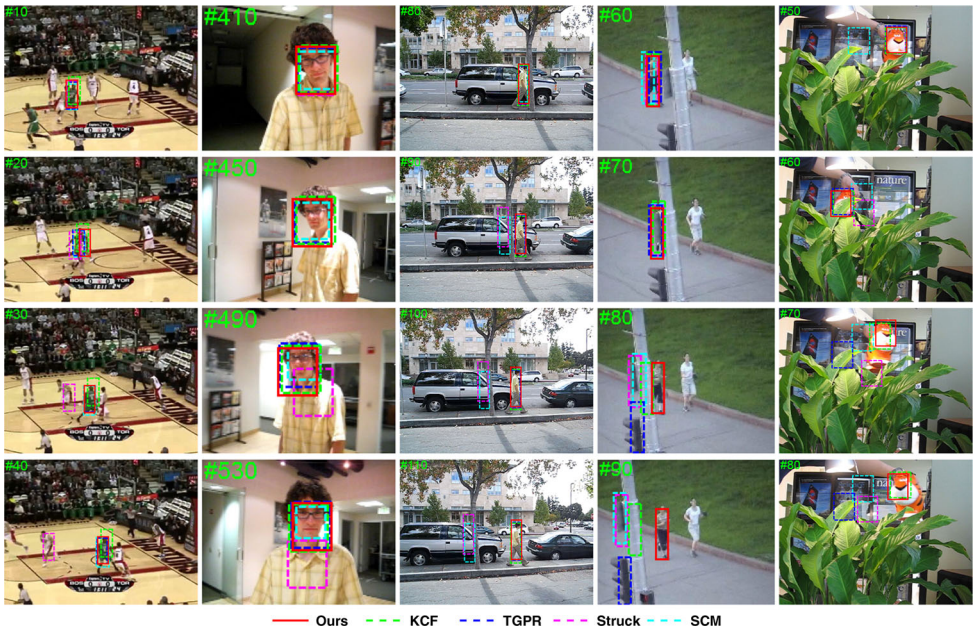

Fig. 4 Tracking comparisons of sequences *Basketball*, *David*, *David3*, *Jogging-1*, *Tiger1*, one sequence in each column. Our tracker is the improved Struck with color Haar-like feature and SUS

they extract Haar-like features from each of the image channels, respectively, as described in Sect. 3.1.

From Table 1, we can see that compared with *Struck*, *C-Struck* obviously improves OPE precision and success ratio from 0.656 and 0.474 to 0.713 and 0.498, respectively. Furthermore, *CS-Struck* performs better than C-Struck in all indexes as it prevents bad sample from polluting the target model, it takes the first places in almost every index except TRE precision.

In their newer version of *Struck2015* [11], they made some improvements in feature representation by using multi-feature and multi-kernel. They use 192D gray Haar-like feature with Gaussian kernel and 480D Histogram feature with intersection kernel, resulting in a 672D feature vector for each sample. Note that in *C-Struck*, we use our color Haar-like feature of 192D, but our tracker still performs better than *Struck2015* in the success scores of TRE and SRE and have the same score in SRE precision.

C2-Struck extracts a 576D feature vector for each sample including color information, and this improves the original Struck in all indexes. But their feature extraction method [5] performs worse than ours as the comparison of *C-Struck* and *C2-Struck* shown in Table 1. Besides, as the dimensionality of the feature vectors extracted by their method is greater (3 times of ours), the speed of their method is much slower.

To fairly compare the tracking speed, we record the FPS of a same video sequence (*Basketball*). It can be seen that *C-Struck* and *CS-Struck* barely decrease the tracking speed. The reason is that the computational complexity of the proposed color Haar-like feature is almost the same as gray Haar-like except that it needs to compute the integral images of three channels.

4.3 Compare with state-of-the-art trackers

In this part, we perform comprehensive evaluation to compare our tracker (CS-Struck) with several state-of-the-art

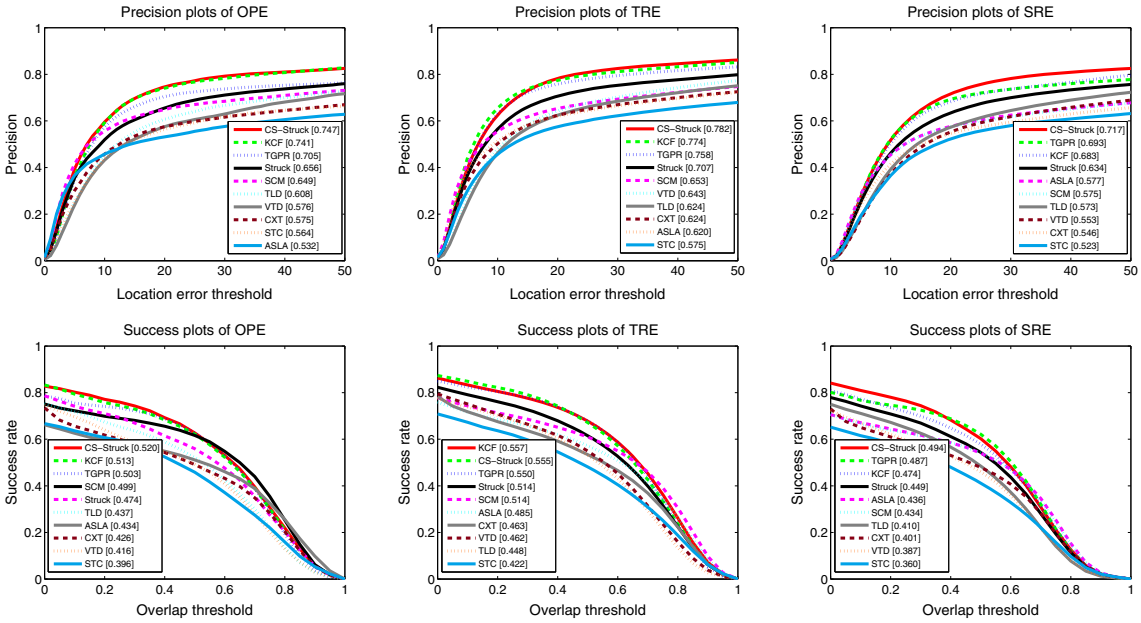

Fig. 5 Compare with state-of-the-art trackers on TB50

trackers on the benchmark TB50 [27]. Besides Struck [12], we choose other top 5 trackers presented in [27], they are: SCM [38], TLD [15], ASLA [14], CXT [8] and VTD [16]. In addition, we add three more recent trackers in our comparison, they are: STC [35], KCF [13] and TGPR [10].

To perform qualitative comparisons, we display some screen shots of the five best trackers in Fig. 4. From the screen shots of sequences *Basketball* and *David*, we can see that as color information is used, the distinction between target and background is more obvious, so our tracker manages to overcome the challenges which make the original Struck tracker fail (as indicated by purple rectangles). From sequences *David3*, *Jogging-1* and *Tiger1*, it can be seen that our tracker deals with occlusions very well.

For quantitative comparisons, the overall results with precision and success plots are shown in Fig. 5. Although STC [35] exploits the dense spatial-temporal context for tracking with fast speed, it does not rank top 10 on the tracking benchmark [27]. This indicates that it is not robust enough for changing environments except fast speed. TGPR [10] robustly handle tracking via Gaussian Processes Regression on this benchmark with a low speed of about 0.5 fps. Finally, KCF [13], as a very fast tracker with the skill of circulant matrix recently get state-of-the-art evaluation on this benchmark. Compared with KCF, our tracker obtains competitive performance. Although the proposed tracking method is slower than KCF, it outperforms KCF slightly in all indexes except the success ratio of TRE. Besides, since SRE tests contain spatial perturbations, the high score in SRE indicates that our tracker is more spatially robust than other trackers.

5 Conclusion

In this work, we present an improved Struck tracker by using Color Haar-like feature and selective update scheme. First, the Color Haar-like features naturally integrate spatial and color information of the target without increasing the dimensionality of the feature vector, which is better than gray Haar-like feature for object representation. Second, by using selective update scheme, our tracking method prevents bad samples from being added into training set, which reduces the risk of model drift problem of the trained discriminative classifier. The experiments on tracking benchmark show that our improved Struck tracker performs better than original Struck with a large margin, meanwhile it also gets competitive results compared with other state-of-the-art trackers.

Acknowledgements This work is supported by Shaanxi Province Natural Science Foundation under Grants 2015JM3110, the Program of the State Key Laboratory of Integrated Services Networks under Grant ISN17-08 and the Fundamental Research Funds for the Central Universities under Grant QN2013055.

References

1. Avidan, S.: Support vector tracking. IEEE Trans. Pattern Anal. Mach. Intell. **26**(8), 1064–1072 (2004)
2. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 983–990. IEEE (2009)
3. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1619–1632 (2011)

4. Black, M.J., Jepson, A.D.: Eigentracking: robust matching and tracking of articulated objects using a view-based representation. Int. J. Comput. Vis. **26**(1), 63–84 (1998)
5. Chang, W.C., Cho, C.W.: Multi-class boosting with color-based haar-like features. In: IEEE Conference on Signal-Image Technologies and Internet-Based System, pp. 719–726. IEEE (2007)
6. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1631–1643 (2005)
7. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–577 (2003)
8. Dinh, T.B., Vo, N., Medioni, G.: Context tracker: exploring supporters and distracters in unconstrained environments. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1177–1184. IEEE (2011)
9. Fu, K., Gong, C., Qiao, Y., Yang, J., Gu, I.Y.H.: One-class support vector machine-assisted robust tracking. J. Electron. Imaging **22**(2), 023,002–023,002 (2013)
10. Gao, J., Ling, H., Hu, W., Xing, J.: Transfer learning based visual tracking with gaussian processes regression. In: European Conference on Computer Vision, pp. 188–203. Springer (2014)
11. Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M.M., Hicks, S.L., Torr, P.H.S.: Struck: structured output tracking with kernels. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 2096–2109 (2016)
12. Hare, S., Saffari, A., Torr, P.H.: Struck: structured output tracking with kernels. In: IEEE International Conference on Computer Vision, pp. 263–270. IEEE (2011)
13. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2015)
14. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1822–1829. IEEE (2012)
15. Kalal, Z., Matas, J., Mikolajczyk, K.: Pn learning: bootstrapping binary classifiers by structural constraints. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 49–56. IEEE (2010)
16. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1269–1276. IEEE (2010)
17. Leistner, C., Saffari, A., Bischof, H.: Miforests: multiple-instance learning with randomized trees. In: European Conference on Computer Vision, pp. 29–42. Springer (2010)
18. Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., van den Hengel, A.: A survey of appearance models in visual object tracking. ACM Trans. Intell. Syst. Technol. **4**(4), 58 (2013)
19. Mei, X., Ling, H.: Robust visual tracking using ℓ_1 minimization. In: IEEE International Conference on Computer Vision, pp. 1436–1443. IEEE (2009)
20. Ning, J., Zhao, Y., Shi, W.: Multiple instance learning based object tracking with multi-channel haar-like feature. J. Image Graph. **19**(7), 1038–1045 (2014). **(in Chinese)**
21. Pang, Y., Ling, H.: Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms. In: IEEE International Conference on Computer Vision, pp. 2784–2791. IEEE (2013)
22. Phadke, G., Velmurugan, R.: Mean lbp and modified fuzzy c-means weighted hybrid feature for illumination invariant mean-shift tracking. Signal Image Video Process. (2016). doi:10.1007/s11760-016-1008-0
23. Platt, J.: Fast training of support vector machines using sequential minimal optimization. Adv. Kernel Methods Support Vector Learn **3**, 185–208 (1999)
24. Porikli, F., Tuzel, O., Meer, P.: Covariance tracking using model update based on lie algebra. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 728–735. IEEE (2006)
25. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1–3), 125–141 (2008)
26. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. I–511–I–518. IEEE (2001)
27. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: a benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2411–2418. IEEE (2013)
28. Yang, H., Shao, L., Zheng, F., Wang, L., Song, Z.: Recent advances and trends in visual tracking: a review. Neurocomputing **74**(18), 3823–3831 (2011)
29. Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. **38**(4), 13 (2006)
30. Yin, Z., Collins, R.T.: Object tracking and detection after occlusion via numerical hybrid local and global mode-seeking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
31. Yu, Q., Dinh, T.B., Medioni, G.: Online tracking and reacquisition using co-trained generative and discriminative trackers. In: European Conference on Computer Vision, pp. 678–691. Springer (2008)
32. Zhang, B., Li, Z., Perina, A., Bue, A.D., Murino, V., Liu, J.: Adaptive local movement modeling (almm) for robust object tracking. IEEE Trans. Circuits Syst. Video Technol. **PP**(99), 1–1 (2016). doi:10.1109/TCSVT.2016.2540978
33. Zhang, B., Perina, A., Li, Z., Murino, V., Liu, J., Ji, R.: Bounding multiple gaussians uncertainty with application to object tracking. Int. J. Comput. Vis. **118**(3), 364–379 (2016)
34. Zhang, K., Song, H.: Real-time visual tracking via online weighted multiple instance learning. Pattern Recognit. **46**(1), 397–411 (2013)
35. Zhang, K., Zhang, L., Liu, Q., Zhang, D., Yang, M.H.: Fast visual tracking via dense spatio-temporal context learning. In: European Conference on Computer Vision, pp. 127–141. Springer (2014)
36. Zhang, K., Zhang, L., Yang, M.H.: Fast compressive tracking. IEEE Trans. Pattern Anal. Mach. Intell. **36**(10), 2002–2015 (2014)
37. Zhang, R., Liu, X., Hu, J., Chang, K., Liu, K.: A fast method for moving object detection in video surveillance image. Signal Image Video Process. (2016). doi:10.1007/s11760-016-1030-2
38. Zhong, W., Lu, H., Yang, M.H.: Robust object tracking via sparsity-based collaborative model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1838–1845. IEEE (2012)
39. Zhou, T., Xie, K., Zhang, J., Yang, J., He, X.: Robust object tracking based on weighted subspace reconstruction error with forward: backward tracking criterion. J. Electron. Imaging **24**(3), 033,005–033,005 (2015)