Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 402185, 13 pages
http://dx.doi.org/10.1155/2014/402185

Research Article

Part-Based Visual Tracking via Online Weighted P-N Learning

Heng Fan,[1] Jinhai Xiang,[2] Jun Xu,[3] and Honghong Liao[4]

[1] *College of Engineering, Huazhong Agricultural University, Wuhan 430070, China*
[2] *College of Science, Huazhong Agricultural University, Wuhan 430070, China*
[3] *Department of Physics, Central China Normal University, Wuhan 430079, China*
[4] *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Correspondence should be addressed to Jinhai Xiang; jimmy_xiang@163.com

Received 11 April 2014; Accepted 11 June 2014; Published 15 July 2014

Academic Editor: Yu-Bo Yuan

We propose a novel part-based tracking algorithm using online weighted P-N learning. An online weighted P-N learning method is implemented via considering the weight of samples during classification, which improves the performance of classifier. We apply weighted P-N learning to track a part-based target model instead of whole target. In doing so, object is segmented into fragments and parts of them are selected as local feature blocks (LFBs). Then, the weighted P-N learning is employed to train classifier for each local feature block (LFB). Each LFB is tracked through the corresponding classifier, respectively. According to the tracking results of LFBs, object can be then located. During tracking process, to solve the issues of occlusion or pose change, we use a substitute strategy to dynamically update the set of LFB, which makes our tracker robust. Experimental results demonstrate that the proposed method outperforms the state-of-the-art trackers.

1. Introduction

Object tracking is one of the most important components of many applications in computer vision, such as human computer interactions, surveillance, and robotics. However, robust visual tracking is still a challenging problem, which is affected by partial or full occlusion, illumination, scale, and poses variation [1]. The key for the object tracking is to construct an effective appearance model. Many tracking algorithms have been proposed recently, but designing a robust appearance model is still a major challenge, which is affected by both extrinsic (e.g., illumination variation, background clutter, and partial or full occlusion) and intrinsic (e.g., scale and pose variation) factors. In order to handle these problems, a wide range of appearance models based on different visual representations and statistical modeling techniques have been presented by researchers. In general, these appearance models can be categorized into two types: appearance model based on visual representation, such as global-based representation [2–6] and local-based representation [7–11]; appearance model based on statistical modeling, such as generative model [12–16] and discriminative model [5, 6, 17–20].

In this paper, we propose a part-based visual tracking algorithm with online weighted P-N learning. Weighted P-N learning is first proposed by assigning weights (property weight and classification weight) to each sample in training sample set, which can decrease false classification and improve the discriminative power of classifier. Then, we segment object into fragments and select parts of them as local feature blocks to represent the object. Finally, we train classifier for each LFB with weighted P-N learning to obtain the corresponding classifier, respectively, and track each LFB independently within the framework of Lucas-Kanade optical flow [21]. During tracking process, a real-time valid detection method is used for each LFB. If certain LFB is invalid, we use a replacing strategy to update the local feature block set, which can ensure successful tracking.

Contributions. The contributions of this paper include the following.

(i) A part-based visual tracking algorithm with online weighted P-N learning is proposed in this work. Object is represented by LFBs and tracked. When occlusion or distortion happens, a strategy is adopted to replace invalid LFB and keep the new LFB set effective.

(ii) We define the weights (property weight and classification weight) for each sample in training process of P-N learning.

(iii) An online weighted P-N learning is presented by assigning weight to each sample in training sample set, which can improve discriminative power of classifier by decreasing classification errors and increasing the accuracy of tracker.

The rest of the paper is organized as follows. Section 2 reviews the related work of this paper. Section 3 introduces weighted P-N learning. Proposed tracking method is presented in detail in Section 4. Experimental results are shown in Section 5. Section 6 concludes the whole paper.

2. Related Work

Recently, many trackers based on local feature representation have been proposed. Adam et al. [8] present a fragment-based tracking approach, and further, Wang et al. [22] embed the fragment-based method into mean shift tracking framework. This tracking method estimates the target based on voting map of each part via comparing its histogram with the template's. Nevertheless, static template with equal importance being assigned to each fragment obviously lowers the performance of tracker. In order to overcome the shortcomings, Jia et al. [7] propose a fragment-based tracking method using online multiple kernel learning (MKL). All the patches are assigned to different weights based on the importance learned by MKL. However, this strategy may still cause drifting problem. Occlusion especially, which makes part patches invalid, leads to errors in computing voting map, even tracking failure. Wang et al. [10] introduce a tracking method based on superpixel. It only computes the probabilities of superpixels belonging to target, which is prone to drift away in color-similar background and whose tracking results will shrink to the unoccluded part of object when occlusion happens.

Another type of tracking method is based on discriminative appearance model. Tang et al. [23] present a tracking method based on semisupervised support vector machines. This tracker employs a small number of labeled samples for semi-supervised learning and develops a classifier to mark the unlabeled data. Babenko et al. [5] propose a multiple instance learning (MIL) method for visual tracking. This approach solves the problem of slight inaccuracies in the tracker leading to incorrectly labeled training samples and can alleviate drift problem to some extent. However, the MIL tracker might detect positives, which are less important because they do not consider the importance of sample in learning process. Further, Zhang and Song [20] suggest a weighted multiple instance learning (WMIL) tracking method. It assigns weight to each sample based on the corresponding importance. This approach improves the robustness of tracker. Kalal et al. [17] propose a method called P-N learning, which learns from positive samples and negative samples, to construct a classifier. In the meanwhile, the discriminative properties of classifier are improved by two categories of constrains that are termed P-constrains and N-constrains. However, false classification in P-N learning degrades the classifier in some degree.

Another work similar to ours is [9], which utilizes blocks to represent the object. However, the blocks are easily invalided when target appearance changes, which undermines its robustness to nonrigid distortion or occlusion. In our work, we employ LFBs to represent target and use a dynamically updating mechanism to update the local feature block set, which guarantees each LFB in the set is valid when occlusion or deformation occurs. Hence, our tracker is more robust and effective.

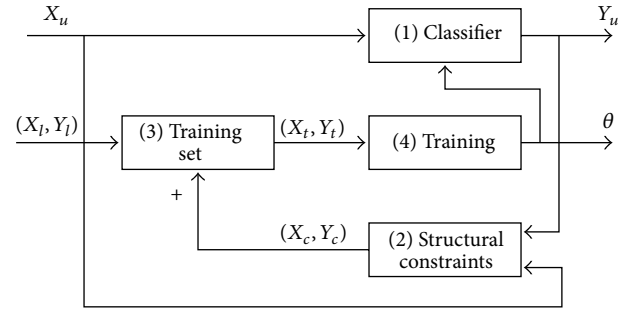


Figure 1: A flowchart in [17] to explain P-N learning. P-N learning algorithm initially develops a classifier from prior knowledge and then iterates over: (1) classify the unlabeled data and label it; (2) reclassify the samples within constrains and label them; (3) expand the training sample set; (4) retrain the classifier.

3. Weighted P-N Learning

P-N learning is a semisupervised online learning algorithm proposed by Kalal et al. [17, 24–26]. Let x be a sample in feature space $\mathcal{X}$, and let y be a label in label space $\mathcal{Y} = \{1, -1\}$. A set of samples X and corresponding set of labels Y are defined as (X, Y), which is termed a labeled set. The aim of P-N learning is to develop a binary classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ based on a priori labeled set (X_l, Y_l) and improve its discriminative performance by unlabeled data X_u. The flowchart of P-N learning approach in [17] is shown in Figure 1.

3.1. Classifier Bootstrapping. The binary classifier f is a function parameterized by θ. Similar to supervised learning, P-N learning is to estimate parameter θ via training sample set (X_t, Y_t). Nevertheless, it is worth noticing that the training set is iteratively expanded through adding samples, which is screened by constraints from unlabeled data. Initially, classifier and its parameter θ^0 are obtained by training labeled

samples. Then, the process proceeds iteratively. In iteration k, all the unlabeled samples are marked by classifier in iteration $k-1$; namely,

$$y_u^k = f\left(x_u \mid \theta^{k-1}\right), \quad x^u \in X_u. \tag{1}$$

Then, the constraints are utilized to revise the classification results and add the corrected labels to training set. Iteration k ends with retraining classifier by using the renewed training sample set.

During training process, the classifier may identify the unlabeled data with wrong labels. Any sample may be screened many times with constraints and hence can be represented mistakenly in the training set repeatedly. Obviously, it can significantly degrade discriminative performance of classifier and therefore lower the accuracy of tracking. In order to further improve the accuracy and robustness of the classifier, we propose a weighted P-N learning method by assigning weight to each sample in training set. Sample j in training set has two categories of weight which are termed P-weight W_j^+ and N-negative W_j^-. P-weight represents the probability of being a positive sample, and N-weight represents the probability of being a negative sample. In iteration k, sample j from training set is represented as positive sample for C_j^+ times and as negative sample for C_j^- times. The positive weight $W1_j^+$ and negative weight $W1_j^-$ are determined by the following formulation:

$$W1_j^+ = \frac{C_j^+}{C_j^+ + C_j^-},$$
$$W1_j^- = \frac{C_j^-}{C_j^+ + C_j^-}. \tag{2}$$

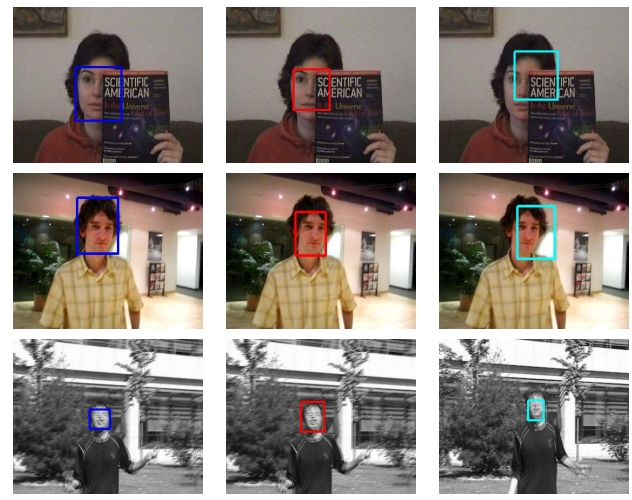
Besides, the probability of sample j being positive or negative in training set obtained by classifier is defined as classification weights $W2_j^+$ and $W2_j^-$ (in Section 3.3). The P-weight and N-weight of sample j can be then obtained by the following formulation:

$$W_j^+ = W1_j^+ + W2_j^+,$$
$$W_j^- = W1_j^- + W2_j^-. \tag{3}$$

At last, sample j is determined to be either positive or negative via the following formulation:

$$\frac{W_j^+}{W_j^-} \geq 1 \quad \text{positive sample},$$
$$\frac{W_j^+}{W_j^-} < 1 \quad \text{negative sample}. \tag{4}$$

Figure 2 demonstrates the tracking results with weighted P-N learning. In Figure 2, the left and middle images are the ground truth and tracking results with weighted P-N learning, and the right images are tracking results based on P-N learning.



(a) (b) (c)

Figure 2: (a) Ground truth. (b) Tracking results with weighted P-N learning. (c) Tracking results based on P-N learning.

3.2. Constraints. In P-N learning, a constraint can be arbitrary function, especially two categories of constraints which we term P and N. P-constrains recognize samples which are labeled negative by the classifier, yet constraints need a positive label. P-constraints add $n^+(k)$ samples to the training set in iteration k. Similarly, N-constraints are employed to identify samples classified as positive but constraints require negative label. In iteration k, N-constraints insert $n^-(k)$ samples to training set.

In iteration k, the error of a classifier is represented by a number of false positives $\alpha(k)$ and a number of false negatives $\beta(k)$. Let $n_c^+(k)$ be the number of samples for which the label is correctly changed to positive in iteration k by P-constraints, and $n_f^+(k)$ is then the number of samples for which the label is incorrectly changed to positive in iteration k. Hence, P-constraints change $n^+(k) = n_c^+(k) + n_f^+(k)$ samples to positive. Similarly, N-constraints change $n^-(k) = n_c^-(k) + n_f^-(k)$ samples to negative, where $n_c^-(k)$ and $n_f^-(k)$ are correct and false assignments. The errors of classifier can be represented as the following formulations:

$$\alpha(k+1) = \alpha(k) - n_c^-(k) + n_f^+(k), \tag{5}$$

$$\beta(k+1) = \beta(k) - n_c^+(k) + n_f^-(k). \tag{6}$$

Equation (5) demonstrates that false positives $\alpha(k)$ decrease if $n_c^-(k) > n_f^+(k)$. In the similar way, false negatives $\beta(k)$ decrease if $n_c^+(k) > n_f^-(k)$. To analyze the convergence of learning process, a model needs to be developed that relates the performance of P-N constraints to $n_c^+(k)$, $n_c^-(k)$, $n_f^+(k)$, and $n_f^-(k)$.

The performance of P-N constraints is represented by four indexes, P-precision P^+, P-recall R^+, N-precision P^-, and

N-recall R^-, determined by the following formulation:

$$P(k)^+ = \frac{n_c^+(k)}{n_c^+(k) + n_f^+(k)},$$
$$R(k)^+ = \frac{n_c^+(k)}{\beta(k)},$$
$$P(k)^- = \frac{n_c^-(k)}{n_c^-(k) + n_f^-(k)}, \quad (7)$$
$$R(k)^- = \frac{n_c^-(k)}{\alpha(k)}.$$

According to formulation (7), it is easy to get

$$n_c^+(k) = R(k)^+ \cdot \beta(k),$$
$$n_f^+(k) = \frac{1 - P(k)^+}{P(k)^+} \cdot R(k)^+ \cdot \beta(k),$$
$$n_c^-(k) = R(k)^- \cdot \alpha(k), \quad (8)$$
$$n_f^-(k) = \frac{1 - P(k)^-}{P(k)^-} \cdot R(k)^- \cdot \alpha(k).$$

By combining formulations (5), (6), and (8), we can obtain new formulations:

$$\alpha(k+1) = \left(1 - R(k)^-\right) \cdot \alpha(k) + \frac{1 - P(k)^+}{P(k)^+} \cdot R(k)^+ \cdot \beta(k),$$
$$\beta(k+1) = \frac{1 - P(k)^-}{P(k)^-} \cdot R(k)^- \cdot \alpha(k) + \left(1 - R(k)^+\right) \cdot \beta(k). \quad (9)$$

After defining state vector $\vec{x}(k) = [\alpha(k) \ \beta(k)]^T$ and transition matrix $\mathbf{M}$ as the following,

$$\mathbf{M} = \begin{bmatrix} 1 - R(k)^- & \frac{1 - P(k)^+}{P(k)^+} \cdot R(k)^+ \\ \frac{1 - P(k)^-}{P(k)^-} \cdot R(k)^- & 1 - R(k)^+ \end{bmatrix}, \quad (10)$$

hence formulation (9) can be rewritten as the following formulation:

$$\vec{x}(k+1) = \mathbf{M}\vec{x}(k). \quad (11)$$

According to [27], formulation (11) is a recursive equation that is related to a discrete dynamical system. Based on the theory of dynamical systems, the state vector $\vec{x}$ converges to zero if eigenvalues λ_1 and λ_2 of the transition matrix $\mathbf{M}$ meet the condition $\lambda_1 < 1$ and $\lambda_2 < 1$. As pointed in [17], the performance of classifier will be improved constantly, only if the two eigenvalues of transition matrix $\mathbf{M}$ are smaller than one.

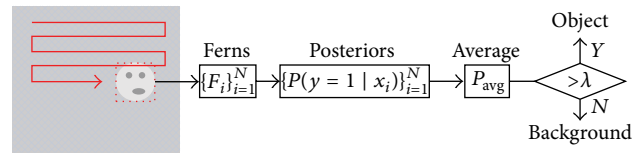


Figure 3: Object detection based on scanning window strategy and randomized forest classifier. The setting of the detector is as follows: 10,000 windows are scanned, 10 ferns per window.

3.3. Object Detecting. In previous subsections, we illustrate the weighted P-N learning method. In this subsection, a classifier will be developed to detect the object. Scanning window strategy is utilized to detect the object in [17]. Similarly, we use this method to detect the object.

In this paper, the randomized forest classifier [28] is adopted. For each input subwindow, classifier consists of N ferns. Each fern i computes the input patch resulting in feature vector x_i, which is used to obtain posterior probability $P(y = 1 \mid x_i)$. The following formulation is defined to discriminate input patch:

$$P_{\text{avg}} > \lambda \quad \text{object},$$
$$P_{\text{avg}} \leq \lambda \quad \text{background}, \quad (12)$$

where $P_{\text{avg}} = \sum_{i=1}^N P(y = 1 \mid x_i)$ denotes the average of all posteriors and λ is the threshold which is set to 0.4 in all experiments. The detection process can be illustrated in Figure 3. Actually, $W2_j^+ = P_{\text{avg}}$ and $W2_j^- = 1 - P_{\text{avg}}$. Feature vector is represented by 2-bit Binary Patterns [25] because of their invariance to illumination and efficient multiscale implementation using integral image. In fact, the posteriors $P(y = 1 \mid x_i)$ represent the parameter θ of the classifier and are estimated incrementally through the entire learning process. Each leafnode of fern records the number of positive p and negative n samples changed into it during iteration. The posteriors are then estimated by the following formulation:

$$P(y = 1 \mid x_i) = \frac{p}{p+n} \quad \text{leaf is not empty},$$
$$P(y = 1 \mid x_i) = 0 \quad \text{leaf is empty}. \quad (13)$$

The classifier is initialized in first frame, and posteriors are initialized to zero and renewed by 500 positive samples produced by affine warping of the selected patch [1]. The classifier is then evaluated on all the patches. In this paper, detections far from the selected patch represent the negative samples and update the posteriors.

4. Tracking

In this paper, the object is represented by independent local feature blocks. The tracking task is then transformed into tracking each local feature block. We train classifier for each LFB with online weighted P-N learning, respectively, and then track each LFB independently within the framework of LK optical flow [21]. During tracking procedure, a real-time

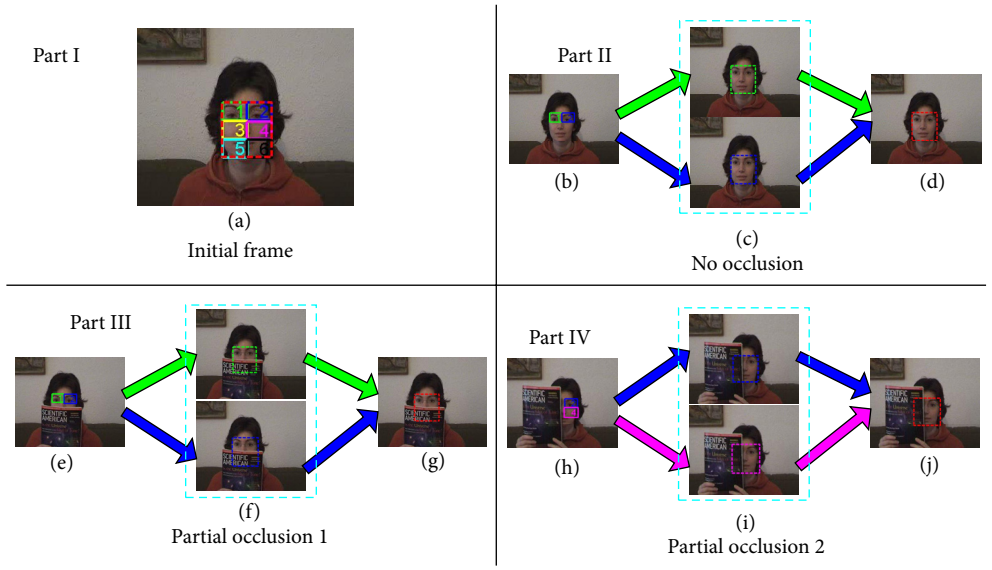


Figure 4: Illustration of tracking process. Part (I): segmentation of object in initial frame (first frame). Object is divided into six blocks. Part (II): tracking object without occlusion. Green block and blue block are selected as LFBs in image (b). Green and blue dotted line rectangles in image (c) are the tracking results of LFBs, and the red dotted line rectangle in image (d) represents the location of object estimated by the tracking results of LFBs. Part (III): tracking object with occlusion yet valid LFBs. Essentially, this process is similar to part (II). Occlusion does not affect the tracking. Part (IV): tracking object with occlusion with invalid LFB. In this part, green LFB is occluded and therefore invalid. We replace it with another new valid LFB, namely, the pink LFB, as shown in image (h). The object is then located in image (j) by tracking results of blue LFB and pink LFB in image (i).

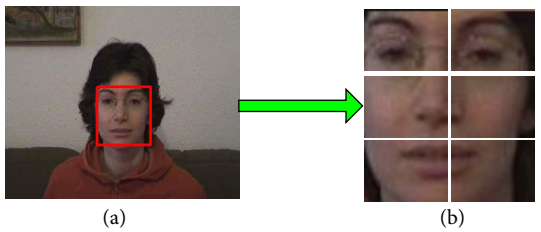


Figure 5: Uniform segmentation of object. Image (a) is object and image (b) is fragments.

valid detection algorithm is utilized for each LFB. If certain LFB is invalid, it will be replaced with an unused block, which makes our tracker robust. Figure 4 illustrates the principle of tracking.

4.1. Set of Local Feature Blocks. Object is represented by LFBs, and thereby, object needs to be segmented into fragments. For simplicity, uniform segmentation is adopted in this paper as shown in Figure 5.

After segmentation, we select part blocks as LFBs. Assume object is divided into K blocks; then we can obtain a candidate set of LFB set $CB = \{b_1, b_2, \ldots, b_K\}$ with K candidate local feature blocks. For candidate LFB b_i, we compute its 2-bit Binary Patterns feature vector x_i. Then, scanning window method is used to compute the similar likelihood between feature vectors of input patch f_i and b_i, and the similarity is represented as L_{ij}. $ML_i = \max(L_{ij})$ represents the highest similarity between b_i and all input patches. Finally, local feature block set $SB = \{sb_1, sb_2, \ldots, sb_M\}$ consists of M ($M < K$) candidate local feature blocks, with smaller ML_i.

4.2. Representations. In frame t, $B^t = (C^t, R^t, W^t, H^t)$ represents the object, where C^t and R^t are coordinates of center position and W^t and H^t are the sizes of object. $sb_k^t = (c_k^t, r_k^t, w_k^t, h_k^t)$ is the kth LFB, where c_k^t and r_k^t are coordinates of center location and w_k^t and h_k^t are the sizes of local feature block. $z_k^t = (oc_k^t, or_k^t, rw_k^t, rh_k^t)$ represents the offset of the kth LFB relative to object, where oc_k^t and or_k^t are the offset of center coordinates and rw_k^t and rh_k^t are the rations of sizes between object and the kth LFB. z_k^t can be determined by the following formulation:

$$\begin{aligned} oc_k^t &= C^t - c_k^t, \\ or_k^t &= R^t - r_k^t, \\ rw_k^t &= \frac{W^t}{w_k^t}, \\ rh_k^t &= \frac{H^t}{h_k^t}. \end{aligned} \tag{14}$$

4.3. Object Tracking. The tracked target is determined in initial frame (first frame) and segmented into fragments according to Section 4.1. Then, we select local feature blocks and compute B^1, sb_k^1, and z_k^1.

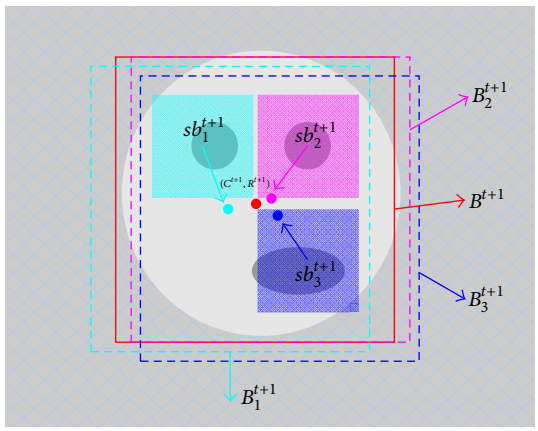


FIGURE 6: The process of locating object. The green, pink, and blue patches are the local feature blocks sb_1^{t+1}, sb_2^{t+1}, and sb_3^{t+1}, and the green, blue, and pink dotted line rectangles are the corresponding tracking results B_1^{t+1}, B_2^{t+1}, and B_3^{t+1}. The red solid line rectangle represents the final tracking result B^{t+1}, which is located by tracking results of local feature blocks.

Assume current frame is t. Each LFB is corresponded with a classifier via weighted P-N learning, and then we track each LFB. For the kth LFB, $sb_k^{t+1} = (c_k^{t+1}, r_k^{t+1}, w_k^{t+1}, h_k^{t+1})$ is used to represent its tracking result in frame $t+1$. By combining tracking results of each LFB and its offset z_k^t, we can obtain the corresponding object $B_k^{t+1} = (C_k^{t+1}, R_k^{t+1}, W_k^{t+1}, H_k^{t+1})$ via formulation the following formulation:

$$\begin{aligned} C_k^{t+1} &= oc_k^t + c_k^{t+1}, \\ R_k^{t+1} &= or_k^t + r_k^{t+1}, \\ W_k^{t+1} &= rw_k^t \cdot w_k^{t+1}, \\ H_k^{t+1} &= rh_k^t \cdot h_k^{t+1}. \end{aligned} \tag{15}$$

Each LFB determines a related candidate object. Object finally can be located by the following formulation:

$$B^{t+1} = \frac{1}{M} \cdot \sum_{i=1}^{M} B_i^{t+1}, \tag{16}$$

where M is the number of LFBs. The entire process can be explained by Figure 6. An adjustment is needed for offset of each LFB relative to object. We divide new object region B^{t+1} into fragments and compute the new offset of each LFB with prior LFBs based on formulation (14). After this, classifier of each LFB needs to be retrained via weighted P-N learning.

Representation by LFBs has significant advantages. Firstly, compared with entire object, local feature block is more prone to recognition in background, and this guarantees the accuracy and stability of tracking. Besides, object is located by averaging all the tracking results of local feature blocks, which decreases tracking errors by counteracting positive and negative errors; therefore, the robustness of proposed algorithm is improved.

4.4. Updating Set of Local Feature Blocks. During the tracking process, update is essentially adaptive to complex environment variation. In this paper, the learning procedure is online, and then the main problem is how to handle the situation of local feature blocks being invalid. A strategy of replacing is adopted to solve this problem. During the tracking procedure, we make a real-time valid detection (in Section 3.3) for all local feature blocks. If $P_{\text{avg}} \leq \lambda$, local feature block is invalid. When certain local feature block is invalid, it will be replaced with an appropriate block, which is selected from the outside of the LFB set.

Let UB be the unused block set, and $UB = \{ub_1, ub_2, \ldots, ub_P\}$, $P = K - M, UB \bigcup SB = CB$. In current frame t, LFB sb_k^t from SB is invalid and needs to be replaced. We first segment object into blocks and obtain UB. For block j from UB, we compute similar likelihood $l_j^t = \text{sim}(x_j^t, x_j^q)$, where sim is function of computing similar likelihood, x_j^t is feature vector of block j in frame t, and x_j^q is feature vector of block j before it is used the last time in frame q (if block j is never used, q equals one). ub_τ is used to replace sb_k^t via the following formulation:

$$ub_\tau = \arg\max l_j^t \left(ub_j\right). \tag{17}$$

The whole update process can be illustrated as shown in Figure 7.

So far, we have introduced the overall procedure of the proposed tracking algorithm as shown in Algorithm 1.

5. Experimental Results

In order to evaluate the performance of our tracking algorithm, we test our tracker on thirteen challenging image sequences. These sequences cover most challenging situations in visual tracking as shown in Table 1. For comparison, we run six state-of-the-art tracking algorithms with the same initial position of object. These algorithms are ℓ_1 tracking [2], FG tracking [8], IVT tracking [3], MIL tracking [5], TLD tracking [26], and CT tracking [6] approaches. Some representative results are shown in this section.

5.1. Quantitative Comparison. We evaluate the above-mentioned trackers via overlapping rate [29] as well as center location error, and the comparing results are shown in Tables 2 and 3.

Figure 8 shows the center location error of utilized tracker on thirteen test sequences. Overall, the tracker proposed in this paper outperforms the state-of-the-art algorithms.

5.2. Qualitative Comparison

Heavy Occlusion. Occlusion is one of the most common yet crucial issues in visual tracking. We test four image sequences (Woman, Subway, PersonFloor, and Occlusion1) characterized in severe occlusion or long-time partial occlusion. Figure 9(a) demonstrates the robustness performance of proposed tracking method in handling occlusion. Object is represented by local feature blocks in proposed algorithm.

Initialization:

(1) Segment object into N blocks and obtain $CB = \{cb_i\}_{i=1}^{N}$;
(2) Select the set of LFBs $SB = \{sb_k\}_{k=1}^{M} (M < N)$;
(3) Compute the set of offset in first frame $\{z_k^1\}_{k=1}^{M}$;
(4) Generate positive samples $(\mathcal{X}_k^+, \mathcal{Y}_k^+)$ and negative samples $(\mathcal{X}_k^-, \mathcal{Y}_k^-)$ for the k^{th} LFB;
(5) Train classifier f_k for the k^{th} LFB with data (X, Y) via weighted P-N learning, where $X = \{\mathcal{X}_k^+, \mathcal{X}_k^-\}$, and $Y = \{\mathcal{Y}_k^+, \mathcal{Y}_k^-\}$;

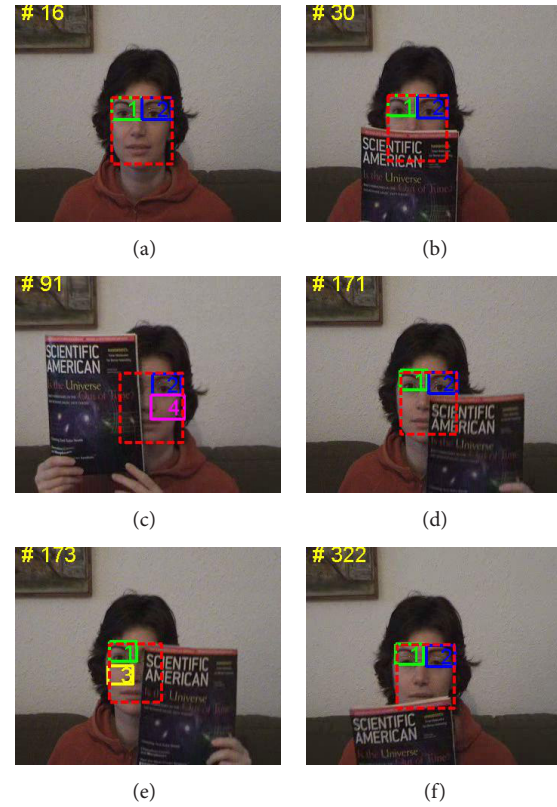
Object Tracking:

(6) **for** $t = 2$ to the end of the sequence **do**
(7) **for** $k = 1$ to M **do**
(8) Estimate sb_k^t via detecting k^{th} LFB with classifier in LK framework;
(9) Compute B_k^t via (15);
(10) Retrain classifier f_i;
(11) **end for**
(12) Estimate B^t via (16);
(13) Adjust the set of offset via (14);
(14) **for** $k = 1$ to M **do**
(15) Check each LFB with corresponding classifier
(16) **if** $P_{\text{avg}}^k < \lambda$ **do**
(17) Update $sb_k \leftarrow ub_\tau$ based on Section 4.4;
(18) Generate positive samples $(\mathcal{X}_k^+, \mathcal{Y}_k^+)$ and negative samples $(\mathcal{X}_k^-, \mathcal{Y}_k^-)$ for the k^{th} LFB;
(19) Train new classifier f_k for the k^{th} LFB
(20) **break**;
(21) **end if**
(22) **end for**
End

Algorithm 1: Tracking based on proposed method.

Table 1: The tracking sequences used in our experiments.

Sequence	Frames	Main challenges
Cup	303	Direction variation and scale variation
DavidIndoor	770	Illumination, scale variation, and pose change
DavidOutdoor	569	Illumination and scale variation
Deer	71	Fast motion, motion blur, and background clutter
Juice	404	Direction variation and scale variation
Jumping	313	Fast motion and motion blur
Lemming	900	Fast motion, motion blur, and occlusion
Occlusion1	415	Occlusion
OneLSR	559	Scale variation and occlusion
OSOW2cor	320	Scale variation and pose change
PersonFloor	387	Scale variation and occlusion
Subway	175	Occlusion and background clutter
Woman	551	Occlusion, scale variation, and pose change



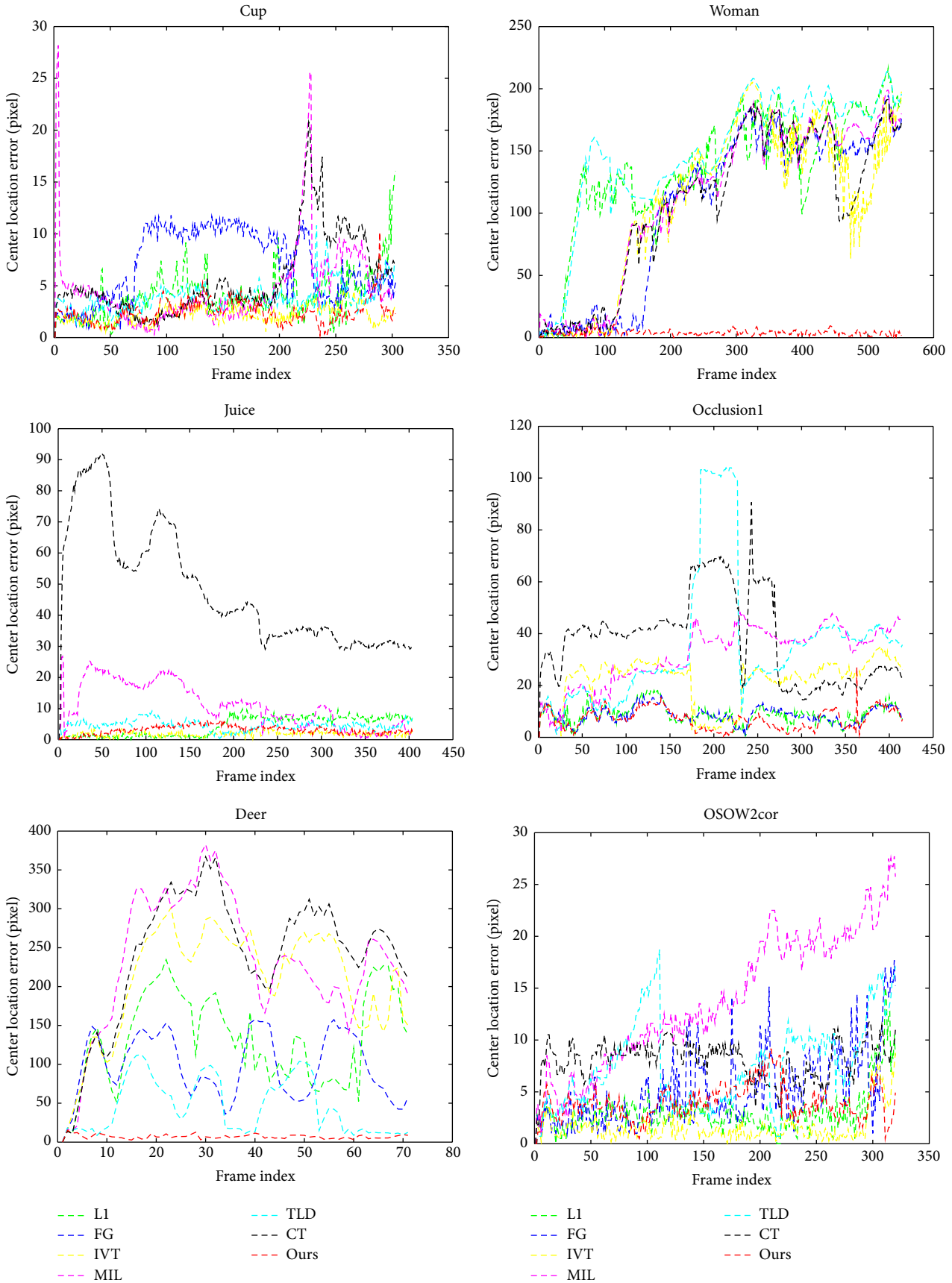
(a) (b) (c) (d) (e) (f)

Figure 7: The process of updating. The object is located by LFBs in (a) without occlusion. When occlusion occurs, yet each LFB is valid, the object can be tracked by LFBs exactly, as shown in (b). If occlusion happens and certain LFB is invalid, then it will be replaced and target still can be successfully located, as shown in (c), (d), (e), and (f).

Table 2: Center location errors (in pixels). The best result is shown in bold and the second best in italic fonts.

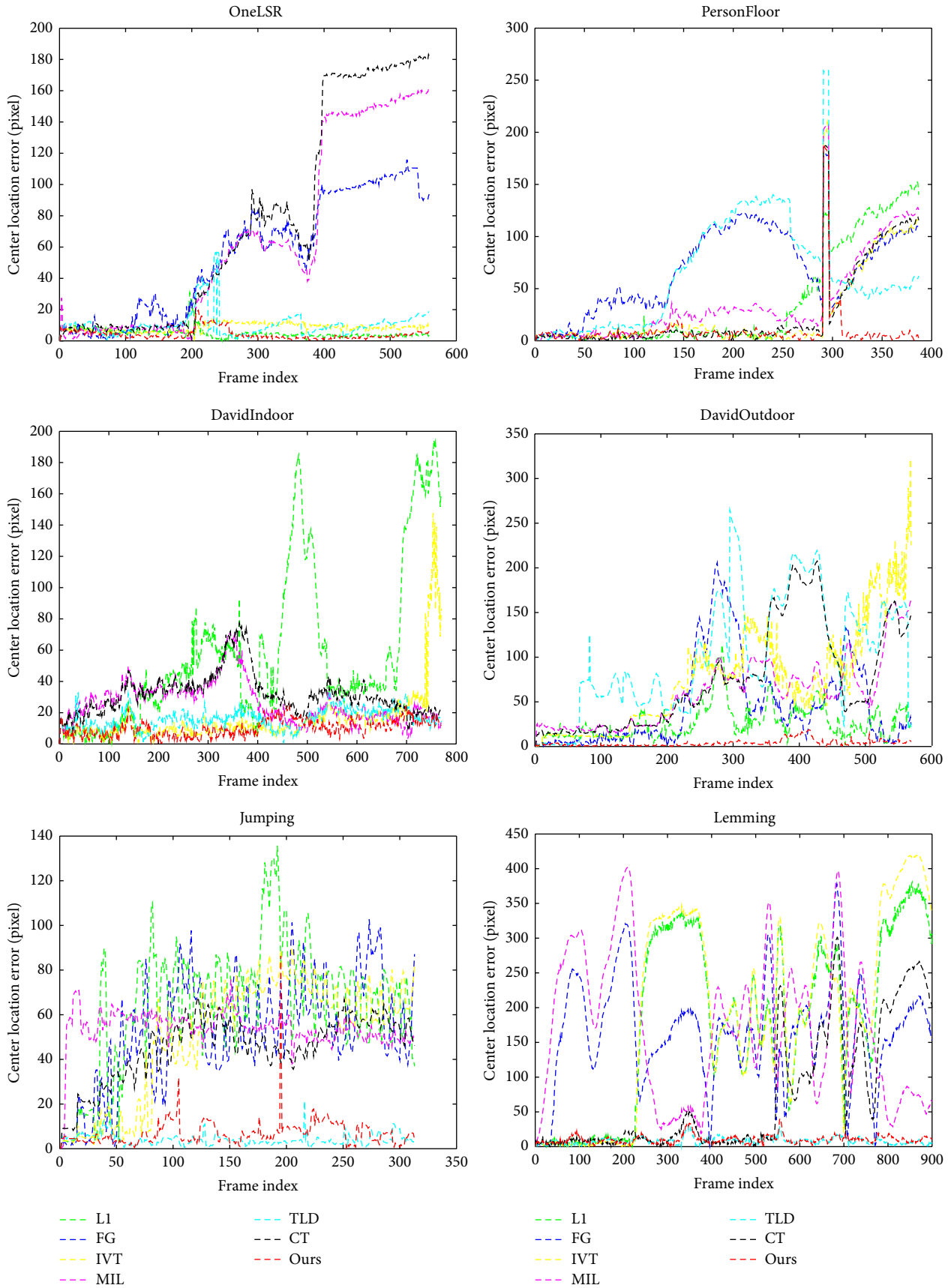
	ℓ_1	FG	IVT	MIL	TLD	CT	Ours
Woman	136.69	*106.21*	110.46	114.02	148.66	107.74	**3.77**
DavidIndoor	57.52	—	*15.78*	26.43	16.19	31.83	**10.34**
Cup	4.09	7.02	**2.23**	4.78	3.86	5.62	*2.47*
Juice	4.39	—	**1.80**	11.07	4.57	47.97	*3.07*
Deer	132.59	97.28	206.68	231.29	*45.83*	240.36	**7.02**
Lemming	181.13	164.87	192.12	168.87	**8.19**	73.28	*10.14*
OneLSR	*4.87*	54.70	8.43	65.48	10.50	76.47	**3.95**
PersonFloor	37.63	68.15	27.06	37.13	62.94	*26.89*	**9.52**
Subway	153.60	8.06	124.60	137.86	*5.26*	11.50	**3.19**
OSOW2cor	*3.06*	5.22	**1.62**	13.83	7.50	8.08	3.97
DavidOutdoor	*26.50*	45.87	75.86	59.77	102.20	73.72	**3.71**
Occlusion1	8.85	*8.60*	21.81	31.91	35.43	37.96	**7.11**
Jumping	64.23	52.34	50.18	54.03	**4.06**	45.11	*6.87*
Average	62.70	56.21	64.51	73.57	*35.01*	60.50	**5.78**

When occlusion happens, other tracking algorithms cannot track object well because they are prone to update background into object. However, our tracker can employ a new unused block to replace the invalid local feature block when occlusion occurs, which can make local feature block set effective to continue tracking.



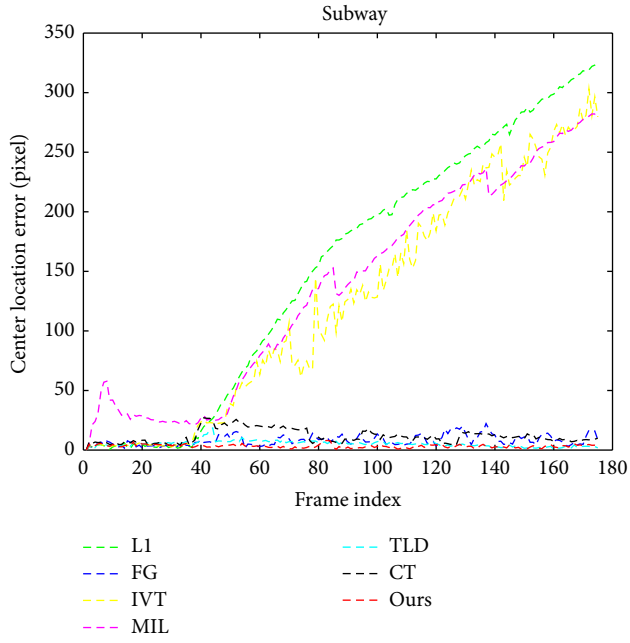
(a)

Figure 8: Continued.



(b)

Figure 8: Continued.



(c)

Figure 8: Quantitative evaluation of the trackers in terms of position errors (in pixels).

Table 3: Overlapping rate. Bold fonts indicate the best performance while the italic fonts indicate the second best ones.

	ℓ_1	FG	IVT	MIL	TLD	CT	Ours
Woman	0.06	*0.19*	0.14	0.15	0.04	0.15	**0.72**
DavidIndoor	0.21	—	0.3	0.45	**0.52**	0.39	*0.47*
Cup	0.64	0.67	0.61	**0.77**	0.63	*0.73*	0.62
Juice	*0.73*	—	**0.79**	0.64	0.66	0.07	0.69
Deer	0.04	0.07	0.03	0.04	*0.34*	0.04	**0.65**
Lemming	0.19	0.05	0.19	0.07	*0.64*	0.43	**0.7**
OneLSR	*0.54*	0.24	0.3	0.26	0.45	0.26	**0.56**
PersonFloor	0.47	0.25	0.53	0.47	0.21	*0.6*	**0.64**
Subway	0.15	0.61	0.17	0.09	**0.68**	0.5	*0.67*
OSOW2cor	0.59	0.43	**0.63**	0.39	0.5	0.4	*0.62*
DavidOutdoor	0.35	*0.42*	0.17	0.3	0.08	0.31	**0.62**
Occlusion1	**0.71**	**0.72**	0.5	0.55	0.48	0.41	0.66
Jumping	0.07	0.1	0.17	0.01	**0.78**	0.07	*0.75*
Average	0.37	0.36	0.35	0.32	*0.46*	0.33	**0.64**

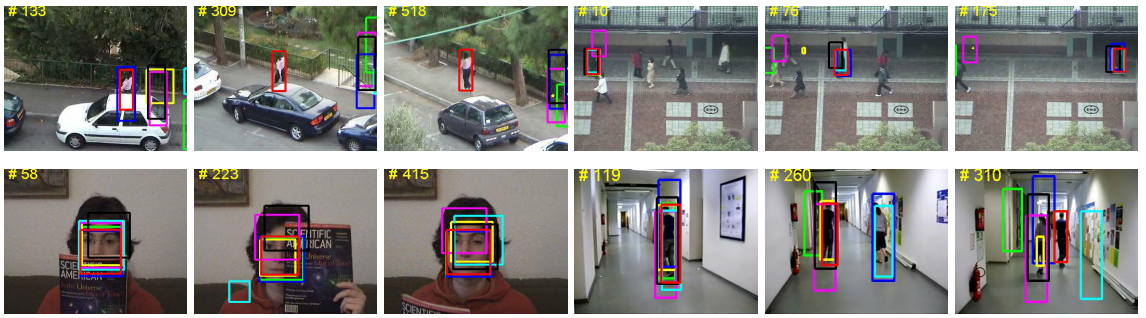
Scale Variation. Figure 9(b) presents the tracking results on four image sequences (OneLSR, OSOW2cor, Juice, and Cup) with large scale variation, even more with slight rotation. Our tracker can tail object throughout the whole sequences, which can be attributed to the discriminative classifier based on weighted P-N learning. We also observe that local feature blocks can better represent object, which makes the tracker focus on the stable part of the object.

Fast Motion and Motion Blur. Figure 9(c) demonstrates experimental results on three challenging sequences (Deer, Lemming, and Jumping). Because the target undergoes fast and abrupt motion, it is more prone to cause blur, which causes drifting problem. It is worth noticing that the suggested approach in this paper performs better than other algorithms. When motion blur occurs, our tracker can guarantee that the object's local feautres are still available. The advantages of using local feature blocks to represent object are shown incisively and vividly. By combining improved P-N learning and local feature blocks, we can obtain a discriminative classifier of stable object parts, which can locate the object. Then we track each local feature block, respectively, and determine the object based on tracking results of local feature blocks.

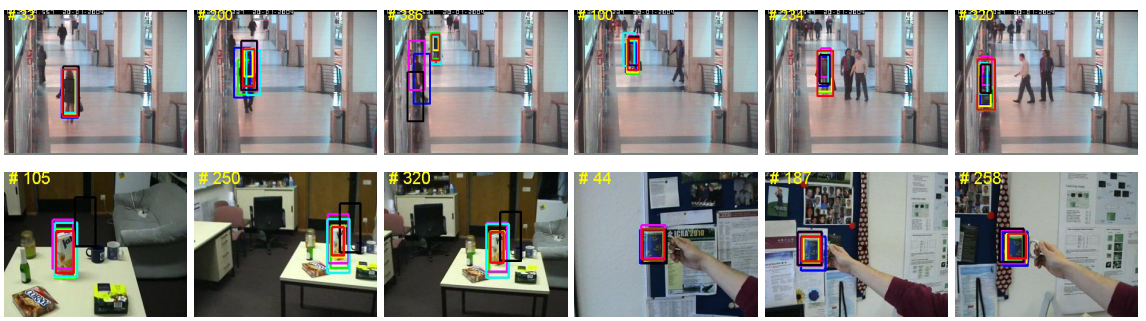
Illumination Variation. Illumination is a critical factor in visual tracking. Two typical image sequences (DavidIndoor and DavidOutdoor) are employed to test our tracker as shown in Figure 9(d). When illumination varies, some local regions of target are insensitive actually. Our tracker captures these insensitive regions to track local areas of object and further locate entire object via local tracking information.

6. Conclusions

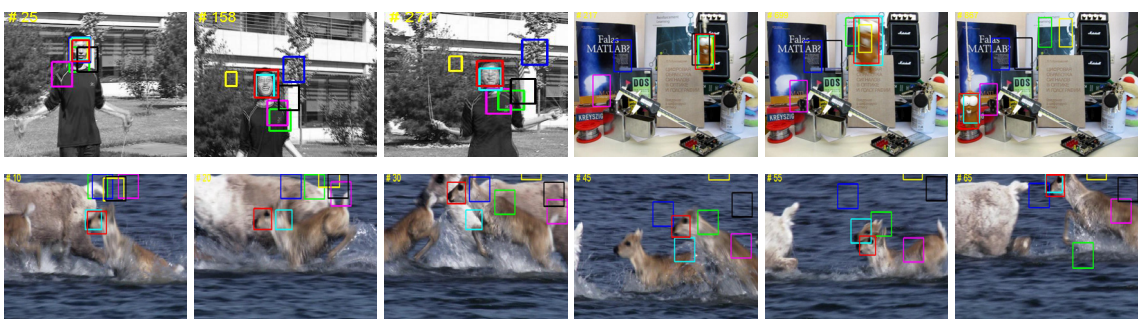
In this paper, we propose a part-based visual tracking algorithm with online weighted P-N learning. An online P-N learning is presented by assigning weight (property weight and classification weight) to each sample in training sample set, which can decrease classification errors and can improve the discriminative power of classifier. Firstly, the target is segmented into fragments, and parts of them are chosen to be local feature blocks to represent object. We



(a) *Woman*, *Subway*, *Occlusion1*, and *PersonFloor* with heavy occlusion



(b) *OneLSR*, *OSOW2cor*, *Juice*, and *Cup* with scale variation



(c) *Jumping*, *Lemming*, and *Deer* with fast motion and motion blur

(d) *DavidIndoor* and *DavidOutdoor* with illumination changes

Figure 9: Tracking results on various challenging sequences.

then train classifier for each LFB with weighted P-N learning, obtain the corresponding classifier, respectively, and track each LFB independently within the framework of LK optical flow. In addition, a substitute strategy is adopted to update dynamically the set of LFBs, which ensures robust tracking. Experimental results demonstrate that our algorithm outperforms state-of-the-art trackers. However, our algorithm fails to track object exactly in some scenes. If the tracked target is nonrigid and has an extremely heavy deformation or is fully occluded for long time, the performance of proposed tracker drops.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Jinhai Xiang is supported by the Fundamental Research Funds for the Central Universities under Grant no. 2013QC024. Jun Xu is supported by the National Natural Science Foundation of China under Grant no. 11204099 and self-determined research funds of CCNU from the colleges' basic research and operation of MOE.

References

[1] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: a benchmark," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, pp. 2411–2418, 2013.

[2] X. Mei and H. Ling, "Robust visual tracking using ℓ 1 minimization," in *Proceedings of the 12th IEEE Conference on Computer Vision and Pattern Recognition(CVPR '09)*, pp. 1436–1443, 2009.

[3] D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1–3, pp. 125–141, 2008.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[5] B. Babenko, S. Belongie, and M. Yang, "Visual tracking with online multiple instance learning," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR '09)*, pp. 983–990, Miami, Fla, USA, June 2009.

[6] K. Zhang, L. Zhang, and M. H. Yang, "Real-time compressive tracking," in *Proceedings of the European Conference on Computer Vision (ECCV '12)*, pp. 864–877, Firenze, Italy, October 2012.

[7] X. Jia, D. Wang, and H. Lu, "Fragment-based tracking using online multiple kernel learning," in *Proceedings of the 19th IEEE International Conference on Image Processing (ICIP '12)*, pp. 393–396, October 2012.

[8] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 798–805, June 2006.

[9] S. M. Shahed Nejhum, J. Ho, and M. H. Yang, "Visual tracking with histograms and articulating blocks," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, June 2008.

[10] S. Wang, H. Lu, F. Yang, and M. Yang, "Superpixel tracking," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 1323–1330, Barcelona, Spain, November 2011.

[11] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel, "Part-based visual tracking with online latent structural learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, pp. 2363–2370, Portland, Ore, USA, 2013.

[12] Q. Yu, B. T. Dinh, and G. Medioni, "Online tracking and reacquisition using co-trained generative and discriminative trackers," in *Proceedings of European Conference on Computer Vision (ECCV '08)*, pp. 678–691, 2008.

[13] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 2042–2049, June 2012.

[14] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 1269–1276, San Francisco, Calif, USA, June 2010.

[15] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 1830–1837, Providence, RI, USA, June 2012.

[16] B. Liu, J. Huang, L. Yang, and C. Kulikowsk, "Robust tracking using local sparse appearance model and K-selection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pp. 1313–1320, June 2011.

[17] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N learning: bootstrapping binary classifiers by structural constraints," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 49–56, June 2010.

[18] H. Grabner, C. Leistner, and H. Bischof, "Semisupervised online boosting for robust tracking," in *Proceedings of European Conference on Computer Vision (ECCV '08)*, pp. 234–247, 2008.

[19] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, "Tracking the invisible: Learning where the object might be," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 1285–1292, June 2010.

[20] K. Zhang and H. Song, "Real-time visual tracking via online weighted multiple instance learning," *Pattern Recognition*, vol. 46, no. 1, pp. 397–411, 2013.

[21] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, vol. 2, pp. 674–679, Vancouver, Canada, 1981.

[22] F. Wang, S. Yu, and J. Yang, "A novel fragments-based tracking algorithm using mean shift," in *Proceedings of 10th International Conference on Control, Automation, Robotics and Vision (ICARCV '08)*, pp. 694–698, December 2008.

[23] F. Tang, S. Brennan, Q. Zhao, and H. Tao, "Co-tracking using semi-supervised support vector machines," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, pp. 1–8, October 2007.

[24] Z. Kalal, K. Mikolajczyk, and J. Matas, "Face-TLD: tracking-learning-detection applied to faces," in *Proceedings of the 17th IEEE International Conference on Image Processing (ICIP '10)*, pp. 3789–3792, Hong Kong, September 2010.

[25] Z. Kalal, J. Matas, and K. Mikolajczyk, "Online learning of robust object detectors during unstable tracking," in *Proceedings of the IEEE 12th International Conference on Computer Vision Workshops (ICCV '09)*, pp. 1417–1424, October 2009.

[26] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[27] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[28] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[29] M. Everingham, L. V. Gool, C. Williams et al., "Partbased visual trackin g with online latent structural learning," in *Proceedings of the PASCAL Visual Object Classes Challenge (VOC '10) Results*, 2010.