Learn++ for Robust Object Tracking

Feng Zheng[1]
cip12fz@sheffield.ac.uk
Ling Shao[1]
ling.shao@ieee.org
James Brownjohn[2]
J.Brownjohn@exeter.ac.uk
Vitomir Racic[3]
v.racic@sheffield.ac.uk

[1] Department of Electronic and Electrical Engineering,
The University of Sheffield, UK

[2] College of Engineering, Mathematics and Physical Sciences,
University of Exeter, UK

[3] Department of Civil and Structural Engineering,
The University of Sheffield, UK

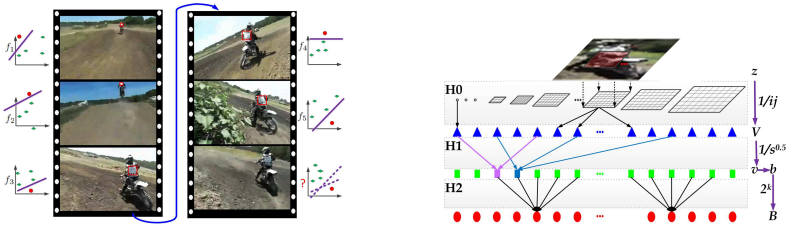
Abstract

In this paper, a Learn++ (LPP) tracker is proposed to efficiently select specific classifiers for robust and long-term object tracking. In contrast to previous online methods, LPP tracker dynamically maintains a set of basic classifiers which are trained sequentially without accessing original data but preserving the previously acquired knowledge. The different subsets of basic classifiers can be specified to solve different sub-problems occurred in a non-stationary environment. Thus, an optimal classifier can be approximated in an active subspace spanned by selected adaptive basic classifiers. As a result, LPP tracker can address the "concept drift", by automatically adjusting the active subset and searching the optimal classifier in an active subspace spanned by the subset according to the distribution of the samples and recent performance. Experimental results show that LPP tracker yields state-of-the-art performance under various challenging environmental conditions and, especially, can overcome several challenges simultaneously.

1 Introduction

Tracking-by-detection approaches [3, 4, 7, 19, 20], which treat the tracking problem as a classification task, have recently been proposed to overcome difficulties in the non-stationary environment (NSE) [22]. To improve the flexibility of the model, strategies for feature and sample selection are used in a large number of methods. Two classical discriminative feature selection methods were proposed in [7] and [19]. Semi-supervised learning [9] and multiple instance learning [4] were adopted for sample selection. AdaBoost [3] was used for both feature and sample selection. Compressive sensing [24] and sparse representation [16] methods are also explored to build appearance models for object tracking. Moreover, to avoid wrong updates, multi-models [5, 8, 14, 20, 23] which have more diversities were employed for visual tracking. Large scale experiments with various evaluation criteria to gauge the state-of-the-art are given in [21].

Most machine learning algorithms can learn from data that are assumed to be drawn from a fixed but unknown distribution. However, this assumption cannot be valid in case of



(a) (b)

Figure 1: (a) Assuming that the best classifiers for the previous frames are available, which classifiers should be used in the current frame (bottom right)? f_2, f_5 or their combination? Also, when the target moves out of view then comes back, which classifiers are the best to be used? This paper tries to solve these problems in object tracking. (b) Framework of SR.

tracking problem. Traditional machine learning methods applied to the tracking problem will fail when there is a "concept drift" in the NSE. That is because the function learnt on a fixed sample set previously collected may not reflect the current state of nature due to a change in the underlying environment [12]. In object tracking, the distribution of samples changes a lot due to the deformation of the object and the change of the background. Especially during the transition between different difficulties (sub-problems), such as from occlusion to varying viewpoints, the samples in the two different situations differ significantly. Thus, the separability of features and classifiers used in previous frames will decrease in the new situation. If x are the samples and $y \in \{1,-1\}$ are classes, then the "concept drift" can be defined as any scenarios where the posterior probability changes over time:

$$KL(p^{t+n}(y|x), p^{t}(y|x)) < \tau_d \tag{1}$$

where t is the time, n is the time step of drift, τ_d is a small value and the Kullback-Leibler (KL) divergence describes the dissimilarity of the two distributions.

Learn++ [18] is a new type of machine learning method to learn additional information from new data without retrieving the original samples. In [12], Learn++ is used to solve the problems in NSE, however, their method does not have a mechanism to make the classifier members learn from the new samples, exit from the active subset and revive from the ensemble, which are important for both the speed and the robustness of the model. In this paper, by enabling and designing these critical and flexible functions, we propose a fundamentally new Learn++ method for robust and long-term object tracking, named as LPP tracker. LPP tracker dynamically maintains a set of basic classifiers $f_i \in \Omega_e^t$ which are trained sequentially without accessing original data but preserving the previously acquired knowledge. The "concept drift" problems can be solved by adaptively selecting the most suitable classifiers named as active subset $\Omega_a^t \subset \Omega_e^t$ as shown in Fig. 1(a). These basic classifiers are independent from each other and used to address different sub-problems. For each frame, the democratic mechanism can be adopted, where all classifiers should compete with each other to be added into an active subset to suit the present environment. Next, the optimal classifier $\mathbf{f}^t$ in the present environment can be fast searched in a function space linearly spanned by these basic classifiers in the active subset.

As far as we know, LPP tracker is the first tracking method that designs an explicit model for each sub-problem and the models can be automatically altered according to the environment. As a result, LPP tracker can address various "concept drift" problems appeared in one video simultaneously. The rest of the paper is organized as follows. A fast and brief descriptor for image patches is introduced in Section 2. Section 3 details the proposed

Learn++ based method for visual tracking. Experimental results are reported and analyzed in Section 4. Finally, conclusions are drawn in Section 5.

2 Structural Representation for Image Patches

A fast structural representation (SR) is introduced to represent an image patch as shown in Fig. 1(b). SR has a three-level hierarchy: H0-virtual stage of filtering, H1-stage of random projection, and H2-stage of encoding. The advantage of SR is that the optimal projections and filters can be decided using selecting the third-level nodes by our proposed classifiers.

H0: filtering. Given a patch $z \in R^{I\times J}$, a set of rectangular smoothing filters $\{h_{i\times j} \in R^{i\times j}, 1 \le i \le I, 1 \le j \le J\}$ are defined, for which all entries of each filter $h_{i\times j}$ equal $1/(i\times j)$. In total, there are $I \times J$ filters, each of which is convolved with the entire patch and produces $I \times J$ values. So, the dimension $n_V = (IJ)^2$ of the original feature $V \in R^{n_V}$ is very high and much information is redundant.

H1: random projection. Next, a sparse random matrix is used for dimensionality reduction, which is defined as: $P(i,j) = 1$ with the probability $1/2s$, $P(i,j) = -1$ with the probability $1/2s$ and $P(i,j) = 0$ with the probability $1 - 1/s$, where p is the probability. In [1], Achlioptas pointed out that this matrix with $s = 2$ or 3 satisfies the Johnson-Lindenstauss lemma. Compressive sensing theory ensures that the extracted features preserve almost all the information of the original image patch. In this paper, we set $s = n_V/4$. Thus, the value $v_k \in R$ projected by each row of the random matrix is: $v_k = P(k,\cdot)V$. For patches with a different size $z^* \in R^{I^*\times J^*}$, the number of rectangular features will be different. In fact, we need not to resize the patch. Applying a scale $IJ/(I^*J^*)$ to the locations of elements in V^* will be feasible to realize scale invariance. For each value v_k, its mean μ_k and variance σ_k of positive samples will be computed when its corresponding classifier is trained. Thus, for each projected value v_k, a binary feature can be defined as: $b_k = \Gamma\lfloor v_k \in [\mu_k - \sigma_k, \mu_k + \sigma_k]\rfloor$, where $\Gamma\lfloor\rfloor$ is the indicative function.

H2: encoding. The third level is constructed similarly as Fern [17], in which a feature was calculated by comparing two randomly selected pixels in a patch. However, directly comparing two pixels is very sensitive to noise, especially when the two pixels are located around an edge. Normally, to eliminate this drawback, filters will be used firstly. Instead of comparing the pixels, the binary feature b_k can be considered as basic cues. Each Fern consists of a set of binaries, and in this paper, the size of the set n_b is set to 7. Thus, the node of the deepest level in the hierarchy is defined as: $B = \sum_{k=0}^{n_b} 2^k b_k$. Moreover, if the size of a filter is fixed, the number of nonzero entries of random projection is set to two, and the two weights are also opposite numbers, then SR will become the classical framework of Fern.

3 Learn++ for Solving the Problem of "Concept Drift"

The motion model $p(a^t|a^{t-1})$, i.e., Optical Flow (OP) [15], reflects the motion characteristic of the target by predicting the state a^t based on the previous state a^{t-1}. If $p(a^t|a^{t-1}) > \tau_1$, then a^t is a valid result no matter whether "concept drift" occurs or not. The P-N constraints [11], $p(a^t|f_i)$, were proposed to estimate the confidence of the classifier f_i. If $p(a^t|f_i) > \tau_2(i)$ where $f_i \in \Omega_a^t$, the outcome of classifier f_i is validated. This triggers the application of the P-N constraints that exploit the structure of the data. From the manifold

perspective, P-N constraints maintain a purified sub-manifold for positive and negative samples. If $p(a^t|\Omega_a^t) > \tau_2$, where $p(a^t|\Omega_a^t) = \max_i p(a^t|f_i)$, there is no occurrence of drifting.

3.1 Objective function

Assume $\Omega_c^t \bigcup \Omega_a^t = \Omega_e^t$, $n_e^t = |\Omega_e^t|$ and $n_a^t = |\Omega_a^t|$, where $|\cdot|$ denotes the number of members of the set. W_i^t denotes the historical weights of all existing $f_i \in \Omega_e^t$. In frame t, x_l^t and y_l^t denote the structural representation and the label of image patch z_l^t, respectively. Each entry $x_l^t(i)$ contains the n_B number of nodes $\{B_{i,j} : j = 1, \cdots, n_B\}$, for the classifier f_i. Also, X^t is the set of collected samples and $n_X^t = |X^t|$. The distribution of samples D^t will be calculated according to the results of old classifiers and used to describe the importance of samples. For simplicity, we define $f_i(x_l^t) = f_i(x_l^t(i))$, $w^t = (w_1^t, \cdots, w_{n_a}^t)$ and $\Omega_e^t = (f_1, \cdots, f_{n_a})^T$. Our goal is to find an optimal classifier $\mathbf{f}^t$ with most discriminative features in the function space $\mathcal{H}^t$ linearly spanned by a set of classifiers Ω_e^t which are trained in previous frames, where $\mathcal{H}^t = \{h^t : h^t = w^t \Omega_e^t\}$. Moreover, to improve the efficiency of the system, the weights w^t are required to be sparse so that most basic classifiers are not used in the current frame. Thus, in theory, the objective function is defined as:

$$\mathbf{w}^t = \arg\min_{w^t} \sum_l L(h^t(x_l^t), y_l^t) + \lambda \left|\left|w^t\right|\right|_0 \tag{2}$$

where L and λ are the loss function and regularization parameter, respectively. Therefore, we obtain the hypothesis as:

$$\mathbf{f}^t = \mathbf{w}^t \Omega_e^t \tag{3}$$

The optimal classifier $\mathbf{f}^t$ can be used to detect the object in the current frame (new environment). The final classification for each image patch x_i^t is achieved as: $y_l^t = sign(\mathbf{f}^t(x_l^t))$.

Eqn. 2 cannot be optimized directly, due to that the true label y_l^t of image patch x_l^t is unknown. However, based on the assumption of the "concept drift" (Eqn. 1), we can approximate to the optimal solution using the classifiers that have yielded good performance in recent n frames or in the same situations by Learn++. In the following, how to train basic classifiers f_i and how to approximate the optimal classifier $\mathbf{f}^t$ will be introduced.

3.2 Basic classifier

The Naïve Bayesian is used as the basic classifier in our proposed system. Thus, f_i can be defined by posterior probabilities by combining n_B nodes (assuming an uniform prior $p(y)$):

$$f_i(x_l) = \arg\max_y p(y|x_l(i)) \tag{4}$$

where $p(y|x_l(i)) \propto \prod_j^{n_B} p(x_l(i,j)|y)$. Therefore, for each f_i, the posterior probabilities will be trained and updated to adapt to the changes of the environment and the object by calculating and updating the class conditional distribution $p^t(B_{i,j}|y)$ of each Fern.

Training. The parameters of SR for each classifier f_i will be generated randomly. Once generated, these parameters will be fixed during the whole lifespan of the classifier f_i. At frame t, based on a set X_1^t with all positive samples and 2000 negative samples in the set X^t and distribution D^t, we can define two quantities which are used to train or update classifiers: $N^t(y, B_{i,j}) = \sum_l D_l^t \Gamma\lfloor x_l^t(i,j) = B_{i,j} \rfloor \Gamma\lfloor y_l^t = y \rfloor$ and $N^t(y) = \sum_l D_l^t \Gamma\lfloor y_l^t = y \rfloor$, where $x_l^t \in X_1^t$. Other negative samples are used to evaluate the classifier. Therefore, $\tau_2(i) = max_{a_l^t} p(a^t|f_i)$, where a_l^t denotes the corresponding states of negative samples $x_l^t \in X^t / X_1^t$. Thus, the class conditional distributions for f_i are calculated by:

$$p^t(B_{i,j}|y) = \frac{1+N(y,B_{i,j})}{1+N(y)} \tag{5}$$

where $N(y,B_{i,j}) = N^t(y,B_{i,j})$ and $N(y) = N^t(y)$.

Learning. If f_i has been used in frame t. The set X^t can be used to update the class conditional distribution $p^t(B_{i,j}|y)$ so as to adapt to the changes by:

$$N(y,B_{i,j}) \Leftarrow N(y,B_{i,j}) + N^t(y,B_{i,j}); N(y) \Leftarrow N(y) + N^t(y) \tag{6}$$

By recalculating Eqn. 5, the updated distributions are obtained.

3.3 Tracking by detection

At the beginning $(t=0)$, random Ferns f_1 needs to be trained according to the selected target in the first frame and we can directly jump to the sample collection step. At frame $t(t>0)$, the following steps which are similar to most tracking-by-detection approaches are processed sequentially. First, by applying the sliding window method to the current frame, the classifier in the active subset is used to classify each patch of this frame. Second, the OP method is used to compare the two targets in the two successive frames. Third, the probabilities $p(a^t|\mathbf{a}^{t-1})$ and $p(a^t|\Omega_a^t)$ are calculated. Fourth, all states classified as positive samples by $\mathbf{f}^t$ will be fused and the optimal state $\mathbf{a}^t$ with the highest confidence in the current frame will be obtained. Finally, the classifiers will be updated according to the present performance. The entire procedure is organized as in Algorithm 1.

Algorithm 1 LPP tracker

Initialization Define a target in the first frame and build a classifier f_1.
Repeat $t = 1, \cdots$
(0) Capture a new frame. **If** no frame: **Exit**.
(1) Run each classifier $f_i \in \Omega_a^t$ of the active subset on the present frame.
(2) Combine the results $\mathbf{f}^t$ according to Eqn. 3 and obtain the best results: $\mathbf{a}^t$.
(3) **If** $\mathbf{a}^t$ is valid target: Compute the probabilities $p(a_t|\mathbf{a}_{t-1})$ and $p(a_t|f_i)$.
(4) **If** $p(\mathbf{a}^t|\mathbf{a}^{t-1}) > \tau_1$: Collect and weight samples X^t,
(5) **If** $p(\mathbf{a}^t|\Omega_a^t) > \tau_2$: Update the old classifiers f_i,
(6) **Else If** $p(\mathbf{a}^t|\Omega_c^t) > \tau_2$: Revive a classifier from Ω_c^t;
(7) **Else**: Train a new classifier $f_{n_a^t+1}$.
End
End
(8) Resampling and evaluate the classifiers.
Return Update the classifier $\mathbf{f}^{t+1}$ and set the state $\mathbf{a}^t$, **Go To** (0).

3.4 Collecting and weighting samples

If $p(\mathbf{a}^t|\mathbf{a}^{t-1}) > \tau_1$ is satisfied, it means that the tracked target is valid and can be used to update the set of classifiers. Otherwise, when no valid target is in the current frame, we can directly jump to the classifier sampling step.

Collecting. The sample set X^t is constructed as follows: If the overlap of $\mathbf{a}^t$ and a_l^t exceeds 0.5, the patch z_l^t of state a_l^t will be considered as the positive sample; otherwise if the overlap of $\mathbf{a}^t$ and a_l^t is lower than 0.2, it is considered as the negative sample. Also, according to the fused results $\mathbf{a}^t$, 400 positive samples will be generated by the affine warping of the selected patch $\mathbf{z}^t$ to increase the richness of positive samples.

Weighting. At the beginning $(t=0)$, the distribution of samples D^t used to train the first classifier is set to be equal to $1/n_X^t$. If $(t>0)$, the distribution of patches in the tth

frame will be computed. Firstly, the current ensemble $\mathbf{f}^t$ is evaluated on the new patches X^t: $E^t = \frac{1}{n_X^t}\sum_{l=1}^{n_X^t}\Gamma\lfloor sign(\mathbf{f}^t(x_l^t)) \neq y_l^t\rfloor$. Secondly, sample weights D_l^t of x_l^t are defined by:

$$D_l^t = \begin{cases} E^t, & sign(\mathbf{f}^t(x_l^t)) = y_l^t; \\ 1, & otherwise. \end{cases} \tag{7}$$

Finally, set $D_l^t \Leftarrow D_l^t / \sum_{l=1}^{n_X^t} D_l^t$. Normalizing the error weights by their sum then provides us the updated penalty distribution. Samples of the new environment x_l^t, which are not recognized by the existing knowledge base $\mathbf{f}^t$, are identified.

3.5 Sampling the classifiers

If the current active subset can deal with the changes, the optimal classifier in the next frame has the same basic classifiers. To increase the adaptivity, the new samples will be learnt by existing classifiers. For each $f_i \in \Omega_a^t$, if $p(\mathbf{a}^t|f_i) > \tau_2(i)$, then f_i will be updated by the samples X^t according to their distribution D_l^t following Eqn. 6. Otherwise, reviving the old classifiers or training a new classifier will be considered.

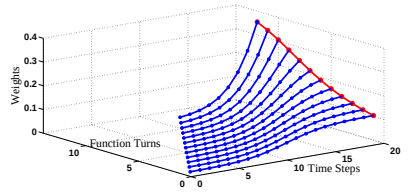
Reviving. If $p(\mathbf{a}^t|\Omega_a^t) < \tau_2$ and $p(\mathbf{a}^t|\mathbf{a}^{t-1}) > \tau_1$, due to the "concept drift", it means that the current ensemble cannot deal with the changes. So, a new set of basic classifiers need to be built. New classifiers will be added into the ensemble so that the optimal classifier will be searched in a new set of classifiers. Firstly, all existing classifiers $f_i \in \Omega_c^t$ will be used to check whether the current appearance can be recognized or not by old classifiers. If a similar "concept drift" has occurred before, an old classifier can be revived. This procedure is efficient to compute because no sliding window is needed. If $p(\mathbf{a}^t|\Omega_c^t) > \tau_2$, where $p(\mathbf{a}^t|\Omega_c^t) = \max_i p(\mathbf{a}^t|f_i)$, there exists one classifier f_i that can recognize the current state. Thus, this classifier f_i will be revived directly without adding a new one. Otherwise, a new classifier will be trained and added to the ensemble following Eqn. 5.

Resampling. No matter whether the valid target has been detected in the current frame or not, some classifiers killed before will be revived through the resampling procedure according to the historical weights W_i^t. This will increase the diversity and avoid the local optimal solution. The adaptive rejection sampling method [6] is employed to realize this step.

Evaluating. For finding the optimal classifier for the next frame, evaluating all classifiers $f_i \in \Omega_a^{t+1}$ on the new data X^t is necessary. Firstly, the error of each $f_i \in \Omega_a^{t+1}$ on weighting samples is defined as: $\varepsilon_i^t = \sum_{l=1}^{n_X^t} D_l^t \Gamma\lfloor f_i(x_l^t) \neq y_l^t\rfloor$. Thus, $\varepsilon_i^t \Leftarrow \varepsilon_i^t/(1-\varepsilon_i^t)$. ε_i^t can be considered as the performance of the function. If f_i contributes mostly to the error of the ensemble classifier $\mathbf{f}^t$, ε_i^t will be larger than others. Secondly, for incorporating the performance on recent frames, a sigmoidal error weight is defined as: $\gamma_i^t(m) = 1/(1+\exp(\lambda_1 m - \lambda_2))$, $\{m = 0, \cdots, n_e^t + \eta - i\}$, where λ_1, λ_2 are two parameters, η is the time step and i is the index of the function in the ensemble. Thus, the weights are normalized so that $\gamma_i^t(m) \Leftarrow \gamma_i^t(m)/\sum_m \gamma_i^t(m)$ (see Fig. 2(a)). Finally, the error of $f_i \in \Omega_a^{t+1}$ is weighted with respect to time so that recent competence (error rate) is considered more heavily for categorizing knowledge. The weighted errors are defined by:

$$\beta_i^t = \sum_{m=0}^{n_e^t+\eta-i} \gamma_i^t(n_e^t + \eta - i - m)\varepsilon_i^{t-m} \tag{8}$$

Thus, we calculate the classifier voting weights: $w_i^t = \log 1/\beta_i^t$ and normalize them: $\mathbf{w}_i^{t+1} \Leftarrow w_i^t/\sum_i w_i^t$. The instant voting weights can be used to update the historical weights according to $W_i^{t+1} \Leftarrow (1-\alpha)W_i^t + \alpha\mathbf{w}_i^t$, where α is the updating rate and is set to 0.05.



(a)

David: IV, SV, OCC, DEF, MB, IPR, OPR
CarScale: SV, OCC, FM, IPR, OPR
FaceOcc2: IV, OCC, IPR, OPR
Jogging: OCC, DEF, OPR
Jumping: MB, FM
Dudek: SV, OCC, DEF, FM, IPR, OPR, BC
Doll: IV, SV, OCC, IPR, OPR
Mhyang: IV, DEF, OPR, BC
Sylvester: IV, IPR, OPR
CarDark: IV, BC
Couple: SV, DEF, FM, OPR, BC
Football: OCC, IPR, OPR, BC
Football1: IPR, OPR, BC
Dog1: *SV, IPR, OPR*
FaceOcc1/ Fish: OCC/IV
Crossing: SV, DEF, FM, OPR, BC
Boy: SV, MB, FM, IPR, OPR
Singer1: IV, SV, OCC, OPR
Walking: SV, OCC, DEF
David2: IPR, OPR

(b)

Figure 2: (a) Signoidal weights used in Eqn. 8. λ_1, λ_2 and η are set to 0.5, 10 and 8, respectively. (b) The details of test videos. 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; BC-Background Clutters.

Challenges	LPP	Struck	VTS	IVT	VTD	MIL	OAB	Frag	CT	SemiT
IV	0.932	0.860	**0.957**	0.900	0.888	0.569	0.697	0.565	0.704	0.463
OPR	**0.858**	0.775	0.754	0.718	0.766	0.670	0.704	0.627	0.625	0.544
SV	**0.928**	0.816	0.763	0.779	0.771	0.769	0.793	0.609	0.805	0.449
OCC	**0.772**	0.659	0.723	0.749	0.733	0.583	0.677	0.675	0.608	0.519
DEF	**0.871**	0.682	0.595	0.674	0.6000	0.618	0.753	0.624	0.643	0.677
MB	**0.919**	0.776	0.726	0.513	0.710	0.847	0.487	0.549	0.614	0.293
FM	**0.875**	0.856	0.546	0.459	0.547	0.767	0.579	0.713	0.571	0.448
IPR	0.861	0.867	0.869	0.819	**0.885**	0.778	0.672	0.622	0.659	0.489
BC	0.882	**0.912**	0.725	0.769	0.709	0.714	0.697	0.661	0.594	0.609
Overall	**0.844**	0.817	0.736	0.734	0.720	0.669	0.664	0.658	0.605	0.552

Table 1: The precision rankings of 10 tracking methods on challenging sequences. Bold numbers denote the best precision scores.

Optimal approximation. To balance the increase of the diversity of the ensemble and efficiency of the model, the following conditions will be considered: (1) For any $f_i \in \Omega_a^{t+1}$ with $\mathbf{w}_i^{t+1} < \tau_3$, the classifiers will be killed and moved to Ω_c^{t+1}; (2) For any $f_i \in \Omega_c^{t+1}$ with $W_i^{t+1} < \tau_3$, the classifiers will be deleted for ever. Because the size of Ω_a^{t+1} is much smaller than Ω_e^{t+1}, the weights $\mathbf{w}^{t+1}$ are sparse. Therefore, the optimal approximation classifier used in the next frame will be defined by: $\mathbf{f}^{t+1} = \sum_{f_i \in \Omega_e^{t+1}} \mathbf{w}_i^{t+1} f_i$.

4 Experiments

In our experiments, τ_1, τ_2 and τ_3 are set to 0.75, 0.9 and 0.05, respectively. The greyscale images are taken as input in our experiments. LPP tracker will be compared with 9 state-of-the-art methods, including IVT [19], VTD [13], VTS [14], MIL [4], OAB [8], SemiT [9], Frag [2], Struck [10] and CT [24], most of which were recently proposed. The 21 videos summarized in Fig. 2(b) are used for testing. Our experiments follow the setting in [21] and the results of other methods come from this report as well. Each sequence is repeated 5 times with different random seeds by LPP tracker, and the median results are reported. More results are included as supplemental material[1]. To compare with various methods, two types of metric are used to evaluate the different methods. (1) Center location distance: following [21], if the distance between the center of the tracked patch and the center of ground truth

[1]More video results and code will be released on our website: https://sites.google.com/site/lpptracker/.

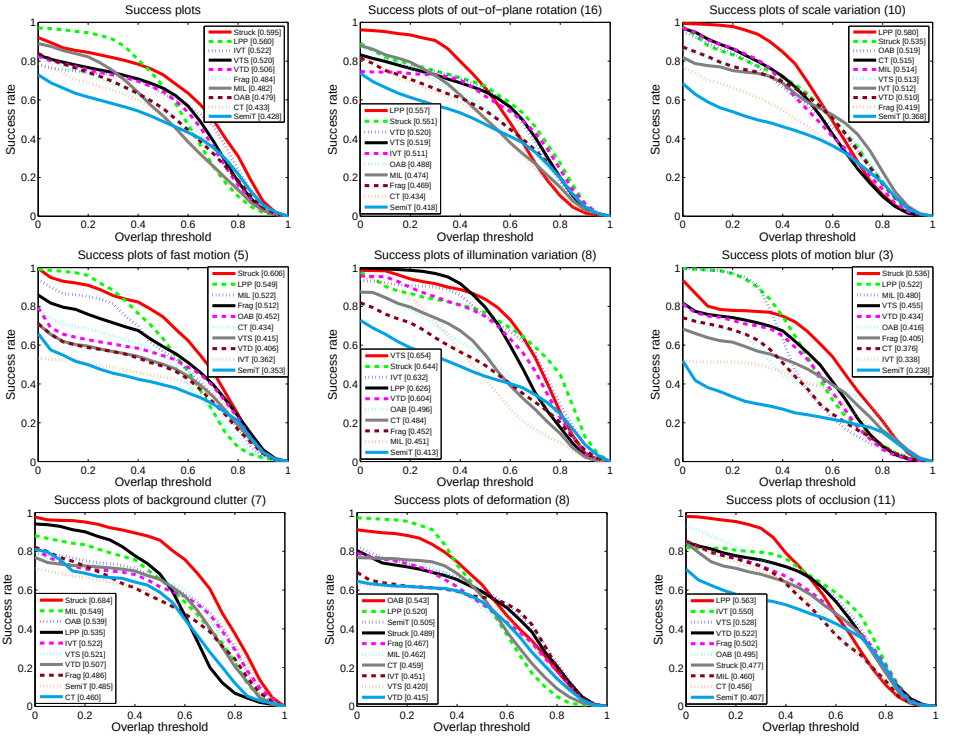


Figure 3: The success plots and AUC rankings of 10 tracking methods on challenging sequences.

is within 20 pixels, the estimated target is considered as correct. Thus, the precision can be defined as the proportion of the correctly tracked frames to the total number of frames. The precision rankings of the 10 methods on the 21 videos are given in Table 1. (2) Bounding box overlap: the success plot shows the ratios of successful frames at the thresholds varying from 0 to 1. The area under curve (AUC) [21] of each success plot is used to rank the tracking algorithms. Both success plots and AUC rankings are shown in Fig. 3 and some screenshots are shown in Fig. 4.

4.1 Comparison with state-of-the-art methods

Firstly, taking the sequence *singer1* (624×352) for example, CT, LPP tracker and Struck take the average time per frame of 17*ms*, 55*ms* and 209*ms* respectively on a Dell M4600 (Intel Core 2.8GHz and 8G RAM). Thus, the LPP tracker can address most real-world problems in real-time (more than 18 FPS). Secondly, from Table 1, LPP tracker can achieve the best performance among all the 10 methods on most of the challenges. On the one hand, for challenges of out-of-plane rotation, scale variation, occlusion and motion blur, LPP tracker has a great advantage over the other methods. On the other hand, for the other three challenges, LPP tracker is not the best one but the score of precision is close to the best one. The score differences to the best one are less than 0.03. The overall performance of LPP tracker on all sequences is 0.844 and higher than that of the second method Struck by 0.027. Thirdly, from top left of Fig. 3, we can see that LPP tracker is ranked as the second best on all the 21 test-

Figure 4: Screenshots of top 6 tracking methods (according to AUC rankings in the top left plot of Fig. 3) on challenging sequences. In total, 11843 frames are tested.

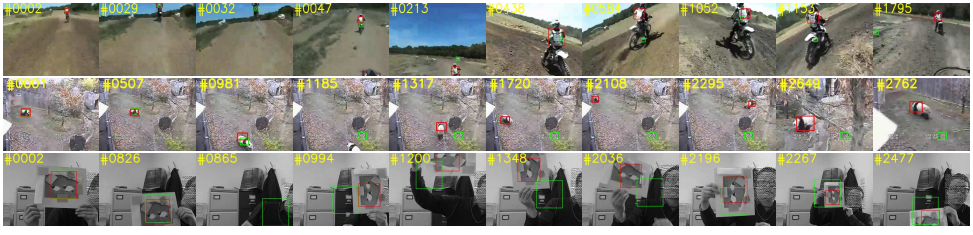


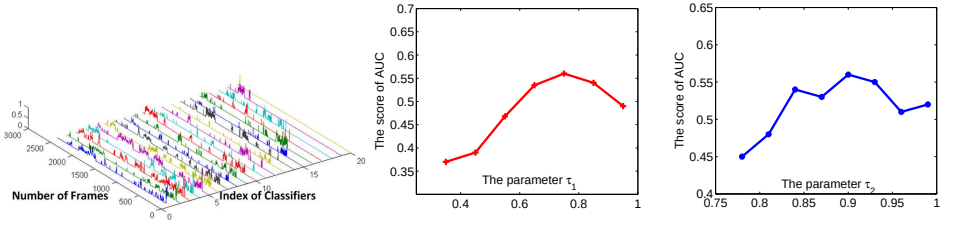
Figure 5: Comparsion of tracking results on three more challenging sequences between LPP tracker (Red) and Struck (Green). *motorcross* (top row), *panda* (middle row) and *sheep* (bottom row) have 1800, 3000 and 2532 frames, respectively.

ing videos and outperforms other methods by at least 0.38 except for Struck. It demonstrates that our proposed LPP tracker is relatively robust to various challenges. Also, LPP tracker works much better than all other methods on the challenges of occlusion, out-of-plane rotation and scale variation. Particularly for scale variation, LPP tracker outperforms the second best method by 0.55. In addition, on the challenges including deformation, motion blur and illumination variation, LPP tracker performs closely to the best method.

In total, LPP tracker gains six firsts, two seconds and one fourth by the precision ranking, and it gains three firsts, three seconds and two fourths by the AUC ranking. The differences between the two rankings are on deformation, motion blur and fast motion. That is because LPP tracker can build a new classifier for one part of the object when there are some large deformations in the remaining part in these challenges, where the object has been tracked by LPP tracker but the score of overlap is relatively low.

4.2 More analysis of the LPP tracker

There are two parameters of motion constraints τ_1 and τ_2 to guide the learning of LPP tracker. In this section, when we investigate one parameter, other parameters will be set to default (same values for all videos). In Fig. 6(b) and (c), the overall performance on all the videos vs. the different settings for the two parameters are given. We can see that the parameter τ_1 achieves the best performance around 0.75 while the parameter τ_2 achieves the best performance around 0.9. If the two parameters are set too small, the model will become more flexible but less stable. More erroneous information will be added into the model and the



(a) (b) (c)

Figure 6: (a) The weights for the optimal classifier $\mathbf{f}^t$. (b) The AUC performance vs. parameter τ_1. (c) The AUC performance vs. parameter τ_2.

performance will deteriorate. However, if the two parameters are set too large, the model cannot adapt to the new environment and the performance of the model will decrease as well. Moreover, the scores of AUC are relatively stable around the best values of the two parameters, which means they are not very sensitive.

To further demonstrate the capabilities of our system, we compare LPP tracker with Struck (the best method in [21]) on three more challenging sequences named *motorcross*, *panda* and *sheep*. There are several difficulties, which are normally not considered by other methods before: (1) the target makes a complete rotation; (2) the target moves out of view and gets back with a totally different appearance and location; (3) the video is very long and various challenges appear simultaneously. To some extent, the assumptions of smooth motion and smooth variation necessary for most methods are not valid anymore in such sequences. The three sequences with the above three difficulties will be good examples to test the flexibility and stability of a model. Firstly, Struck fails at frames 30, 1016 and 828 for sequences *motorcross*, *panda* and *sheep*, respectively, when the target starts to move out of view. However, LPP tracker can successfully reject the learning from wrong samples and keep its stability. Secondly, from Fig. 5, we can see that LPP tracker can tackle all these problems simultaneously because LPP tracker builds one classifier for each problem. Finally, Fig. 6(a) demonstrates the weights of classifiers on all the frames of sequence *sheep*. When no valid target is detected, LPP tracker will sample the classifiers according to their historical weights. Once the predefined target appears in view, LPP tracker will select the most effective classifier to track the target. From Fig. 6, we can see that the weights are very sparse and just a few members will be run for each frame.

5 Conclusion

In this paper, we have proposed a novel Learn++ tracker for visual tracking. By means of automatically adjusting the members of the active subset, LPP tracker achieves an optimal balance between flexibility and stability of the classifiers and between the efficiency and performance of the model. In future work, it is worth considering using other constraints to guide the sampling of classifiers. Moreover, for abrupt deformation of the target when typically $n < 5$, LPP tracker may refuse to add a new classifier to the ensemble. How to define an adaptive quantity to tackle such a situation is under investigation.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66:671–687, 2003.

[2] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proc. CVPR*, 2006.

[3] Shai Avidan. Ensemble tracking. *IEEE Tansactions on PAMI*, 29(2):261–271, 2007.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE Tansactions on PAMI*, 33(8):1619–1632, 2011.

[5] Qinxun Bai, Zheng Wu, Stan Sclaroff, Margrit Betke, and Camille Monnier. Randomized ensemble tracking. In *Proc. ICCV*, 2013.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, ISBN: 0-38731073-8, 2007.

[7] Robert T. Collins, Yanxi Liu, and Marius Leordeanu. Online selection of discriminative tracking features. *IEEE Tansactions on PAMI*, 27(10):1631–1643, 2005.

[8] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Proc. CVPR*, 2006.

[9] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Proc. ECCV*, 2008.

[10] Sam Hare, Amir Saffari, and Philip H. S. Torr. Struck: Structured output tracking with kernels. In *Proc. ICCV*, 2011.

[11] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *Proc. CVPR*, 2010.

[12] Matthew Karnick, Metin Ahiskali, Michael D. Muhlbaier, and Robi Polikar. Learning concept drift in nonstationary environments using an ensemble of classifiers based approach. In *Proc. IJCNN*, 2008.

[13] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *Proc. CVPR*, 2010.

[14] Junseok Kwon and Kyoung Mu Lee. Tracking by sampling trackers. In *Proc. ICCV*, 2011.

[15] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. IJCAI*, 1981.

[16] Xue Mei and Haibin Ling. Robust visual tracking using l1 minimization. In *Proc. ICCV*, 2009.

[17] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. CVPR*, 2007.

[18] Robi Polikar, Lalita Udpa, Satish S. Udpa, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Tansactions on System, Man, and Cybernetics-Part C: Application and Reviews*, 31(4):497–508, 2001.

[19] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(3): 125–141, 2008.

[20] Jakob Santner, Christian Leistner, Amir Saffari Thomas Pock, and Horst Bischof. Prost: Parallel robust online simple tracking. In *Proc. CVPR*, 2010.

[21] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proc. CVPR*, 2013.

[22] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.

[23] Ju Hong Yoon, Du Yong Kim, and Kuk-Jin Yoon. Visual tracking via adaptive tracker selection with multiple features. In *Proc. ECCV*, 2012.

[24] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *Proc. ECCV*, 2012.