Video Tracking Using Learned Hierarchical Features

Li Wang, *Member, IEEE,* Ting Liu, *Student Member, IEEE,* Gang Wang, *Member, IEEE,*
Kap Luk Chan, *Member, IEEE,* and Qingxiong Yang, *Member, IEEE*

***Abstract*—In this paper, we propose an approach to learn hierarchical features for visual object tracking. First, we offline learn features robust to diverse motion patterns from auxiliary video sequences. The hierarchical features are learned via a two-layer convolutional neural network. Embedding the temporal slowness constraint in the stacked architecture makes the learned features robust to complicated motion transformations, which is important for visual object tracking. Then, given a target video sequence, we propose a domain adaptation module to online adapt the pre-learned features according to the specific target object. The adaptation is conducted in both layers of the deep feature learning module so as to include appearance information of the specific target object. As a result, the learned hierarchical features can be robust to both complicated motion transformations and appearance changes of target objects. We integrate our feature learning algorithm into three tracking methods. Experimental results demonstrate that significant improvement can be achieved by using our learned hierarchical features, especially on video sequences with complicated motion transformations.**

***Index Terms*—Object tracking, deep feature learning, domain adaptation.**

I. Introduction

LEARNING hierarchical feature representation (also called deep learning) has emerged recently as a promising research direction in computer vision and machine learning. Rather than using hand-crafted features, deep learning aims to learn data-adaptive, hierarchical, and distributed representation from raw data. The learning process is expected to extract and organize discriminative information from data. Deep learning has achieved impressive performance on image classification [1], action recognition [2], and speech recognition [3], etc.

Feature representation is an important component for visual object tracking. Deep learning usually requires a lot of training data to learn deep structure and its related parameters. However, in visual tracking, only the annotation of the target object in the first frame of the test video sequence is available. Recently, Wang and Yeung [4] have proposed a so-called deep learning tracker (DLT). They propose to offline learn generic features from auxiliary natural images. However, using unrelated images for training, they cannot obtain deep features with temporal invariance, which is actually very important for visual object tracking. Moreover, they do not have an integrated objective function to bridge offline training and online tracking. They transfer knowledge from offline training to online tracking by simply feeding the deep features extracted from the pre-trained encoder to the target object classifier and tune the parameters of the pre-trained encoder when significant changes of object appearances are detected.

To address these two issues in DLT [4], we propose a domain adaptation based deep learning method to learn hierarchical features for model-free object tracking. Figure 1 presents an overview of the proposed feature learning method. First, we aim to learn deep features robust to complicated motion transformations of the target object, which are not considered by DLT [4]. Also, we intend to learn features which can handle a wide range of motion patterns in the test video sequences. Therefore, we adopt the feature learning method proposed in Zou et al. [6] as a basic model to pre-learn features robust to diverse motion patterns from auxiliary video sequences (offline learning part shown in Figure 1). Given the corresponding patches in the training video sequences, the basic model learns patch features invariant between two consecutive frames. As a result, high-level features which are robust to non-linear motion patterns can be discovered. Zou et al. [6] employ the learned features for generic object recognition. We argue that this method is also beneficial to object tracking, as temporal robustness can help a tracker to find corresponding patches reliably.

As stated above, Wang and Yeung [4] do not have an extra united objective function connecting offline learning and online tracking. As a result, the learned features from their method do not include appearance information of specific target objects. To solve this issue, we propose a domain adaptation module to effectively adapt the pre-learned features according to the specific target object (online learning part shown in Figure 1). The adaptation module is seamlessly incorporated into both layers of the stacked architecture of our deep learning model. As a result, the adapted features can be robust to both complicated motion transformations and appearance changes of the target object. As shown in Figure 1, we can observe that the adapted features are more relevant to the specific object "face" as they contain more facial edges and corners in the first layer and more semantic elements which look like faces or face parts in the second layer.

In order to capture appearance changes of specific target objects, we online adapt pre-learned generic features according to the new coming data of the test video sequence. Due to high dimensions of the parameter space in our deep learning model, we employ the limited memory BFGS (L-BFGS) algorithm [7] to solve the optimization problem in the adaptation module. As a result, convergence can be quickly reached in each adaptation.

We validate the proposed method on benchmark test video sequences. Experimental results demonstrate that significant

L. Wang, T. Liu, G. Wang and K.L. Chan are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: wa0002li@e.ntu.edu.sg; liut0016@e.ntu.edu.sg; wanggang@ntu.edu.sg; eklchan@ntu.edu.sg).

G. Wang is also with the Advanced Digital Science Center, Singapore.

Q. Yang is with the Department of Computer Science, City University of Hong Kong, China (e-mail: qiyang@cityu.edu.hk).

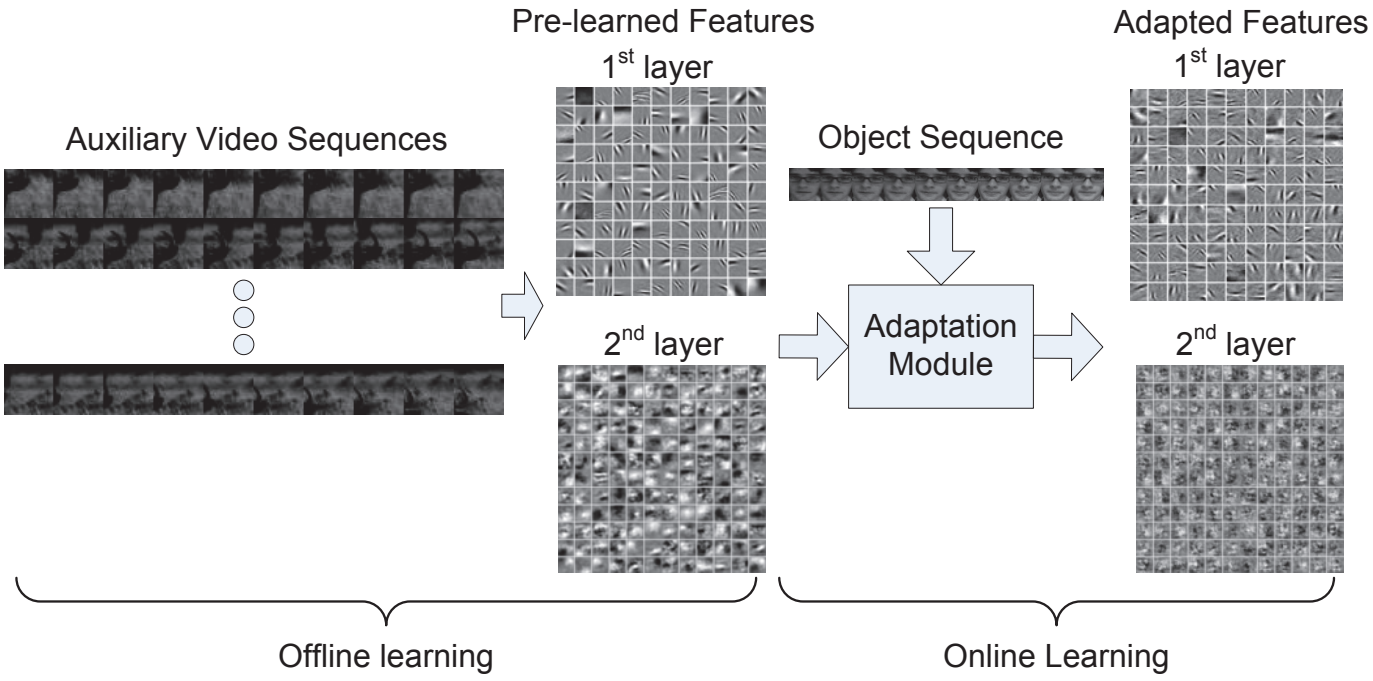


Fig. 1: Overview of the proposed feature learning algorithm. First, we pre-learn generic features from auxiliary data obtained from Hans van Hateren natural scene videos [5]. A number of learned feature filters from two layers are visualized. Then, we adapt the generic features to a specific object sequence. The adapted feature filters are also visualized, from which we can find that the adapted features are more relevant to the specific object "face" as they contain more facial edges and corners in the first layer and more semantic elements which look like faces or face parts in the second layer.

improvement can be obtained by using our learned hierarchical features for object tracking.

II. Related Work

Object tracking For decades, many interesting methods have been proposed for object tracking which has a wide range of applications, e.g. video surveillance [8] [9]. Eigentracker [10] has had a deep impact on subspace based trackers [11] [12]. The method named as "Condensation" [13] is well-known because it is the first one to apply particle filter [14] to object tracking. In [15], mean-shift [16] is used to optimize the target localization problem in visual tracking. The "Lucas-Kanade" algorithm [17] is famous for defining the cost function by using the sum of squared difference (SSD). Another pioneering method [18] paves the way for the subsequent trackers based on the adaptive appearance model (AAM).

Recently, the tracking problem has also been considered as a binary classification problem due to the significant improvement on object recognition [19] [20]. In [21], the Support Vector Machine (SVM) is integrated into an optical-flow based tracker. Subsequently, the ensemble tracker [22] trains an ensemble of weak classifiers online to label pixels as objects or backgrounds. In [23], an online semi-supervised boosting method is proposed to handle the drifting problem caused by inaccuracies from updating the tracker. In [24], on-line multiple instance learning is also proposed to solve the drifting problem. P-N learning [25] is proposed to train a binary classifier from labeled and unlabeled examples which are iteratively corrected by positive (P) and negative (N) constraints. Also, correlation filters [26] have achieved very promising results for visual object tracking.

Many advanced trackers are also developed based on sparse representation [27]. ℓ_1 tracker [28] solves an ℓ_1-regularized least squares problem to achieve the sparsity for target candidates, in which the one with the smallest reconstruction error is selected as the target in the next frame. Two pieces of works [29] [30] focus on accelerating the ℓ_1 tracker [28] because the ℓ_1 minimization requires high computational costs. There are some other promising sparse trackers [31] [32] [33]. The tracker [33] employing the adaptive structural local sparse appearance model (ASLA) achieves especially good performance, and this is used as the baseline tracking system in this paper.

Feature representation Some tracking methods focus on feature representation. In [34], an online feature ranking mechanism is proposed to select features which are capable of discriminating between object and background for visual tracking. Similarly, an online AdaBoost feature selection algorithm is proposed in [35] to handle appearance changes in object tracking. In [36], keypoint descriptors in the region of the interested object are learned online together with background information. The compressive tracker (CT) [37] employs random projections to extract data independent features for the appearance model and separates objects from backgrounds using a naive Bayes classifier. Recently, Wang and Yeung [4] has proposed to learn deep compact features for visual object tracking.

Deep learning Deep learning [38] [39] has recently attracted much attention in machine learning. It has been successfully applied in many computer vision applications, such as visual object tracking [40], shape modeling [41], action recognition [42], image set classification [43], attribute prediction [44],

face recognition [45], scene image classification [46] and scene labeling [47] [48]. Deep learning aims to replace hand-crafted features with high-level and robust features learned from raw pixel values, which is also known as unsupervised feature learning [49] [50] [51] [52]. In [6], the temporal slowness constraint [53] is combined with deep neural networks to learn hierarchical features. Inspired by this work, we intend to learn deep features to handle complicated motion transformations in visual object tracking.

Domain adaptation Recently, there have been increasing interests in visual domain adaptation problems. Saenko et al. [54] apply domain adaptation to learn object category models. In [55], domain adaptation techniques are developed to detect video concepts. In [56], Duan et al. adapt learned models from web data to recognize visual events. Recently, Glorot et al. [57] develop a meaningful representation for large-scale sentiment classification by combining deep learning and domain adaptation. Domain adaptation has also been applied in object tracking. Wang et al. [58] pre-learn an over-complete dictionary and transfer the learned visual prior for tracking specific objects.

Our method The principles behind our method are deep learning and domain adaptation learning. We first utilize the temporal slowness constraint to offline learn generic hierarchical features robust to complicated motion transformations. Then, we propose a domain adaptation module to adapt the pre-learned features according to the specific target object. The differences between DLT [4] and our method are as follows. First, their method pre-learns features from untracked images. In contrast, our method uses tracked video sequences and focuses on learning features robust to complex motion patterns. Second, their method does not have a united objective function with the regularization term for domain adaptation, whereas our method has an adaptation module integrating the specific target object's appearance information into the pre-learned generic features. Our method is also different from [58], in which the dictionary is pre-defined and the tracking object is reconstructed by the patterns in the pre-defined dictionary. The method in [58] may fail if the pre-defined dictionary does not include the visual patterns of the target object. Last, it is necessary to mention that Zou et al. [6] learn hierarchical features from video sequences with tracked objects for image classification whereas our method focuses on visual object tracking.

III. Tracking System Overview

We aim to learn hierarchical features to enhance the state-of-the-art tracking methods. The tracking system with the adaptive structural local sparse appearance model (ASLA) [33] achieves very good performance. Hence, we integrate our feature learning method into this system. But note that our feature learning method is general for visual tracking, and it can be used with other tracking systems as well by replacing original feature representations.

In this section, we briefly introduce the tracking system. Readers may refer to [33] for more details. Suppose we have an observation set of target $x_{1:t} = \{x_1, \ldots, x_t\}$ up to the t^{th} frame and a corresponding feature representation set $z_{1:t} = \{z_1, \ldots, z_t\}$, we can calculate the target state y_t as follows

$$y_t = \arg\max_{y_t^i} p\left(y_t^i | z_{1:t}\right) \tag{1}$$

where y_t^i denotes the state of the i^{th} sample in the t^{th} frame. The posterior probability $p\left(y_t | z_{1:t}\right)$ can be inferred by the Bayes' theorem as follows

$$p\left(y_t | z_{1:t}\right) \propto p\left(z_t | y_t\right) \int p\left(y_t | y_{t-1}\right) p\left(y_{t-1} | z_{1:t-1}\right) dy_{t-1} \tag{2}$$

where $z_{1:t}$ denotes the feature representation, $p\left(y_t | y_{t-1}\right)$ denotes the motion model and $p\left(z_t | y_t\right)$ denotes the appearance model. In [33], the representations $z_{1:t}$ simply use raw pixel values. In contrast, we propose to learn hierarchical features from raw pixels for visual tracking.

IV. Learning Features for Video Tracking

Previous tracking methods usually use raw pixel values or hand-crafted features to represent target objects. However, such features cannot capture essential information which is invariant to non-rigid object deformations, in-plane and out-of-plane rotations in object tracking. We aim to enhance tracking performance by learning hierarchical features which have the capability of handling complicated motion transformations. To achieve this, we propose a domain adaptation based feature learning algorithm for visual object tracking. We first adopt the approach proposed in [6] to learn features from auxiliary video sequences offline. These features are robust to complicated motion transformations. However, they do not include appearance information of specific target objects. Hence, we further use a domain adaptation method to adapt pre-learned features according to specific target objects.

We integrate our feature learning method into the tracking system ASLA [33] and its details are given in Algorithm 1.

Algorithm 1 Our tracking method

1: **Input:** the previous tracking state y_{t-1}, the existing feature learning parameter $\hat{\Theta}$ and the exemplar library.
2: Apply the affine transformation on y_{t-1} to obtain a number of tracking states y_t^i and the corresponding candidate image patches x_t^i.
3: Extract feature representations z_t^i from the candidate image patches x_t^i under the existing feature learning parameter $\hat{\Theta}$.
4: Calculate the posterior probability $p\left(y_t^i | z_{1:t}\right)$ according to Equation 2.
5: Predict the tracking state by $y_t = \arg\max_{y_t^i} p\left(y_t^i | z_{1:t}\right)$.
6: Update the feature learning parameter and the exemplar library every M frames.
7: **Output:** the predicted tracking state y_t, the up-to-date feature learning parameter Θ and the up-to-date exemplar library.

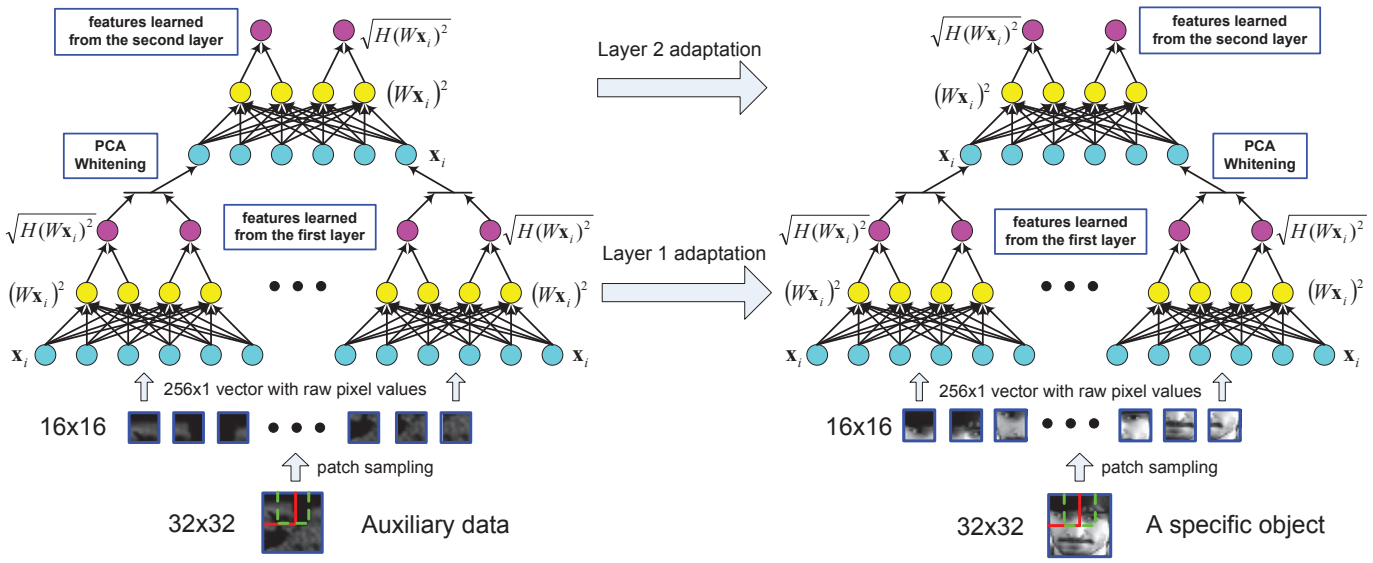


Fig. 2: Stacked architecture of our deep feature learning algorithm. The output of the first layer is whitened using PCA and then used as the input of the second layer. For the adaptation module, given a specific object sequence, the pre-learned features learned from auxiliary data are adapted respectively in two layers by minimizing the objective function in Equation 4.

A. *Pre-Learning Generic Features from Auxiliary Videos*

Since the appearance of an object could change significantly due to its motion, a good tracker desires features robust to motion transformations. Inspired by [6], we believe that there exist generic features robust to diverse motion patterns. Therefore, we employ the deep learning model in [6] to learn hierarchical features from auxiliary videos [5] to handle diverse motion transformations of objects in visual tracking. Note that this is performed offline.

The deep learning model has two layers as illustrated in Figure 2. In our case, the first layer works on smaller patches (16×16). The second layer works on larger patches (32×32). We learn the feature transformation matrix W of each layer as below.

Given the offline training patch $\mathbf{x}_i$ from the i^{th} frame, we denote the corresponding learned feature as $\mathbf{z}_i = \sqrt{H(W\mathbf{x}_i)^2}$, where H is the pooling matrix and $(W\mathbf{x}_i)^2$ is the element-wise square on the output of the linear network layer. To better explain the basic learning module in each layer, we make use of the illustration in Figure 2. First, it is necessary to mention that the blue, yellow and purple circles denote the input vector $\mathbf{x}_i$, the intermediate vector $(W\mathbf{x}_i)^2$ and the output vector $\sqrt{H(W\mathbf{x}_i)^2}$ respectively w.r.t. the basic learning module. Then, H can be illustrated as the transformations between the intermediate vector (yellow circles) and the output one (purple circles). The pooling mechanism is to calculate the summation of two adjacent feature dimensions of the intermediate vector (yellow circles) in a non-overlapping fashion. Also, W can be illustrated as the transformations between the input vector (blue circles) and the intermediate one (yellow circles). Essentially, each row of the feature transformation matrix W can be converted to an image patch filter as shown in Figure 1. The feature transformation matrix W is learned by solving the following unconstrained minimization problem,

$$\min_{W} \lambda \sum_{i=1}^{N-1} \|\mathbf{z}_i - \mathbf{z}_{i+1}\|_1 + \sum_{i=1}^{N} \|\mathbf{x}_i - W^T W \mathbf{x}_i\|_2^2, \quad (3)$$

where $\mathbf{z}_{i+1}$ denotes the learned feature from the $(i+1)^{th}$ frame and N is the total length of all video sequences in the auxiliary data. Essentially, multiple video sequences are organized sequence-by-sequence. Between two sequences, our learning algorithm does not take into account the differences between the non-continuous frames, the last frame $\mathbf{z}_i$ of the current sequence and the first frame $\mathbf{z}_{i+1}$ of the next sequence. The first term forces learned features to be temporally continuous and the second term is an auto-encoder reconstruction cost [51]. As a result, we obtain the feature $\mathbf{z}$ which is robust to complicated motion transformations.

The input of the first layer is the raw pixel values of smaller patches (16×16). We can learn the feature transformation matrix W^{L1} for the first layer by Equation 3. Then, we apply W^{L1} to convolve with the larger patches (32×32). The larger patch is divided into a number of sub-patches (16×16). We use W^{L1} to conduct feature mapping for each sub-patch and concatenate features of all the sub-patches to represent the larger patch. Next, PCA whitening is applied to the concatenated feature vector. Finally, we use the whitened feature vector of the larger patch as the input to the second layer and learn the feature transformation matrix W^{L2} for the second layer.

The first layer can extract features robust to local motion patterns e.g. translations. From the second layer, we could extract features robust to more complicated motion transformations e.g. non-linear warping and out-of-plane rotations (See Figure 1). We concatenate features from two layers as our generic features. Moreover, we pre-learn the generic features from a lot of auxiliary video data. As a result, the pre-learned

features can provide our tracker with capabilities of handling diverse motion patterns.

B. Domain Adaption Module

Although the generic features are robust to non-linear motion patterns in visual tracking, they do not include appearance information of specific target objects, e.g. shape and texture. Hence, we propose a domain adaptation module to adapt the generic features according to specific target objects. The domain adaptation module is illustrated in Figure 2.

Given a target video sequence, we employ ASLA [33] to track the target object in the first N frames and use the tracking results as the training data for the adaptation module. The adapted feature is denoted as $\mathbf{z}_i^{adp} = \sqrt{H(W\mathbf{x}_i^{obj})^2}$, where $\mathbf{x}_i^{obj}$ indicates the object image patch in the i^{th} frame of the training data for adaptation and W is the feature transformation matrix to be learned. We formulate the adaptation module by adding a regularization term as follows,

$$\begin{aligned} W_{adp} &= \arg\min_{W} \lambda \sum_{i=1}^{N-1} \|\mathbf{z}_i^{adp} - \mathbf{z}_{i+1}^{adp}\|_1 \\ &+ \gamma \sum_{i=1}^{N} \|W\mathbf{x}_i^{obj} - W_{old}\mathbf{x}_i^{obj}\|_2^2 \\ &+ \sum_{i=1}^{N} \|\mathbf{x}_i^{obj} - W^T W\mathbf{x}_i^{obj}\|_2^2, \end{aligned} \tag{4}$$

where W_{old} denotes the pre-learned feature transformation matrix. The second term refers to the adaptation module and aims to make the adapted feature close to the old one for the sake of preserving the pre-learned features' robustness to complicated motion transformations. Meanwhile, using the training data $\mathbf{x}_i^{obj}$ is intended to include the appearance information of the specific target object, e.g. shape and texture. γ is the trade-off parameter which controls the adaptation level.

We adapt the generic features in a two-layer manner. It means that we conduct the minimization in Equation 4 with respect to W in both layers respectively.

C. Optimization and Online Learning

Succinctly, we denote the objective function of the adaptation module as $f(\mathbf{X}; \Theta, \hat{\Theta})$, where $\mathbf{X}$ denotes a number of training images of object regions for the adaptation, $\Theta = \{w_{ij} | i, j = 1, \ldots, N\}$ indicates the parameter set representing all entries in the transformation matrix W and $\hat{\Theta}$ refers to the known parameter set w.r.t. W_{old}. We employ limited-memory BFGS (L-BFGS) algorithm [7] to optimize the objective function $f(\mathbf{X}; \Theta, \hat{\Theta})$ w.r.t. the parameter set Θ.

The Quasi-Newton methods, such as BFGS algorithm [59], need to update the approximate Hessian matrix B_k at the i^{th} iteration to calculate the search direction $p_k = -B_k^{-1}\nabla f_k$, where ∇f_k is the derivative of the objective function f w.r.t. Θ_k at the k^{th} iteration. The cost of storing the approximate Hessian matrix B_k ($N^2 \times N^2$) is prohibitive in our case because the dimension N^2 of the parameter set Θ is high ($\approx 10^4$). Therefore, we use L-BFGS in which the search direction p_k is calculated based on the current gradient ∇f_k and the curvature information from m most recent iterations $\{s_i = \Theta_{i+1} - \Theta_i, y_i = \nabla f_{i+1} - \nabla f_i | i = k-m, \ldots, k-1\}$. Algorithm 2 presents calculation on L-BFGS search direction p_k. In our implementation, m is set to 5.

Algorithm 2 Calculation on L-BFGS search direction p_k

Input: the derivative ∇f_k of the objective function f w.r.t. Θ_k, the curvature information from m most recent iterations $\{s_i, y_i | i = k-m, \ldots, k-1\}$.
$p_k = -\nabla f_k$;
for $i = k-1, k-2, \ldots, k-m$ **do**
  $\alpha_i = \frac{s_i^T p_k}{y_i^T s_k}$;
  $p_k = p_k - \alpha_i y_i$;
end for
$p_k = B_0^{-1} p_k$
for $i = k-m, k-m+1, \ldots, k-1$ **do**
  $\beta = \frac{y_i^T p_k}{y_i^T s_i}$;
  $p_k = p_k + s_i(\alpha_i - \beta)$;
end for
Output: L-BFGS search direction p_k

Given the search direction p_k obtained from Algorithm 2, we compute $\Theta_{k+1} = \Theta_k + \alpha_k p_k$, where α_k is chosen to satisfy the Wolfe conditions [59]. When $k > m$, we discard the curvature information $\{s_{k-m}, y_{k-m}\}$ and compute and save the new one $\{s_k = \Theta_{k+1} - \Theta_k, y_k = \nabla f_{k+1} - \nabla f_k\}$. Using L-BFGS to optimize the adaptation formulation, the convergence can be reached after several iterations.

To capture appearance changes of target objects, we online learn the parameter set Θ of the adaptation module every M frames. We also use L-BFGS algorithm to solve the minimization problem $\arg min_{\Theta} f(\Theta; \mathbf{X}, \tilde{\Theta})$, where $\mathbf{X} = \{\mathbf{x}_1 : \mathbf{x}_M\}$ denotes training data within object regions from M most recent frames and $\tilde{\Theta}$ indicates the old parameter set. The learned parameter set Θ converges quickly in the current group of M frames and it will be used as the old parameter set $\tilde{\Theta}$ in the next group of M frames. In our implementation, M is set to 20 in all test video sequences.

D. Implementation Details

Auxiliary data We pre-learn the generic features using the auxiliary data from Hans van Hateren natural scene videos [5]. As mentioned in [6], features learned from sequences containing tracked objects can encode more useful information such as non-linear warping. Hence, we employ video sequences containing tracked objects for pre-learning features (see Figure 1).

Initialization We use tracking results from ASLA [33] in the first 20 frames as the initial training data for our adaptation module. It is fair to compare with other methods under this setting. Many tracking methods have this sort of initialization. For example, Jia et al. [33] utilize a k-d tree matching scheme to track target objects in first 10 frames of sequences and then build exemplar libraries and patch dictionaries based on these tracking results.

TABLE I: Average center error (in pixels). The best two results are shown in red and blue fonts. We compare our tracker using learned features with 4 state-of-the-art trackers using other feature representations: the raw pixel values (ASLA[33]_RAW), the hand-crafted HOG feature (ASLA[33]_HOG), the sparse representation (ℓ_1_APG [60]) and the data-independent feature (CT_DIF [37]). We also present the results of the variant of our tracker (Ours_VAR) which does not use the temporal slowness constraint in feature learning.

Sequence	ASLA[33]_RAW	ASLA[33]_HOG	ℓ_1_APG[60]	CT_DIF[37]	Ours_VAR	Ours
Basketball	70.2	245.6	107.2	123.6	14.0	11.2
Biker	88.0	68.2	79.6	80.7	19.1	11.7
David2	24.0	25.1	44.1	59.8	12.4	1.5
FleetFace	129.2	180.0	123.5	145.5	30.2	24.8
Freeman1	79.2	69.8	21.7	17.8	9.2	8.7
Freeman3	21.4	43.4	15.6	65.6	6.5	4.4
Kitesurf	40.3	38.5	71.7	62.1	22.6	13.2
Lemming	165.5	155.4	138.2	149.3	8.6	6.7
MountainBike	185.9	155.2	210.1	212.9	10.3	9.2
Shaking	86.5	86.6	113.0	30.9	37.9	15.2
Skating1	14.6	15.6	72.3	87.9	6.6	6.5
Sylvester	27.2	27.7	31.6	17.6	52.1	6.4
Tiger1	71.2	112.9	61.7	85.6	70.3	40.9
Tiger2	61.8	96.2	58.4	83.3	27.1	19.2
Trellis	31.9	18.8	62.5	47.4	13.2	3.0
Average	73.1	89.3	80.7	84.7	22.7	12.2

TABLE II: Average overlap rate (%). The best two results are shown in red and blue fonts. We compare our tracker using learned features with 4 state-of-the-art trackers using other feature representations: the raw pixel values (ASLA[33]_RAW), the hand-crafted HOG feature (ASLA[33]_HOG), the sparse representation (ℓ_1_APG [60]) and the data-independent feature (CT_DIF [37]). We also present the results of the variant of our tracker (Ours_VAR) which does not use the temporal slowness constraint in feature learning.

Sequence	ASLA[33]_RAW	ASLA[33]_HOG	ℓ_1_APG[60]	CT_DIF[37]	Ours_VAR	Ours
Basketball	0.46	0.27	0.29	0.36	0.54	0.59
Biker	0.52	0.41	0.29	0.40	0.51	0.67
David2	0.59	0.26	0.29	0.05	0.61	0.85
FleetFace	0.13	0.13	0.41	0.24	0.58	0.60
Freeman1	0.32	0.28	0.34	0.27	0.45	0.51
Freeman3	0.47	0.52	0.45	0.13	0.49	0.59
Kitesurf	0.35	0.36	0.34	0.24	0.50	0.63
Lemming	0.42	0.41	0.40	0.41	0.69	0.76
MountainBike	0.27	0.37	0.40	0.25	0.61	0.68
Shaking	0.10	0.06	0.06	0.33	0.50	0.58
Skating1	0.38	0.35	0.31	0.26	0.35	0.42
Sylvester	0.35	0.47	0.37	0.55	0.39	0.71
Tiger1	0.25	0.24	0.42	0.15	0.27	0.50
Tiger2	0.32	0.21	0.38	0.10	0.44	0.53
Trellis	0.49	0.49	0.28	0.28	0.69	0.79
Average	0.36	0.32	0.34	0.27	0.51	0.63

Computational cost Learning generic features consumes much time (about 20 minutes) due to the large training dataset. However, it is conducted offline, hence it does not matter. For the online adaptation part, we initialize the transformation matrix W to be learned with the pre-learned W_{old}. Based on the training data collected online, each update of the adaptation module takes only several iterations to achieve the convergence. Another part is feature mapping, in which it is required to extract features from candidate image patches. ASLA [33] requires to sample 600 candidate patches in each frame. We find that it is very expensive if we conduct feature mapping for all candidate patches. Therefore, we conduct a coarse-to-fine searching strategy, in which we first select a number of (e.g. 20) promising candidates in each frame according to the tracking result from ASLA [33] using raw pixel values and then refine the ranking of candidates based on our learned hierarchical features. We run the experiments on a PC with a Quad-Core 3.30 GHz CPU and 8 GB RAM. However, we do not use the multi-core setting of the PC. The speed of our tracker (about 0.8 fps) is roughly twice slower than the one of ASLA [33] (about 1.6 fps) due to the additional feature extraction step. The time (about 625 ms) spent on the feature extraction is about same as on the other parts of our tracker. Note that the main objective here is to show that our learned hierarchical features can improve tracking accuracy. The efficiency of our tracker could be improved further because feature mapping for different patches could be conducted in parallel by advanced techniques, e.g. GPU. Finally, we empirically tune the trade-off parameters of λ and γ in Equation 4 for different sequences. However, the parameters change in a small range. λ and γ are tuned in $[1, 10]$ and $[90, 110]$ respectively.

V. Experiments

First, we evaluate our learned hierarchical features to demonstrate its robustness to complicated motion transformations. Second, we evaluate the temporal slowness constraint and the adaptation module in our feature learning algorithm. Third, we evaluate our tracker's capability of handling typical problems in visual tracking. Then, we compare our tracker with 14 state-of-the-art trackers. Moreover, we present the comparison results between DLT [4] and our tracker. Finally, we present the generalizability of our feature learning algorithm on the other 2 tracking methods.

We use two measurements to quantitatively evaluate tracking performances. The first one is called center location error which measures distances of centers between tracking results and ground truths in pixels. The second one is called overlap rate which is calculated according to $\frac{area(R_T \cap R_G)}{area(R_T \cup R_G)}$ and indicates extent of region overlapping between tracking results R_T and ground truths R_G. It is necessary to mention that there are often subjective biases in evaluating tracking algorithms as indicated in [61].

A. Evaluation on Our Learned Feature's Robustness to Complicated Motion Transformations

We present both quantitative and qualitative results on 15 challenging sequences, in which target objects have complicated motion transformations. e.g. in-plane rotation, out-of-plane rotation and non-rigid object deformation. To demonstrate our learned feature's robustness to complicated motion transformations, we compare our tracker with the other 4 state-of-the-art trackers using different feature representations such as the raw pixel value (ASLA[33]_RAW), the hand-crafted feature of Histogram of Oriented Gradients (HOG) [62] (ASLA[33]_HOG), the sparse representation (ℓ_1_APG [60]) and the data-independent feature (CT_DIF [37]). It is necessary to mention that ASLA_HOG and our tracker use the same tracking framework as in ASLA_RAW [33]. The difference is that ASLA_HOG and our tracker integrate the HOG feature and our learned hierarchical features into the baseline ASLA tracker respectively. However, the other 2 trackers, ℓ_1_APG and CT_DIF, use their own tracking frameworks which are different from ASLA_RAW. The hand-

Fig. 3: Qualitative results on sequences with non-rigid object deformation. The purple, green, cyan, blue and red bounding boxes refer to ASLA[33]_RAW, ASLA[33]_HOG, ℓ_1_APG [60], CT_DIF [37] and our tracker respectively. This figure is better viewed in color.

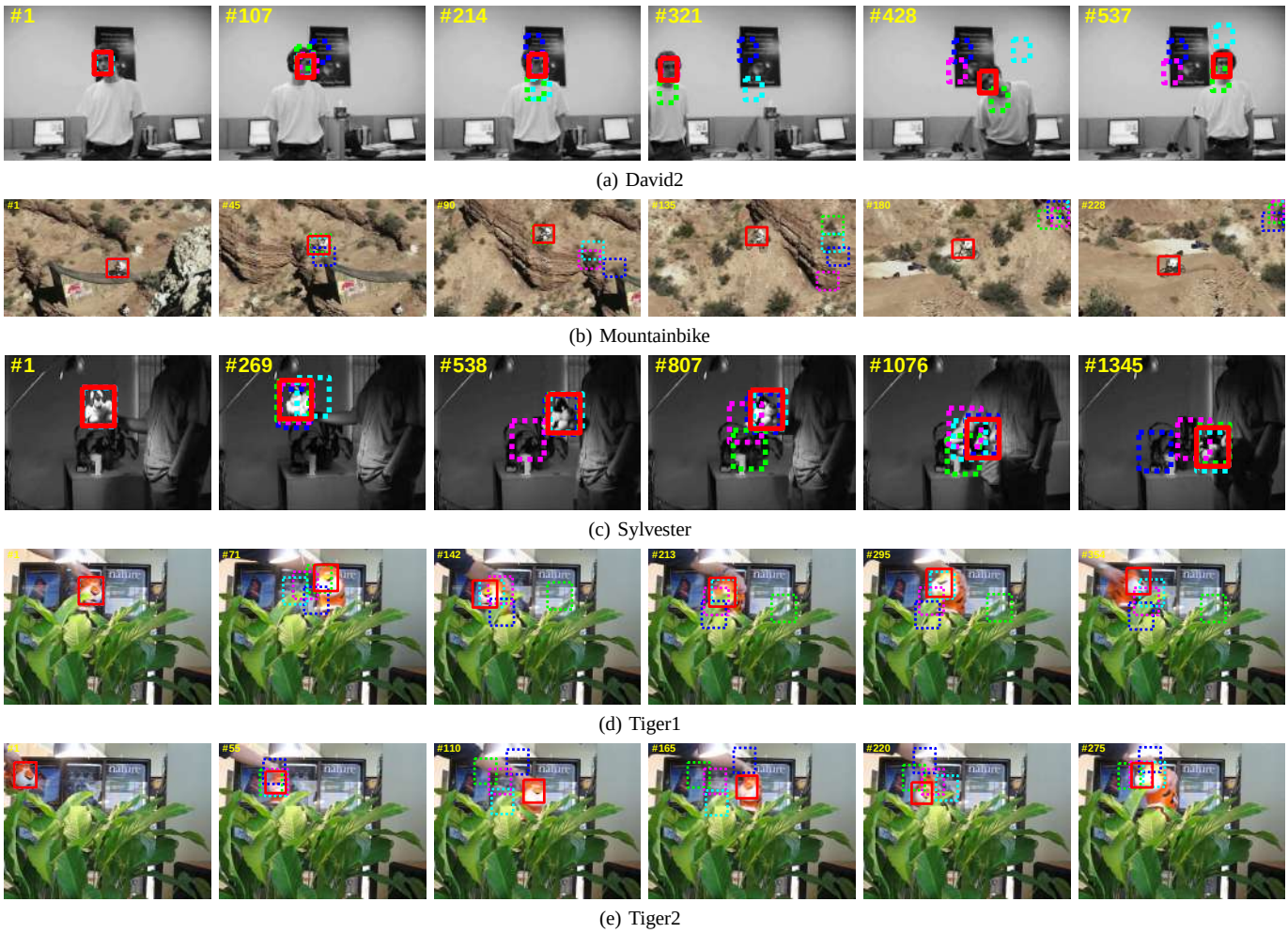
crafted HOG feature and the sparse feature are employed here because of their superior performances in object detection and recognition. Additionally, the data-independent feature is used here because it also aims to solve the problem of insufficient training data in object tracking.

In this evaluation, we test on 13 sequences used in [63]. Also, we have two special sequences of "biker" and "kitesurf", in which the original video sequences are used, but new target objects are defined for tracking. Our sequences are challenging because the newly defined objects contain complicated motion transformations. For example, in the sequence of "biker" (see Figure 3), we track the biker's whole body which has non-rigid object deformation. Tables I and II present quantitative results which demonstrate that our learned features outperform the other state-of-the-art feature representations in terms of handling complicated motion transformations well. Figures 3, 4 and 5 show the qualitative results on sequences with non-rigid object deformation, in-plane rotations and out-of-plane rotations respectively. Then, we explain the qualitative results as follows.

Non-rigid object deformation The sequences (Basketball, Biker, FleetFace, Kitesurf and Skating1) shown in Figure 3 are challenging because the target objects have non-rigid object deformations. For example, the basketball player in Figure 3 (a) has deformable changes due to his running and defending actions. The biker in Figure 3 (b) has dramatic body deformations during his acrobatic actions. The man in Figure 3 (c) has significant facial changes due to his laughing expression. The person in Figure 3 (d) has deformable pose changes because of his surfing actions. The girl in Figure 3 (e) has articulated deformations caused by her arm waving and body spinning. We can observe that the 4 baseline trackers (ASLA[33]_RAW, ASLA[33]_HOG, ℓ_1_APG [60] and CT_DIF [37]) fail to track the target objects in these challenging sequences. In contrast, our tracker succeeds to capture the target objects because our features are learned to be invariant to non-rigid object deformations.

In-plane rotations The target objects in the sequences (David2, MountainBike, Sylvester, Tiger1 and Tiger2) have significant in-plane rotations which are difficult for trackers to capture. In Figure 4 (a), the man's face not only has translations but also in-plane rotations which occur when the face is slanted. In Figure 4 (b), the mountain bike has the in-plane rotations due to its acrobatic actions in the sky. In Figure 4 (c), (d) and (e), the toys have a lot of in-plane rotations. We can see that all the baseline trackers have drifted away from the target objects in these sequences because of in-plane rotations, whereas our tracker can handle this kind of motion transformations effectively by using learned features.

Out-of-plane rotations The sequences (Freeman1, Freeman3, Lemming, Shaking and Trellis) are difficult because the target objects have out-of-plane rotations which change object



(a) David2

(b) Mountainbike

(c) Sylvester

(d) Tiger1

(e) Tiger2

Fig. 4: Qualitative results on sequences with in-plane rotations. The purple, green, cyan, blue and red bounding boxes refer to ASLA[33]_RAW, ASLA[33]_HOG, ℓ_1_APG [60], CT_DIF [37] and our tracker respectively. This figure is better viewed in color.

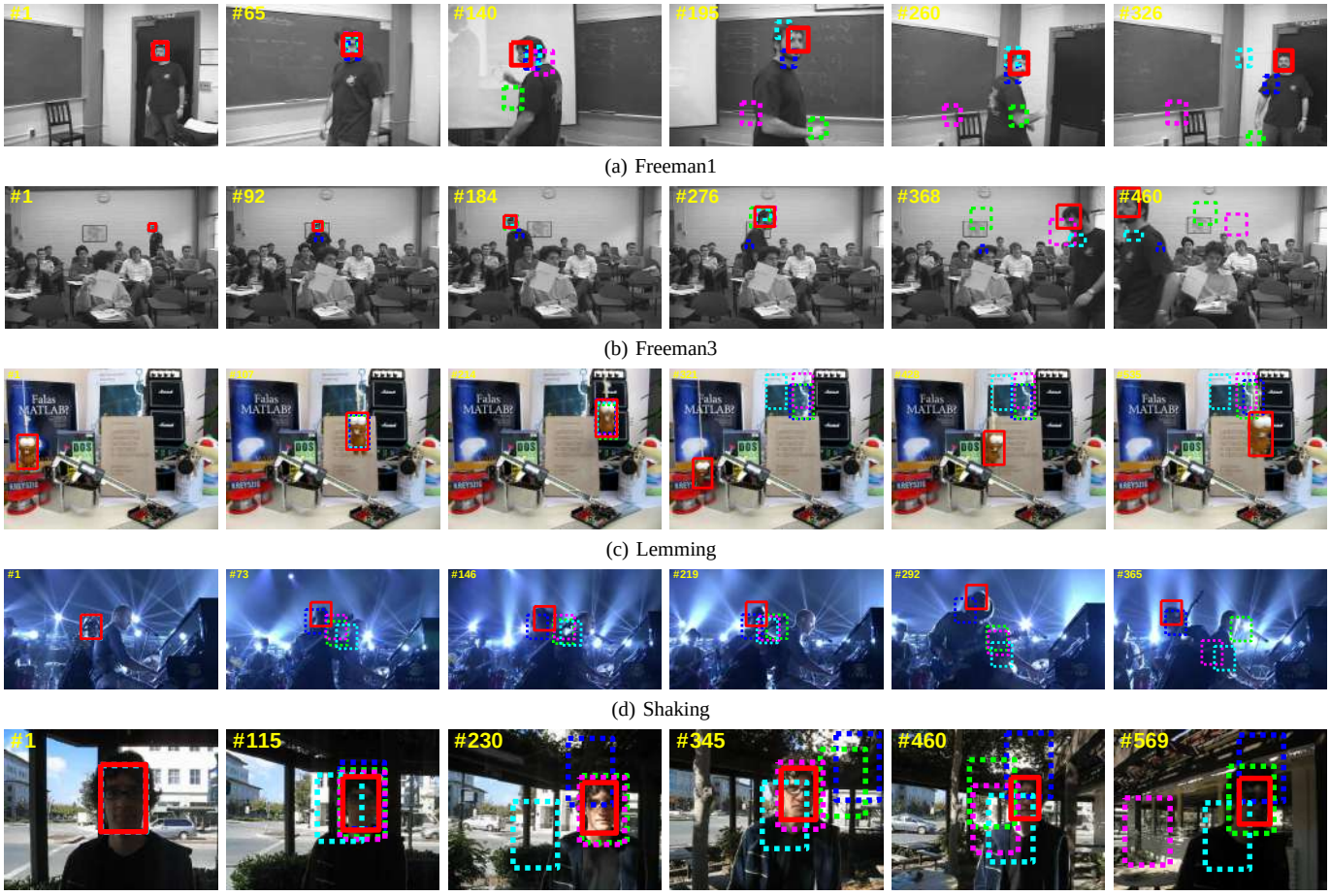
appearances significantly and hence yield tracking failures. For instance, in Figure 5 (a), (b) and (e), the men' faces have significant out-of-plane rotations because the poses of their heads change a lot during walking. The toy in Figure 5 (c) has out-of-plane rotations because it rotates along its vertical axis. The singer's head shown in Figure 5 (d) has out-of-plane rotations because the head shakes up and down. We can observe that our tracker can successfully capture the target objects through these sequences. We owe this success to our learned feature's robustness to out-of-plane rotations. In contrast, the baseline trackers cannot handle this complicated motion transformation because their feature representations are not designed to capture motion invariance.

B. Evaluation on the Temporal Slowness Constraint and the Adaptation Module in Our Feature Learning Algorithm

First, we present the results of the variant of our tracker (Ours_VAR) which does not use the temporal slowness constraint in feature learning in Tables I and II. We can observe that our tracker using the constraint has better performances on 15 challenging video sequences. It demonstrates that the temporal slowness constraint is beneficial for learning features robust to complicated motion transformations. Then, we evaluate the adaptation module in our feature learning method on 8 video sequences reported in ASLA [33]. Tables III and IV respectively present the average center location errors and the average overlap rates of our tracker with (Ours_adp) and without (Ours_noadp) the adaptation module. From the quantitative comparison, we can find that the adaptation module enhances the performance of our tracker. It is due to the fact that the adaptation module not only preserves the pre-learned features' robustness to complicated motion transformations, but also includes appearance information of specific target objects.

C. Evaluation on Our Tracker's Capability of Handling Typical Problems in Visual Tracking

We use the 8 sequences in ASLA [33] to evaluate our tracker's capability of handling typical problems in visual tracking, e.g. illumination change, occlusion and cluttered background. We quantitatively compare our tracker with 4 baseline trackers, ASLA[33]_RAW, ASLA[33]_HOG, ℓ_1_APG [60] and CT_DIF [37], which use the raw pixel values, the hand-crafted HOG feature, the sparse representation and the data-independent feature respectively. From Tables III and IV, we can find that our learned features are



(a) Freeman1

(b) Freeman3

(c) Lemming

(d) Shaking

(e) Trellis

Fig. 5: Qualitative results on sequences with out-of-plane rotations. The purple, green, cyan, blue and red bounding boxes refer to ASLA[33]_RAW, ASLA[33]_HOG, ℓ_1_APG [60], CT_DIF [37] and our tracker respectively. This figure is better viewed in color.

TABLE III: Average center error (in pixels). The best two results are shown in red and blue fonts. We present our tracker's performances with (Ours_adp) and without (Ours_noadp) the adaptation module. We also compare our tracker with 4 baseline trackers using other features such as the raw pixel values (ASLA[33]_RAW), the hand-crafted HOG feature (ASLA[33]_HOG), the sparse feature (ℓ_1_APG [60]) and the data-independent feature (CT_DIF[37]).

Sequence	ASLA[33]_RAW	ASLA[33]_HOG	ℓ_1_APG[60]	CT_DIF[37]	Ours_noadp	Ours_adp
Board	7.3	15.3	259.4	80.3	7.1	8.1
Car11	2.0	2.7	22.2	78.0	1.9	1.4
Caviar	2.3	66.8	95.6	65.5	2.2	2.2
David	3.6	45.8	138.2	12.8	3.5	3.2
Faceocc2	3.8	32.4	17.7	12.8	3.7	3.1
Singer1	4.8	5.0	167.9	13.7	4.5	4.0
Stone	1.8	2.8	136.8	32.4	2.1	1.5
Woman	2.8	140.6	176.1	110.2	2.6	2.2

TABLE IV: Average overlap rates. (%) The best two results are shown in red and blue fonts. We present our tracker's performances with (Ours_adp) and without (Ours_noadp) the adaptation module. We compare our tracker with 4 baseline trackers using other features such as the raw pixel values (ASLA[33]_RAW), the hand-crafted HOG feature (ASLA[33]_HOG), the sparse feature (ℓ_1_APG [60]) and the data-independent feature (CT_DIF[37]).

Sequence	ASLA[33]_RAW	ASLA[33]_HOG	ℓ_1_APG[60]	CT_DIF[37]	Ours_noadp	Ours_adp
Board	0.74	0.72	0.12	0.33	0.73	0.83
Car11	0.81	0.71	0.34	0.23	0.80	0.85
Caviar	0.84	0.40	0.06	0.33	0.87	0.88
David	0.79	0.27	0.19	0.56	0.80	0.81
Faceocc2	0.82	0.52	0.61	0.68	0.87	0.88
Singer1	0.81	0.79	0.14	0.34	0.82	0.85
Stone	0.56	0.57	0.16	0.33	0.59	0.60
Woman	0.78	0.35	0.20	0.41	0.81	0.84

more competitive than the other 4 feature representations for handling typical issues in visual tracking.

D. Comparison with the State-of-the-art Trackers

We compare our tracker against 14 state-of-the-art algorithms on 10 video sequences used in previous works [12] [24] [66] [71] [72]. Tables V and VI respectively show the average center location errors and the average overlap rates of different tracking methods. Our tracker outperforms other state-of-the-art tracking algorithms in most cases and especially improves the baseline ASLA [33]. We owe this success to our learned hierarchical features.

E. Comparison between DLT and Our Tracker

We present the comparison results in terms of average center error (in pixels) between DLT [4] and our tracker in Table VII. We can observe that our tracker outperforms DLT on 5 of 8 sequences.

F. Evaluation on Our Learned Feature's Generalizability

To demonstrate the generalizability of our learned features, we integrate our feature learning algorithm into another baseline tracker which is called the incremental learning

TABLE V: Average center error (in pixels). The best two results are shown in red and blue fonts. We compare the proposed tracker with FragT [64], IVT [12], ℓ_1T [28], MIL [24], TLD [65], VTD [66], LSK [32], CT [37], ASLA [33] ℓ_1APG [60], MTT [67], SCM [68], OSPT [69] and LSST [70]. Our tracker outperforms the state-of-the-art tracking algorithms.

Sequence	FragT[64]	IVT[12]	ℓ_1T[28]	MIL[24]	TLD[65]	VTD[66]	LSK[32]	CT[37]	ASLA[33]	ℓ_1APG[60]	MTT[67]	SCM[68]	OSPT[69]	LSST[70]	Ours
Car4	179.8	2.9	9.0	60.1	18.8	12.3	3.3	229.7	4.3	16.4	37.2	3.5	3.0	2.9	3.0
Car11	63.9	2.1	33.3	43.5	25.1	27.1	4.1	78.0	2.0	1.7	1.8	1.8	2.2	1.6	1.4
Caviar	94.2	66.2	65.9	83.9	53.0	60.9	55.3	65.5	2.3	68.6	67.5	2.2	45.7	3.1	2.2
David	76.7	3.6	7.6	16.1	9.7	13.6	6.3	12.8	3.6	10.8	13.4	3.4	3.2	4.3	3.2
Deer	50.4	127.5	140.5	55.6	25.7	11.9	69.8	10.5	8.0	38.4	9.2	36.8	8.5	10.0	5.5
Faceooc1	4.6	16.3	6.3	19.2	17.6	11.1	5.3	19.9	10.8	6.8	14.1	3.2	4.7	5.3	3.9
Faceocc2	15.5	10.2	11.1	14.1	18.6	10.4	58.6	12.8	3.8	6.3	9.2	4.8	4.0	3.1	3.1
Football	16.9	42.5	48.6	6.6	11.8	4.1	14.1	11.6	18.0	12.4	6.5	10.4	33.7	7.6	6.5
Jumping	58.4	36.8	12.5	9.9	3.6	63.0	55.2	53.0	39.1	8.8	19.2	3.9	5.0	4.8	4.9
Singer1	22.0	8.5	4.6	15.2	32.7	4.1	14.5	13.7	4.8	3.1	41.2	3.8	4.7	3.5	4.0
Average	58.2	31.7	33.9	32.4	21.7	21.9	28.7	50.8	9.7	17.3	21.9	7.4	11.5	4.6	3.8

TABLE VI: Average overlap rate (%). The best two results are shown in red and blue fonts. We compare the proposed tracker with FragT [64], IVT [12], ℓ_1T [28], MIL [24], TLD [65], VTD [66], LSK [32], CT [37], ASLA [33] ℓ_1APG [60], MTT [67], SCM [68], OSPT [69] and LSST [70]. Our tracker outperforms the state-of-the-art tracking algorithms.

Sequence	FragT[64]	IVT[12]	ℓ_1T[28]	MIL[24]	TLD[65]	VTD[66]	LSK[32]	CT[37]	ASLA[33]	ℓ_1APG[60]	MTT[67]	SCM[68]	OSPT[69]	LSST[70]	Ours
Car4	0.22	0.92	0.78	0.34	0.64	0.73	0.91	0.28	0.89	0.70	0.53	0.89	0.92	0.92	0.91
Car11	0.09	0.81	0.44	0.17	0.38	0.43	0.49	0.23	0.81	0.83	0.58	0.79	0.81	0.84	0.85
Caviar	0.19	0.21	0.20	0.19	0.21	0.19	0.58	0.33	0.84	0.13	0.14	0.87	0.25	0.85	0.88
David	0.19	0.72	0.63	0.45	0.60	0.53	0.72	0.56	0.79	0.63	0.53	0.75	0.76	0.75	0.81
Deer	0.08	0.22	0.07	0.21	0.41	0.58	0.35	0.60	0.62	0.45	0.60	0.46	0.61	0.58	0.68
Faceooc1	0.90	0.85	0.89	0.59	0.65	0.77	0.90	0.74	0.83	0.87	0.79	0.93	0.91	0.89	0.91
Faceocc2	0.60	0.59	0.67	0.61	0.49	0.59	0.33	0.68	0.82	0.70	0.72	0.82	0.84	0.86	0.88
Football	0.57	0.55	0.11	0.55	0.56	0.81	0.63	0.46	0.57	0.68	0.71	0.69	0.62	0.69	0.72
Jumping	0.14	0.28	0.55	0.53	0.69	0.08	0.09	0.07	0.24	0.59	0.30	0.73	0.69	0.65	0.70
Singer1	0.34	0.66	0.70	0.33	0.41	0.79	0.52	0.34	0.81	0.83	0.32	0.85	0.82	0.80	0.85
Average	0.33	0.58	0.50	0.40	0.50	0.55	0.55	0.45	0.72	0.64	0.52	0.78	0.72	0.78	0.82

TABLE VII: Comparison between DLT [4] and our tracker on 8 sequences. The better results are shown in red fonts. "BetterCount" means the number of sequences on which the performance of the current tracker is better than the other one.

	Car4	Car11	David	Deer	Shaking	Singer1	Trellis	Woman	BetterCount
DLT	6.0	1.2	7.1	10.2	11.5	3.3	3.3	9.4	3
Ours	3.0	1.4	3.2	5.5	15.2	4.0	3.0	2.2	5

tracker (IVT) [12]. We present the performances on both the original IVT and our tracker (deepIVT) in terms of average center errors and average overlap rates in Figures 6 and 7 respectively. We can observe that our tracker (deepIVT) outperforms the original IVT in most of 12 test sequences. Due to IVT's limited performance, our tracker also misses objects in some sequences. However, the figures presented here aim to show that our learned features can boost performances of the baseline tracker. In addition, we verify our learned feature's generalizability by using ℓ_1_APG tracker [60] and evaluating performances on the same 12 sequences as used for IVT. As shown in Tables I and II, ℓ_1_APG can hardly handle these challenging sequences with complicated motion transformations. In contrast, integrating our learned features into ℓ_1_APG can succeed to track objects in 6 (David2, FleetFace, Freeman1, Freeman3, MountainBike and Sylvester) of 12 sequences. Therefore, we can conclude that our learned features are not only beneficial to ASLA [33], but also generally helpful to other trackers.

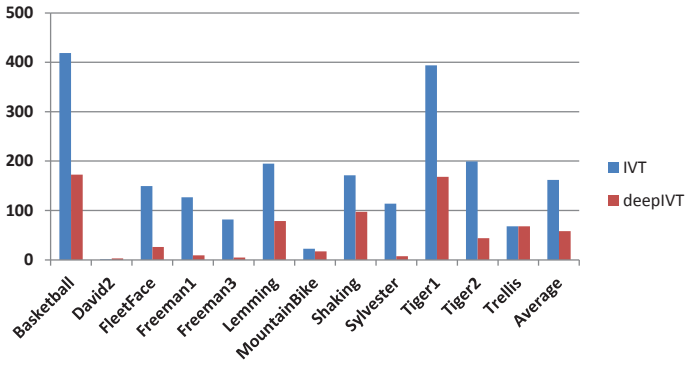


Fig. 6: Average center error (in pixels). We compare performances between the original IVT [12] and our tracker (deepIVT) using the learned features.

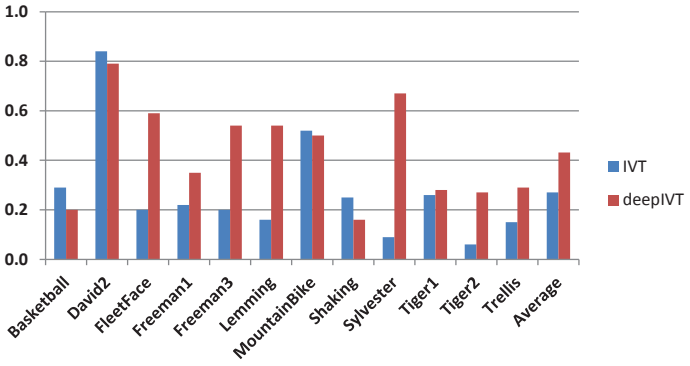


Fig. 7: Average overlap rates (%). We compare performances between the original IVT [12] and our tracker (deepIVT) using the learned features.

VI. Conclusion

In this paper, we propose a hierarchical feature learning algorithm for visual object tracking. We learn the generic features from auxiliary video sequences by using a two-layer convolutional neural network with the temporal slowness

constraint. Moreover, we propose an adaptation module to adapt the pre-learned features according to specific target objects. As a result, the adapted features are robust to both complicated motion transformations and appearance changes of specific target objects. Experimental results demonstrate that the learned hierarchical features are able to significantly improve performances of baseline trackers.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.

[2] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, 2011, pp. 3361–3368.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[4] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *NIPS*, 2013, pp. 809–817.

[5] C. F. Cadieu and B. A. Olshausen, "Learning transformational invariants from natural movies," in *NIPS*, 2008, pp. 209–216.

[6] W. Y. Zou, A. Y. Ng, S. Zhu, and K. Yu, "Deep learning of invariant features via simulated fixations in video," in *NIPS*, 2012, pp. 3212–3220.

[7] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[8] J. Lu, G. Wang, and P. Moulin, "Human identity and gender recognition from gait sequences with arbitrary walking directions," *IEEE Trans Inf. Forensics Security*, vol. 9, no. 1, pp. 51–61, 2014.

[9] B. Wang, G. Wang, K. L. Chan, and L. Wang, "Tracklet association with online target-specific metric learning," in *CVPR*, 2014, pp. 1234–1241.

[10] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Comput. Vis.*, vol. 26, no. 1, pp. 63–84, 1998.

[11] J. Ho, K.-C. Lee, M.-H. Yang, and D. J. Kriegman, "Visual tracking using learned linear subspaces," in *CVPR*, 2004, pp. 782–789.

[12] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1-3, pp. 125–141, 2008.

[13] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, 1998.

[14] A. Doucet, N. De Freitas, and N. Gordon, *An introduction to sequential Monte Carlo methods*. Springer, 2001.

[15] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.

[16] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[17] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *Int. J. Comput. Vis.*, vol. 56, no. 3, pp. 221–255, 2004.

[18] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1296–1311, 2003.

[19] G. Wang, D. Hoiem, and D. A. Forsyth, "Learning image similarity from flickr groups using fast kernel machines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2177–2188, 2012.

[20] G. Wang, D. A. Forsyth, and D. Hoiem, "Improved object categorization and detection using comparative object similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 10, pp. 2442–2453, 2013.

[21] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, 2004.

[22] ——, "Ensemble tracking," in *CVPR*, 2005, pp. 494–501.

[23] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *ECCV*, 2008, pp. 234–247.

[24] B. Babenko, M.-H. Yang, and S. J. Belongie, "Visual tracking with online multiple instance learning," in *CVPR*, 2009, pp. 983–990.

[25] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," in *CVPR*, 2010, pp. 49–56.

[26] T. Liu, G. Wang, and Q. Yang, "Real-time part-based visual tracking via adaptive correlation filters," in *CVPR*, 2015, pp. 4902–4912.

[27] J. Yang, K. Yu, Y. Gong, and T. S. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR*, 2009, pp. 1794–1801.

[28] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *ICCV*, 2009, pp. 1436–1443.

[29] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, "Minimum error bounded efficient ℓ_1 tracker with occlusion detection," in *CVPR*, 2011, pp. 1257–1264.

[30] H. Li, C. Shen, and Q. Shi, "Real-time visual tracking using compressive sensing," in *CVPR*, 2011, pp. 1305–1312.

[31] B. Liu, L. Yang, J. Huang, P. Meer, L. Gong, and C. A. Kulikowski, "Robust and fast collaborative tracking with two stage sparse optimization," in *ECCV*, 2010, pp. 624–637.

[32] B. Liu, J. Huang, L. Yang, and C. A. Kulikowski, "Robust tracking using local sparse appearance model and k-selection," in *CVPR*, 2011, pp. 1313–1320.

[33] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *CVPR*, 2012, pp. 1822–1829.

[34] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1631–1643, 2005.

[35] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC*, 2006, pp. 47–56.

[36] M. Grabner, H. Grabner, and H. Bischof, "Learning features for tracking," in *CVPR*, 2007.

[37] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *ECCV*, 2012, pp. 864–877.

[38] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[39] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[40] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video tracking using learned hierarchical features," *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, 2015.

[41] S. M. A. Eslami, N. Heess, and J. M. Winn, "The shape boltzmann machine: A strong model of object shape," in *CVPR*, 2012, pp. 406–413.

[42] A. Shahroudy, G. Wang, T.-T. Ng, and Q. Yang, "Multimodal multipart learning for action recognition in depth videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015.

[43] J. Lu, G. Wang, W. Deng, P. Moulin, and J. Zhou, "Multi-manifold deep metric learning for image set classification," in *CVPR*, 2015, pp. 1137–1145.

[44] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia, "Multi-task CNN model for attribute prediction," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1949–1959, 2015.

[45] J. Lu, V. E. Liong, G. Wang, and P. Moulin, "Joint feature learning for face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1371–1383, 2015.

[46] Z. Zuo, G. Wang, B. Shuai, L. Zhao, and Q. Yang, "Exemplar based deep discriminative and shareable feature learning for scene image classification," *Pattern Recognition*, vol. 48, no. 10, pp. 3004–3015, 2015.

[47] B. Shuai, G. Wang, Z. Zuo, B. Wang, and L. Zhao, "Integrating parametric and non-parametric models for scene labeling," in *CVPR*, 2015, pp. 4249–4258.

[48] A. Wang, J. Lu, J. Cai, G. Wang, and T. Cham, "Unsupervised joint feature learning and encoding for RGB-D scene labeling," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 4459–4473, 2015.

[49] A. Coates, A. Karpathy, and A. Y. Ng, "Emergence of object-selective features in unsupervised feature learning," in *NIPS*, 2012, pp. 2690–2698.

[50] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *NIPS*, 2011, pp. 2528–2536.

[51] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "Ica with reconstruction cost for efficient overcomplete feature learning," in *NIPS*, 2011, pp. 1017–1025.

[52] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," in *ICML*, 2012.

[53] N. Li and J. J. DiCarlo, "Unsupervised natural experience rapidly alters invariant object representation in visual cortex," *Science*, vol. 321, no. 5895, pp. 1502–1507, 2008.

[54] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *ECCV*, 2010, pp. 213–226.

[55] J. Yang, R. Yan, and A. G. Hauptmann, "Cross-domain video concept detection using adaptive svms," in *ACM Multimedia*, 2007, pp. 188–197.
[56] L. Duan, D. Xu, I. W.-H. Tsang, and J. Luo, "Visual event recognition in videos by learning from web data," in *CVPR*, 2010, pp. 1959–1966.
[57] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *ICML*, 2011, pp. 513–520.
[58] Q. Wang, F. Chen, J. Yang, W. Xu, and M.-H. Yang, "Transferring visual prior for online object tracking," *IEEE Trans. Image Process.*, vol. 21, no. 7, pp. 3296–3305, 2012.
[59] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.
[60] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *CVPR*, 2012, pp. 1830–1837.
[61] Y. Pang and H. Ling, "Finding the best from the second bests - inhibiting subjective bias in evaluation of visual tracking algorithms," in *ICCV*, 2013, pp. 2784–2791.
[62] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. 886–893.
[63] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013, pp. 2411–2418.
[64] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *CVPR*, 2006, pp. 798–805.
[65] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, 2012.
[66] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *CVPR*, 2010, pp. 1269–1276.
[67] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *CVPR*, 2012, pp. 2042–2049.
[68] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *CVPR*, 2012, pp. 1838–1845.
[69] D. Wang, H. Lu, and M.-H. Yang, "Online object tracking with sparse prototypes," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 314–325, 2013.
[70] ——, "Least soft-threshold squares tracking," in *CVPR*, 2013, pp. 2371–2378.
[71] X. Li, C. Shen, A. R. Dick, and A. van den Hengel, "Learning compact binary codes for visual tracking," in *CVPR*, 2013, pp. 2419–2426.
[72] S. He, Q. Yang, R. W. H. Lau, J. Wang, and M. Yang, "Visual tracking via locality sensitive histograms," in *CVPR*, 2013, pp. 2427–2434.