

Learning Hierarchical Features for Visual Object Tracking with Recursive Neural Networks

Li Wang, *Member, IEEE*, Ting Liu, *Student Member, IEEE*, Bing Wang, *Student Member, IEEE*,
Xulei Yang, *Member, IEEE* and Gang Wang, *Member, IEEE*

Abstract—Recently, deep learning has achieved very promising results in visual object tracking. Deep neural networks in existing tracking methods require a lot of training data to learn a large number of parameters. However, training data is not sufficient for visual object tracking as annotations of a target object are only available in the first frame of a test sequence. In this paper, we propose to learn hierarchical features for visual object tracking by using tree structure based Recursive Neural Networks (RNN), which have fewer parameters than other deep neural networks, e.g. Convolutional Neural Networks (CNN). First, we learn RNN parameters to discriminate between the target object and background in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, we create two dictionaries regarding target regions and corresponding local patches based on the learned hierarchical features from both top and leaf nodes of multiple random trees. In each of the subsequent frames, we conduct sparse dictionary coding on all candidates to select the best candidate as the new target location. In addition, we online update two dictionaries to handle appearance changes of target objects. Experimental results demonstrate that our feature learning algorithm can significantly improve tracking performance on benchmark datasets.

Index Terms—Visual object tracking, feature learning, Recursive Neural Networks

I. INTRODUCTION

VISUAL object tracking aims to locate a target object in a video sequence given its location in the first frame. It is a very challenging problem because target appearance may vary dramatically due to illumination change, partial occlusion, object deformation, etc. To solve these issues, some trackers [1] employ local patch based appearance models to achieve very promising performance.

Feature representation is very important to object tracking. Recently, deep learning based trackers [2][3][4][5][6][7] have achieved very promising performance by using learned hierarchical features rather than raw pixel values or hand-crafted features. Deep neural networks usually require a lot of training data to learn a large number of parameters. However, training data is not sufficient for visual object tracking as annotations of a target object are only available in the first frame of a test sequence.

To overcome this problem, existing feature learning based trackers pre-train their neural networks by using auxiliary data

and then fine-tune network parameters according to specific target objects. Different from these methods, we propose to learn hierarchical features for visual object tracking by using tree structure based Recursive Neural Networks (RNN), which have fewer parameters than other deep neural networks, e.g. Convolutional Neural Networks (CNN). As a result, our feature learning method does not require any network pre-training on auxiliary data and will not suffer from fine-tuning network parameters.

First, we learn RNN parameters to discriminate between target object and background in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, we create two dictionaries regarding target regions and corresponding local patches based on the learned hierarchical features from both top and leaf nodes of multiple random trees. In each of the subsequent frames, we conduct sparse dictionary coding on all candidates to select the best candidate as the new target location. In addition, we online update two dictionaries to handle appearance changes of target objects.

The main contribution is that hierarchical features are learned to discriminate between target and background by using RNN which can successfully encode spatial information among local patches of a target object based on multiple random trees. RNN features learned at top nodes of random trees are able to capture structural information of target objects, which are robust to holistic appearance changes caused by illumination change or object deformation. Moreover, RNN features learned at leaf nodes represent local patches and capture local appearance changes due to partial occlusion. Therefore, our hierarchical features learned from both top and leaf nodes are beneficial for visual object tracking. Experimental results demonstrate that using our feature learning method can significantly improve tracking performance on the benchmark dataset [8].

II. RELATED WORK

Visual Object Tracking. During the past few decades, visual object tracking have received much attention. Many tracking algorithms have achieved very promising results. We refer interested readers to some recent surveys [8][9]. Our feature learning algorithm is integrated into a baseline tracker ASLA [1] belonging to generative trackers which focus on modeling target appearance without considering background. Other generative trackers utilize subspace learning [10][11], sparse coding [12][13][14], Gaussian mixture model [15], kernel-based

L. Wang and X. Yang are with Agency for Science, Technology and Research (A*STAR), Singapore. (e-mail: wa0002li@e.ntu.edu.sg; yangxl@i2r.a-star.edu.sg)

T. Liu, B. Wang and G. Wang are with Alibaba AI Labs, China. (e-mail: liut0016@e.ntu.edu.sg; wang0775@e.ntu.edu.sg; gangwang6@gmail.com)

model [16], visual decomposition model [17], etc. Although the baseline tracker in this paper is generative, our features are learned discriminatively to differentiate target objects from background. Discriminative trackers formulate object tracking as a binary classification problem. They use many machine learning algorithms such as SVM [18][19][20][21], boosting [22][23][24], graph embedding [25], multiple instance learning [26], metric learning [27][28], Gaussian process regression [29], etc.

Deep Learning. Recently, deep learning has achieved very promising results in visual object tracking [2][3][4][5][6][7]. Usually, deep neural networks require a lot of training data. However, only the first frame of a test sequence is annotated in visual object tracking. To overcome this problem, some deep learning based trackers pre-train their neural networks by using auxiliary data, e.g. 80 million tiny images dataset [2], face detection dataset [3] and Hans van Hateren natural scene videos [4]. In contrast, our feature learning method does not require any network pre-training on auxiliary data. Some other deep learning based trackers adopt existing deep neural networks, e.g. R-CNN model built upon Caffe Library [5] and VGG network pre-trained on ImageNet [6], and then fine-tune network parameters based on training data of specific target objects. Different from these methods, we learn RNN parameters in the first frame of a test sequence and fix the parameters in the subsequent frames. Therefore, our method does not suffer from fine-tuning network parameters.

Recursive Neural Networks. RNN has been successfully applied to natural language processing for sentiment analysis [30], phrase and sentence modeling [31] and paraphrase detection [32]. Also, RNN is applied for parsing natural scene [33] and 3D object classification [34]. To our best knowledge, we are the first to learn hierarchical features by using RNN for visual object tracking.

III. LEARNING HIERARCHICAL FEATURES USING RNN

Deep learning has achieved very promising performance for visual object tracking. Some deep learning based trackers [2][3][4] require auxiliary data to pre-train their neural networks. This kind of pre-training is necessary for feature learning but inconvenient for visual object tracking. Moreover, some other trackers [5][6] employ the deep neural networks pre-trained already and then fine-tune them during tracking. However, annotation of a target object in a test sequence is still limited for fine-tuning such deep neural networks with large numbers of parameters. To avoid inconvenient pre-training and fine-tuning, we propose to learn hierarchical features for visual object tracking by using tree-structure based RNN which has fewer parameters and hence requires less training data. We learn RNN parameters to discriminate target from background by using target annotation in the first frame and fix them for extracting hierarchical features from candidate regions in subsequent frames. Moreover, our learned hierarchical features are able to capture spatial information over local patches of a target region based on multiple random trees. This capability is beneficial for visual object tracking.

In this section, we present details of our feature learning algorithm. First, we give an overview of RNN. Then, we

explain how to extract hierarchical features based on RNN. Next, we describe how to generate trees over local patches of target regions. Last, we depict how to discriminatively learn RNN parameters.

A. Overview of RNN

RNN is a deep neural network established by applying the same set of parameters recursively over a certain structure. In our case, RNN [33] is based on tree structure over local patches of a target object. There are three types of parameters: W^{raw} , W^{rnn} and W^{label} . W^{raw} and W^{rnn} are used to extract hierarchical features from candidate regions based on multiple random trees. W^{label} is to map extracted features regarding candidate regions into different classes (target object and background in this paper). In the first frame of a test sequence, we jointly learn these three types of parameters together to discriminate target from background. In the subsequent frames, we use the learned W^{raw} and W^{rnn} to extract hierarchical features for candidate regions.

B. Extracting Features Using RNN

Given a tree over local patches of a target region (see details in Section III-C) and learned RNN parameters W^{raw} and W^{rnn} (see details in Section III-D), Figure 1 illustrates an instance of extracting hierarchical features from a target region. We employ the local patch setting in ASLA [1] and decompose a target object observation $x \in \mathbb{R}^{32 \times 32}$ into 9 overlapping local patches $p \in \mathbb{R}^{16 \times 16}$. Each local patch can be vectorized into an 256-dimensional raw pixel value feature $V_i \in \mathbb{R}^{256 \times 1}$, $i = 1, \dots, 9$. Then, we utilize a neural network layer to map raw pixel values (orange circles in Figure 1) into a n -dimensional RNN feature space (blue circles). These RNN features at leaf nodes can be calculated as follows:

$$\zeta_i = f(W^{raw}V_i + b^{raw}), \quad (1)$$

where $W^{raw} \in \mathbb{R}^{n \times 256}$ is the transformation matrix, b^{raw} is the bias, f is the sigmoid function $f(x) = 1/(1 + e^{-x})$ and $\zeta_i \in \mathbb{R}^{n \times 1}$ is the RNN feature at leaf nodes.

In the given tree, each node is associated with the same basic neural network illustrated in Figure 1. The basic network computes parent features based on two child input nodes as follows:

$$\eta_{(i,j)} = f([W^{rnn}, W^{rnn}][\tau_i; \tau_j] + b^{rnn}), \quad (2)$$

where $W^{rnn} \in \mathbb{R}^{n \times n}$ is the transformation matrix, b^{rnn} is the bias, f is the sigmoid function $f(x) = 1/(1 + e^{-x})$, $[\tau_i; \tau_j] \in \mathbb{R}^{2n \times 1}$ is the concatenated feature for two child nodes and $\eta_{(i,j)} \in \mathbb{R}^{n \times 1}$ is the parent feature. Note that the child node pair $([\tau_i; \tau_j])$ possibly includes two leaf nodes $([\zeta_i; \zeta_j])$, or two non-terminal nodes $([\eta_i; \eta_j])$, or one leaf node and one non-terminal node $([\zeta_i; \eta_j])$. Given the tree structure and the RNN parameters, we can extract hierarchical features from a target region at the top node of the RNN tree by recursively using the same basic neural network in a bottom-up manner. As a result, learned hierarchical features can capture spatial information over local patches of the target object.

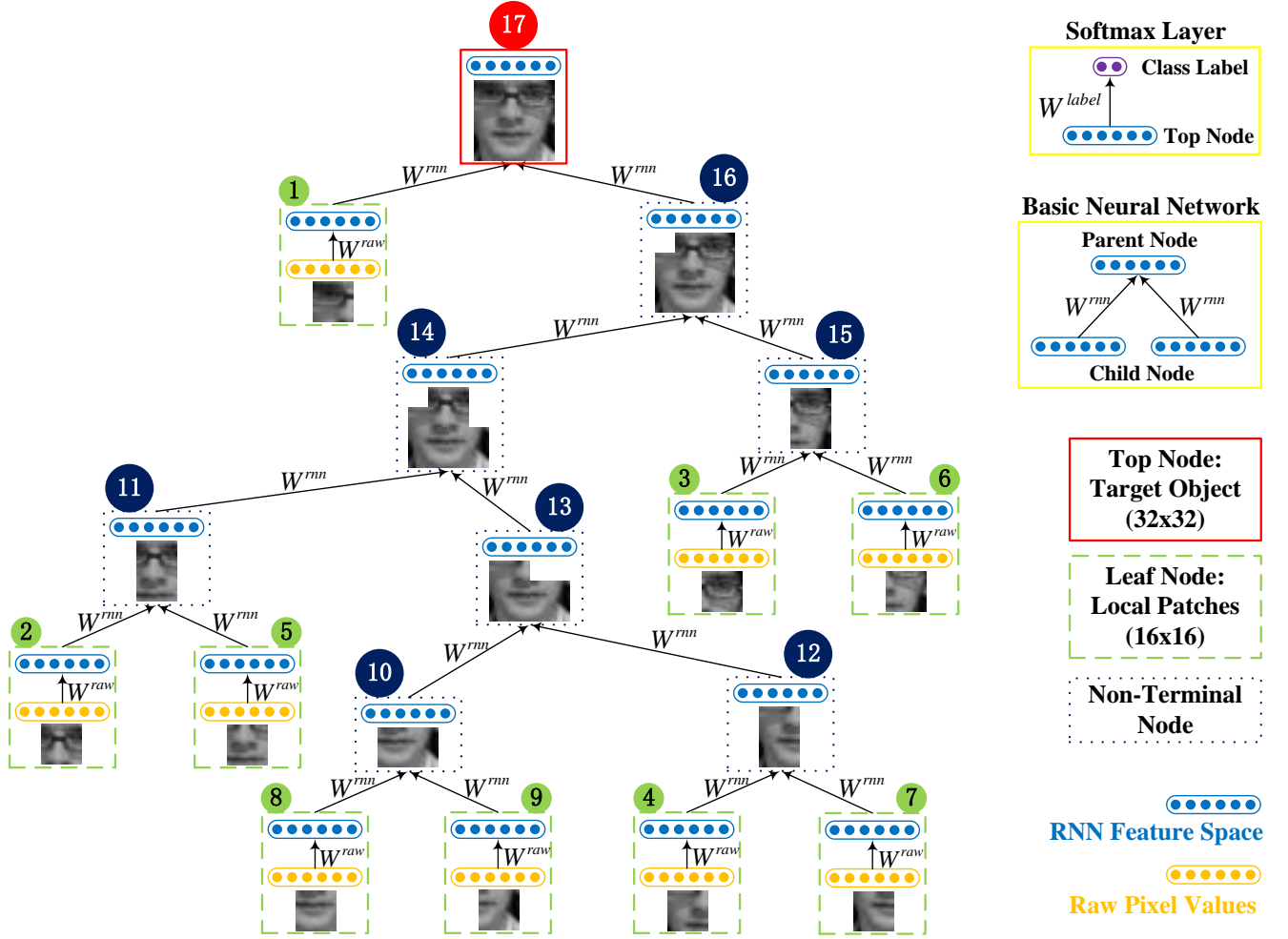


Fig. 1: Illustration of extracting hierarchical features using RNN with a known tree structure over the local patches of a target object.

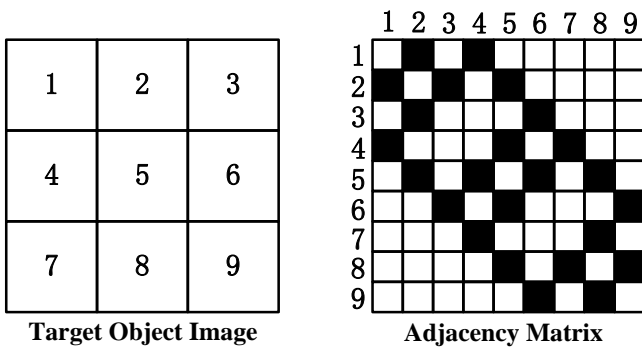


Fig. 2: Illustration of the spatial layout of local patches in a target object and the corresponding adjacency matrix. For example, patch 1 and 2 can be merged as they are spatially neighboring, so the corresponding matrix element is 1.

C. Generating Random Trees

To extract hierarchical features using RNN, we need to generate a tree structure over local patches of a candidate region. There is no objective criterion to identify the best

tree. Therefore, we randomly generate trees by using a greedy bottom-up searching strategy. We employ a number of trees together to extract RNN features, which are expected to be robust to certain noisy tree structure.

We first define an adjacency matrix A , where $A(i, j) = 1$ if local patch i and j are spatially neighboring. It means that local patch i and j can be merged during generating random trees. Figure 2 illustrates the spatial layout of local patches in a target object and the corresponding adjacency matrix, which is fixed in this paper.

Given the adjacency matrix A , we can find the pairs of neighboring local patches and denote the set of these potential child node pairs as follows:

$$C = \{[p_i, p_j] | A(i, j) = 1\}, \quad (3)$$

where p_i indicates the i th local patch. At the beginning (see Figure 2), we have the following pairs: $\{[p_1, p_2], [p_1, p_4], \dots, [p_9, p_6], [p_9, p_8]\}$. Suppose we randomly select the pair $[p_i, p_j]$, we remove all pairs with either p_i and p_j from the set C as follows:

$$C = C - \{[p_i, p_m] | m \in Neigh(i)\} - \{[p_j, p_n] | n \in Neigh(j)\}, \quad (4)$$

where $Neigh()$ denotes the neighborhood of a local patch. Then, we add new child pairs to the set C as follows:

$$C = C \cup \{[p_{(i,j)}, p_k] | k \in Neigh(i) \cup Neigh(j), k \neq i, j\}, \quad (5)$$

where we randomly select another pair to merge. In the same way, we update the set C and then repeat the previous steps until only one child node pair is left in the set C . Finally, we can achieve the top node of the tree over all local patches of a target region by merging the last child node pair in set C .

We use the RNN tree shown in Figure 1 as an example. Circles on top of nodes indicate the index numbers of nodes. We can observe that local patch 8 and 9 merge first and the corresponding child node pair $[p_8, p_9]$ is selected. Consequently, we update $C = \{[p_1, p_2], [p_1, p_4], [p_2, p_1], [p_2, p_3], [p_2, p_5], [p_3, p_2], [p_3, p_6], [p_4, p_1], [p_4, p_5], [p_4, p_7], [p_5, p_2], [p_5, p_4], [p_5, p_6], [p_6, p_3], [p_6, p_5], [p_7, p_4], [p_{(8,9)}, p_5], [p_{(8,9)}, p_6], [p_{(8,9)}, p_7], [p_5, p_{(8,9)}], [p_6, p_{(8,9)}], [p_7, p_{(8,9)}]\}$. Next, we randomly select another pair in the updated C . In this example, local patch 2 and 5 are subsequently merged. Then, we repeat the previous manipulations until only one child node pair is left in the set C . As illustrated in Figure 1, we finally obtain the tree over local patches of the target region after merging local patch 1 and non-terminal node 16.

D. Discriminative Learning of RNN Parameters

Previously, deep learning based tracking methods require pre-training from auxiliary data or fine-tuning neural networks pre-trained already according to specific target object. In contrast, we learn RNN parameters only based on target annotation and background data in the first frame of a test sequence. Then, we fix the learned RNN parameters W^{raw} and W^{rnn} for extracting hierarchical features from candidate regions in the subsequent frames.

To discriminatively learn RNN parameters, we need to add a softmax layer with the parameter W^{label} , which connects RNN features at top nodes and class labels. Suppose we have N training samples $\mathbf{X} = \{x_1, \dots, x_N\}$ in terms of target and background, which are respectively sampled nearby and away from the target location in the first frame. We also have the corresponding labels $\mathbf{L} = \{l_1^{GT}, \dots, l_N^{GT}\}$, $l_i^{GT} \in \{[1; 0], [0; 1]\}$ ($[1; 0]$ and $[0; 1]$ indicate target and background respectively in our implementation). For each training sample x_i , RNN features can be extracted at the top node of a random tree based on the parameters W^{raw} and W^{rnn} . Then, we apply the softmax layer illustrated in Figure 1 for predicting class label:

$$l_i^{RNN} = \text{softmax}(W^{label}(\eta_{top}|T_i)), \quad (6)$$

where $W^{label} \in \mathbb{R}^{2 \times n}$ and η_{top} is the learned RNN features at the top node of a random tree $T_i \in \mathcal{T}(x_i)$. $\mathcal{T}(x_i)$ is all possible trees over local patches of x_i . Consequently, we can compute the loss between the predicted label and the ground truth label for each training sample by using the cross-entropy error as follows:

$$\delta(x_i, l_i^{GT} | \theta, T_i) = - \langle l_i^{GT}, \log(l_i^{RNN}) \rangle, \quad (7)$$

where $\theta = \{W^{raw}, W^{rnn}, W^{label}\}$. Finally, we have the objective function over all training samples as follows:

$$\Phi(\theta) = \frac{1}{N} \sum_{i=1}^N \delta(\theta) + \frac{\lambda}{2} \|\theta\|^2 \quad (8)$$

where λ is the regularization parameter. The gradient of our objective function of Equation 8 w.r.t. the parameter set θ can be computed as follows:

$$\frac{\partial \Phi}{\partial \theta} = \frac{1}{N} \sum_i \frac{\partial \delta(\theta)}{\partial \theta} + \lambda \theta, \quad (9)$$

The optimization can be performed by using backpropagation through structure [35], which splits error messages at each node and then propagates to the child nodes. Then, we employ L-BFGS [36] to optimize our objective function.

IV. OUR TRACKING SYSTEM

Although the softmax layer of RNN can classify a candidate into target or background, it is relatively weak compared to the classifiers in state-of-the-art discriminative trackers. Therefore, we use the softmax layer to learn RNN parameters only and then integrate learned features into a state-of-the-art tracker ASLA [1] with an adaptive local sparse appearance model. Interested readers may refer to ASLA [1] for more details.

Suppose we have an observation set of target $x_{1:t} = \{x_1, \dots, x_t\}$ up to the t^{th} frame and a corresponding feature representation set $z_{1:t} = \{z_1, \dots, z_t\}$, we can calculate the target state y_t as follows:

$$y_t = \arg \max_{y_t^i} p(y_t^i | z_{1:t}), \quad (10)$$

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

$$p(y_t | z_{1:t}) \propto p(z_t | y_t) \int p(y_t | y_{t-1}) p(y_{t-1} | z_{1:t-1}) dy_{t-1}, \quad (11)$$

where $z_{1:t}$ denotes the feature representation, $p(y_t | y_{t-1})$ denotes the motion model and $p(z_t | y_t)$ denotes the appearance model. In ASLA [1], the representations $z_{1:t}$ simply use raw pixel values of local patches, and the appearance model $p(z_t | y_t)$ employs sparse coding. In our tracker, we learn hierarchical features from raw pixel values by using RNN. First, we learn RNN parameters by using 20 positive (nearby target) and 100 negative (background) samples in the first frame of a test sequence. Then, we create two dictionaries regarding target regions and corresponding local patches based on learned RNN features respectively from top and leaf nodes of multiple random trees in the first 10 frames. In each of subsequent frames, we conduct sparse coding on all candidate regions in terms of two dictionaries regarding target regions and local patches respectively. In addition, we online update two dictionaries every 5 frames. In our implementations, we fix the dimension size of RNN feature space $n = 50$, the regularization parameter $\lambda = 0.0001$ and the motion parameters at $[10, 10, 0.01, 0, 0.005, 0]$ for all test sequences. We summarize our tracker using learned RNN features in Algorithm 1.

Algorithm 1 Our Tracker Using Learned RNN Features

- 1: **Input:** the last target state y_{t-1} , the RNN parameters θ learned in the first frame and two existing dictionaries respectively regarding target regions and local patches.
 - 2: Generate 600 target state candidates y_t^i and the corresponding image observations x_t^i nearby the previous target state y_{t-1} .
 - 3: Extract hierarchical features z_t^i from each image observation x_t^i by using the RNN parameter θ learned in the first frame.
 - 4: Calculate the posterior probability $p(y_t^i | z_{1:t})$ according to Equation 11.
 - 5: Predict the target state by $y_t = \arg \max_{y_t^i} p(y_t^i | z_{1:t})$.
 - 6: Update two dictionaries every 5 frames with learned RNN features respectively from top and leaf nodes of multiple random trees over the target region regarding the predicted state y_t .
 - 7: **Output:** the predicted target state y_t and two updated dictionaries.
-

V. DISCUSSION

Parameter Size. As sharing parameters in different layers of tree structures, RNN has fewer parameters than other deep neural networks, e.g. Convolutional Neural Networks (CNN). The parameter set for RNN is $\theta = \{W^{raw}, W^{rnn}, W^{label}\}$, where $W^{raw} \in \mathbb{R}^{50 \times 256}$, $W^{rnn} \in \mathbb{R}^{50 \times 50}$ and $W^{label} \in \mathbb{R}^{2 \times 50}$. We can find that our RNN model has totally 15400 parameters, which are much less than CNN (e.g. 60 million parameters in [37]). Therefore, our feature learning algorithm using RNN does not require any pre-training and fine-tuning, which are actually not easy for visual tracking applications.

Computational Cost. We run experiments on a PC without using any multi-core setting. The baseline tracker ASLA [1] can achieve about 3 fps. Our tracker using learned hierarchical features can achieve about 1.5 fps. We have two major factors affecting efficiencies of our tracker: i) Learning RNN parameters in the first frame of a test sequence; ii) Extracting hierarchical features from candidate regions based on multiple random trees in each frame of the subsequent frames. As extracting features from 600 candidates consumes too much time, we use a coarse-to-fine strategy which first identifies 20 promising candidates according to tracking results from the baseline tracker ASLA [1] and then ranks these 20 candidates based on our learned hierarchical features. Anyway, the main objective in this paper is to show that our learned hierarchical features can improve tracking performance. The efficiency of our tracker could be enhanced by using parallel programming skills (e.g. parfor in MATLAB) or other advanced hardware (e.g. GPU).

VI. EXPERIMENTS

Benchmark. We evaluate tracking performance on a recent public benchmark [8] containing 50 sequences which covers almost all challenging issues such as illumination changes, pose variations, occlusion, in/out-of-plane motions and cluttered background. The benchmark dataset reports the results

from 29 trackers. Here, we compare our tracker “RNN” with the top 6 trackers in the benchmark: ASLA [1], CXT [38], SCM [44], Struck [20], TLD [42] and VTS [43]. In addition, we also compare our tracker with 3 recent state-of-the-art trackers: KCF [39], MEEM [40] and TGPR [29], which have reported their results on the benchmark.

The benchmark uses two measurements: i) Precision vs. Location error threshold, the percentage of the frames in which distances between tracking results and ground truth are below certain thresholds. ii) Success rate vs. Overlap threshold, the percentage of the frames in which overlapping percentages of tracking results against ground truth are higher than certain thresholds. We rank different trackers according to location error thresholding at 20 pixels for precision and Area Under Curve (AUC) for success rate. In addition, we use the one-pass evaluation (OPE) setting in the benchmark.

Quantitative results. The precision plot and the success plot of tracking results from the top 10 trackers on all sequences of the benchmark [8] are presented in Figure 3. We can find that our tracker outperforms the baseline ASLA [1] in terms of both precision and success rate with large margins. We owe this significant improvement to our learned hierarchical features using tree structure based RNN which successfully encodes spatial information over local patches of target objects. Also, we find that our tracker using RNN features can achieve comparable performance against the state-of-the-art trackers.

To further evaluate tracking performance, in Table I and Table II, we present the comparison results from the top 10 trackers in terms of 11 attributes mentioned in the benchmark [8]. We can find that our tracker using learned RNN features can consistently improve the baseline ASLA [1] in terms of handling variational tracking challenges. Also, we can observe that our tracker achieves comparable performance against the state-of-the-art trackers in all attributes.

To investigate the effects of the numbers of random trees used in our RNN model, we present in Figure 4 the comparison results from different variants of our tracker in terms of both precision and success plots. We can find that increasing the number of random trees can significantly enhance tracking performance. It is because that learned hierarchical features based on more tree structures can encode more spatial information over local patches of target objects.

Qualitative results. We present some qualitative results from MEEM [40], TGPR [29], KCF [39], Struck [20] and our tracker using RNN features on 10 sequences (car4, soccer, ironman, singer1, jogging-1, motorRolling, walking2, freeman4, carScale, tiger1) in Figure 5, from which we can observe that our tracker can handle a large variety of tracking challenges, e.g. scale variation, occlusion, motion blur, illumination variation and in-plane rotation. In particular, our tracker can achieve very promising performance in handling scale variation (car4, singer1, walking2, carScale). It is because that: i) the baseline tracker can generate candidates with different scales; ii) our learned hierarchical features encode spatial information over local patches of target regions; iii) RNN features are learned to discriminate target from background. The later two factors further enhance the first factor for

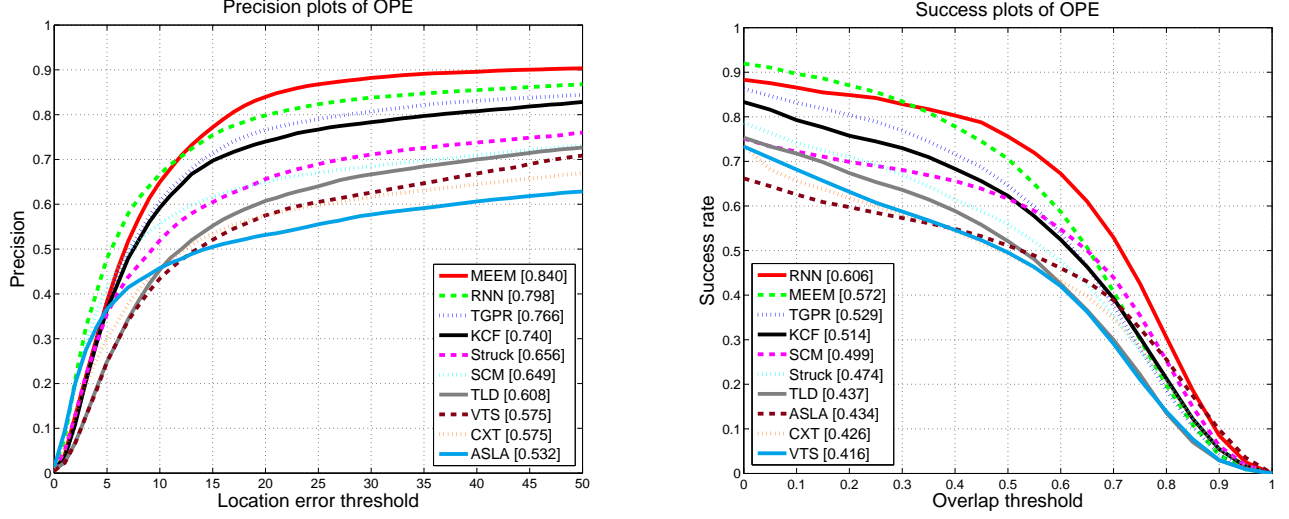


Fig. 3: Precision and success plots of the tracking results from the top 10 trackers on all sequences of the benchmark [8].

TABLE I: Average precision scores on different attributes: background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). The best and the second best results are in red and blue respectively.

	ASLA[1]	CXT[38]	KCF[39]	MEEM[40]	SCM[41]	Struck[20]	TGPR[29]	TLD[42]	VTS[43]	RNN
BC	0.496	0.443	0.753	0.808	0.578	0.585	0.761	0.428	0.578	0.779
DEF	0.445	0.422	0.740	0.859	0.586	0.521	0.768	0.512	0.487	0.762
FM	0.253	0.515	0.602	0.757	0.333	0.604	0.575	0.551	0.353	0.624
IPR	0.511	0.610	0.725	0.809	0.597	0.617	0.706	0.584	0.579	0.766
IV	0.517	0.501	0.728	0.778	0.594	0.558	0.687	0.537	0.573	0.726
LR	0.156	0.371	0.381	0.494	0.305	0.545	0.539	0.349	0.187	0.660
MB	0.278	0.509	0.650	0.740	0.339	0.551	0.578	0.518	0.375	0.623
OCC	0.460	0.491	0.749	0.814	0.640	0.564	0.708	0.563	0.534	0.743
OPR	0.518	0.574	0.729	0.853	0.618	0.597	0.741	0.596	0.604	0.780
OV	0.333	0.510	0.650	0.730	0.429	0.539	0.495	0.576	0.455	0.556
SV	0.552	0.550	0.679	0.808	0.672	0.639	0.703	0.606	0.582	0.750
Average	0.532	0.575	0.740	0.840	0.649	0.656	0.766	0.608	0.575	0.798

TABLE II: Average success rates on different attributes: background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). The best and the second best results are in red and blue respectively.

	ASLA[1]	CXT[38]	KCF[39]	MEEM[40]	SCM[41]	Struck[20]	TGPR[29]	TLD[42]	VTS[43]	RNN
BC	0.408	0.338	0.535	0.578	0.450	0.458	0.543	0.345	0.428	0.587
DEF	0.372	0.324	0.534	0.582	0.448	0.393	0.556	0.378	0.368	0.570
FM	0.247	0.388	0.459	0.568	0.296	0.462	0.441	0.417	0.300	0.482
IPR	0.425	0.452	0.497	0.535	0.458	0.444	0.487	0.416	0.416	0.576
IV	0.429	0.368	0.493	0.548	0.473	0.428	0.486	0.399	0.429	0.563
LR	0.157	0.312	0.312	0.367	0.279	0.372	0.351	0.309	0.168	0.520
MB	0.258	0.369	0.497	0.565	0.298	0.433	0.440	0.404	0.304	0.485
OCC	0.376	0.372	0.514	0.563	0.487	0.413	0.494	0.402	0.398	0.565
OPR	0.422	0.418	0.495	0.569	0.470	0.432	0.507	0.420	0.425	0.579
OV	0.312	0.427	0.550	0.597	0.361	0.459	0.431	0.457	0.443	0.477
SV	0.452	0.389	0.427	0.510	0.518	0.425	0.443	0.421	0.400	0.573
Average	0.434	0.426	0.514	0.572	0.499	0.474	0.529	0.437	0.416	0.606

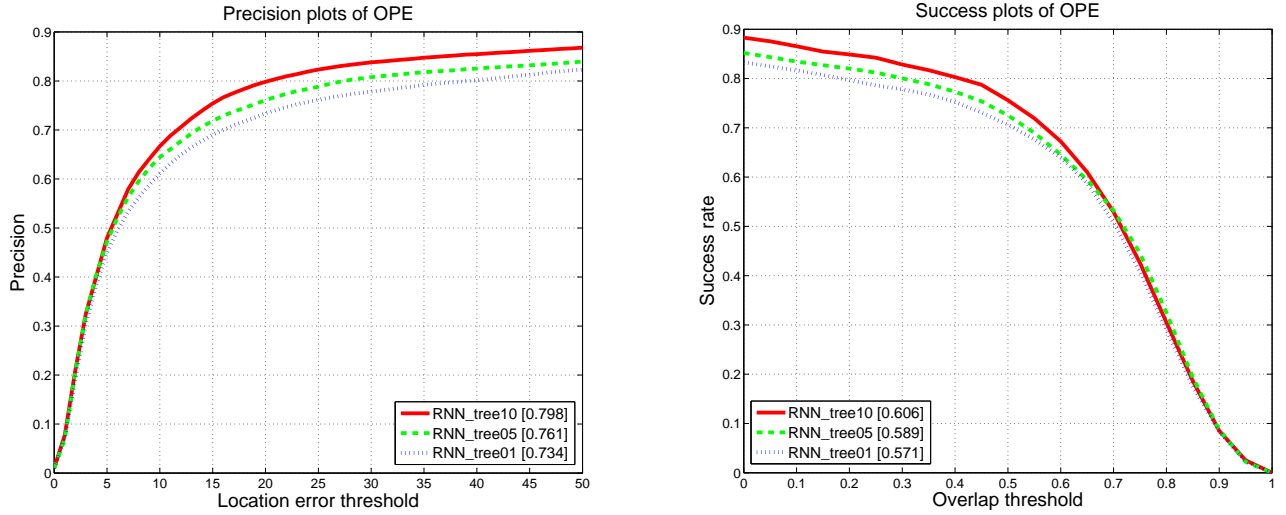


Fig. 4: Precision and success plots of the tracking results from different variants of our tracker on all sequences of the benchmark [8].

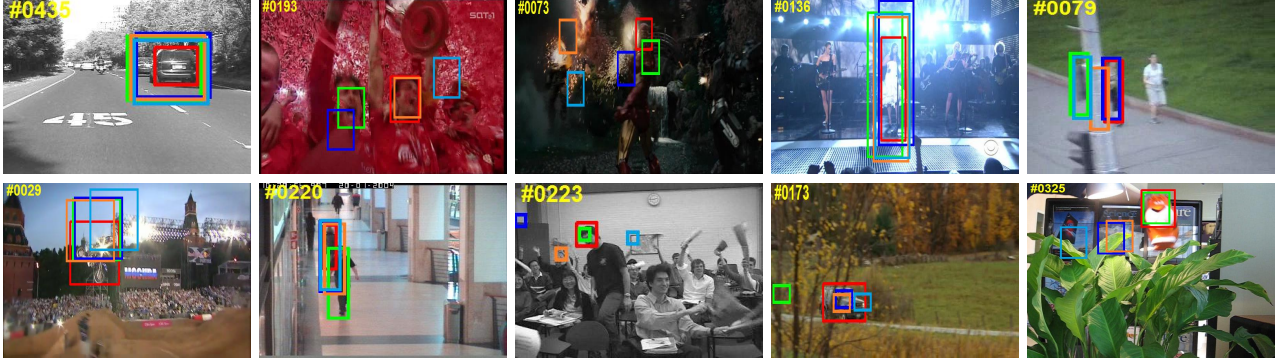


Fig. 5: Qualitative results comparing our tracker (red) with the other 4 state-of-the-art trackers: MEEM [40] (Green), TGPR [29] (dark blue), KCF [39] (orange), Struck [20] (light blue) on 10 sequences (car4, soccer, ironman, singer1, jogging-1, motorRolling, walking2, freeman4, carScale, tiger1).

handling scale changes. As a result, the bounding boxes of our tracking results can be accurately fit to target regions.

VII. CONCLUSIONS

In this paper, we propose to learn hierarchical features for visual object tracking by using tree structure based RNN which can encode spatial information over local patches of target objects. For tree structures, we use a bottom-up greedy searching strategy to generate random trees. Given a test sequence, we learn RNN parameters to discriminate target from background in the first frame and then fix them for extracting RNN features from candidate regions in the subsequent frames. With learned RNN parameters, we create two dictionaries regarding target regions and corresponding local patches based on learned RNN features respectively from top and leaf nodes of multiple random trees in the first 10 frames. In each of the subsequent frames, we conduct sparse dictionary coding on all candidate regions to find the optimal candidate as the new target location of the current frame. Additionally, we online update two

dictionaries to handle target appearance changes. Experimental results demonstrate that our feature learning algorithm can significantly improve tracking performance.

REFERENCES

- [1] X. Jia, H. Lu, and M. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1822–1829.
- [2] N. Wang and D. Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in Neural Information Processing Systems*, 2013, pp. 809–817.
- [3] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking," in *British Machine Vision Conference*, 2014.
- [4] 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.
- [5] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International Conference on Machine Learning*, 2015, pp. 597–606.
- [6] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *IEEE International Conference on Computer Vision*, 2015.

- [7] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *IEEE International Conference on Computer Vision*, 2015.
- [8] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.
- [9] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, 2014.
- [10] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.
- [11] D. A. Ross, J. Lim, R. Lin, and M. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1–3, pp. 125–141, 2008.
- [12] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *IEEE International Conference on Computer Vision*, 2009, pp. 1436–1443.
- [13] H. Li, C. Shen, and Q. Shi, "Real-time visual tracking using compressive sensing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1305–1312.
- [14] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1830–1837.
- [15] S. J. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image Vision Comput.*, vol. 17, no. 3–4, pp. 225–231, 1999.
- [16] 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.
- [17] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1269–1276.
- [18] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, 2004.
- [19] F. Tang, S. Brennan, Q. Zhao, and H. Tao, "Co-tracking using semi-supervised support vector machines," in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.
- [20] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *IEEE International Conference on Computer Vision*, 2011, pp. 263–270.
- [21] Y. Bai and M. Tang, "Robust tracking via weakly supervised ranking SVM," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1854–1861.
- [22] H. Grabner and H. Bischof, "On-line boosting and vision," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 260–267.
- [23] R. Liu, J. Cheng, and H. Lu, "A robust boosting tracker with minimum error bound in a co-training framework," in *IEEE International Conference on Computer Vision*, 2009, pp. 1459–1466.
- [24] B. Zeisl, C. Leistner, A. Saffari, and H. Bischof, "On-line semi-supervised multiple-instance boosting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, p. 1879.
- [25] X. Zhang, W. Hu, S. J. Maybank, and X. Li, "Graph based discriminative learning for robust and efficient object tracking," in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.
- [26] B. Babenko, M. Yang, and S. J. Belongie, "Visual tracking with online multiple instance learning," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009, pp. 983–990.
- [27] X. Wang, G. Hua, and T. X. Han, "Discriminative tracking by metric learning," in *European Conference on Computer Vision*, 2010, pp. 200–214.
- [28] N. Jiang, W. Liu, and Y. Wu, "Order determination and sparsity-regularized metric learning adaptive visual tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1956–1963.
- [29] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with gaussian processes regression," in *European Conference on Computer Vision*, 2014, pp. 188–203.
- [30] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 151–161.
- [31] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 1201–1211.
- [32] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems*, 2011, pp. 801–809.
- [33] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of International Conference on Machine Learning*, 2011, pp. 129–136.
- [34] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 665–673.
- [35] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in *IEEE International Conference on Neural Networks*, vol. 1, 1996, pp. 347–352.
- [36] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.
- [37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1106–1114.
- [38] T. B. Dinh, N. Vo, and G. G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1177–1184.
- [39] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, 2015.
- [40] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *European Conference on Computer Vision*, 2014, pp. 188–203.
- [41] W. Zhong, H. Lu, and M. Yang, "Robust object tracking via sparsity-based collaborative model," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1838–1845.
- [42] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N learning: Bootstrapping binary classifiers by structural constraints," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 49–56.
- [43] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *IEEE International Conference on Computer Vision*, 2011, pp. 1195–1202.
- [44] W. Zhong, H. Lu, and M. Yang, "Robust object tracking via sparse collaborative appearance model," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.