SANet: Structure-Aware Network for Visual Tracking

Heng Fan Haibin Ling
[1]Department of Computer and Information Sciences, Temple University, Philadelphia, USA
[2]HiScene Information Technologies, Shanghai, China
{hengfan, hbling}@temple.edu

Abstract

Convolutional neural network (CNN) has drawn increasing interest in visual tracking owing to its powerfulness in feature extraction. Most existing CNN-based trackers treat tracking as a classification problem. However, these trackers are sensitive to similar distractors because their CNN models mainly focus on inter-class classification. To address this problem, we use self-structure information of object to distinguish it from distractors. Specifically, we utilize recurrent neural network (RNN) to model object structure, and incorporate it into CNN to improve its robustness to similar distractors. Considering that convolutional layers in different levels characterize the object from different perspectives, we use multiple RNNs to model object structure in different levels respectively. Extensive experiments on three benchmarks, OTB100, TC-128 and VOT2015, show that the proposed algorithm outperforms other methods. Code is released at www.dabi.temple.edu/~hbling/code/SANet/SANet.html.

1. Introduction

Object tracking is one of the most important components in computer vision and has a variety of applications such as video surveillance, robotics, human-computer interaction and so forth [52]. Despite great progress in recent decades, visual tracking remains a challenging task due to appearance changes caused by deformation, illumination variations, occlusion and so on.

The deep neural networks [29], which demonstrate the powerfulness in extracting high-level feature representations [16], have drawn extensive attention in computer vision, such as image classification [27], recognition [40], saliency detection [48], semantic segmentation [33] and so on. Inspired by this, many CNN-based trackers [8, 14, 21, 30, 34, 36, 42, 47] have been proposed. Among them, [36] presents an on-line tracking method based on a multi-domain CNN architecture and achieves state-of-the-art performances on various benchmarks. By leveraging extensive annotated videos, it learns a robust shared representation to classify object from background. However, this tracker may be sensitive to similar distractors because the learned CNN model mainly focuses on inter-class classification. In the presence of distractors, the tracker has a high chance to misclassify the object and background.

Recently, recurrent neural networks (RNNs) [10], which show great success in neural language process (NLP) [17], have been brought to the computer vision community [3, 5, 11, 39, 43, 57, 58] owing to the capability of capturing long-range dependencies among sequential data. With this property, RNNs are able to model the self-structure of object.

Inspired by the above observations, in this paper we propose a novel Structure-Aware Network (SANet) architecture for visual tracking by utilizing RNNs to model self-structure of object. Different from conventional CNNs in tracking, which mainly pay attention to inter-class classification and thus are prone to drift in presence of similar distractors, our SANet leverages RNNs to encode self-structure of object during learning, which helps improve our model in discriminating not only background objects of inter-class but also similar distractors of intra-class. Because when similar distractors occur, our model is able to capture even slight difference between the reference and distractors, and use the discrepancies to distinguish object from distractors. Taking into account that convolutional layers at different levels characterize the object from different perspectives, we apply multiple RNNs to modeling structure of object in different levels respectively, which strengthens robustness of the proposed model. Besides, to supply our SANet with richer information, we adopt a skip concatenation strategy to fuse CNN and RNN feature maps, and demonstrate its effectiveness in improving performance. Figure 1 illustrates the proposed method in this paper. Extensive experimental results on two large-scale tracking benchmarks demonstrate the advances of our method.

In summary, we make the following contributions:

- We propose the structure-aware network architecture for tracking by using RNNs to encode self-structure

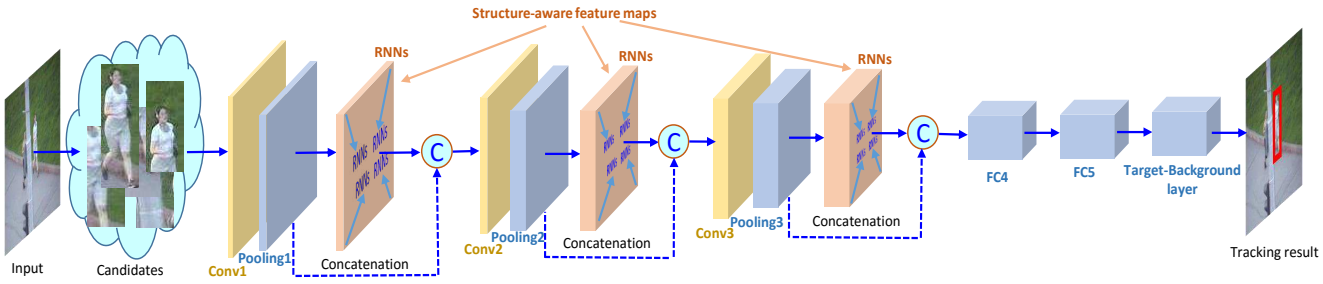

Figure 1. Illustration of the proposed SANet for visual tracking.

of object during learning, which helps our model improve not only the capability of discriminating background objects of inter-class but also similar distractors of intra-class.

- To supply our networks with richer information, we adopt a skip concatenation strategy to fuse CNN and RNN features, and show its effectiveness in improving tracking performance.

- Extensive experiments on three large-scale tracking benchmarks, OTB100 [51], TC-128 [32] and VOT2015 [24], demonstrate that the proposed tracker outperforms other state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 briefly summarizes related work. Section 3 illustrates self-structure modeling of object with RNNs. Section 4 introduces the proposed tracking algorithm in details. Experiments are described in Section 5, followed by conclusion in Section 6.

2. Related Work

Object tracking is one of the most challenging problems in computer vision and has been extensively studied [52]. In the following we highlight three lines of works which are most related to ours.

Visual tracking: Roughly speaking, tracking algorithms can be categorized into two types: discriminative methods [1, 7, 20, 23, 50, 54] and generative methods [2, 12, 13, 28, 35, 38, 46, 55]. Discriminative methods regard tracking as a classification problem which aims to separate object from ever-changing background. These methods employ both the foreground and background information to learn classifiers via P-N learning [23], multiple instance learning (MIL) [1], correlation filters [7, 20] and so forth. On the contrary, generative approaches formulate the tracking problem as searching for regions most similar to the target object. These methods are based on either subspace models or templates and update appearance model dynamically. Some representative generative methods includes incremental subspace learning [38], sparse representation [2, 35, 55], probabilistic model [28, 46] and so on.

Despite promising results for tracking in some constrained situations, the performances of aforementioned approaches are vulnerable due to the limitation of low-level hand-crafted features in complex environments where object appearances are simultaneously affected by various factors (e.g., motion blur, occlusion, deformation, scale changes, illumination variations). One possible solution is to adopt the learned high-level features for object appearance representation.

Deep networks in tracking: Owing to the powerfulness in feature extraction, deep networks haven been introduced into visual tracking. [14] proposes a human-tracking method based on CNNs. [49] introduces a deep compact tracker based on stacked autoencoder. [30] presents an online learning method based on a pool of CNNs. However, these trackers suffer from lack of enough training data to learn a robust representation, which degrades the performance of tracker. To address this problem, [8, 21, 34, 47] transfer CNNs pretrained on a large-scale dataset for image classification, however, the representation may not be very effective due to the fundamental difference between classification and tracking tasks [36]. To deal with this issue, [36, 42] propose to train the CNNs on a set of annotated video sequences, and showed that the CNNs trained on video sequences are more robust. In particular, [36] introduces an effective strategy, i.e., multi-domain learning [9], to train the CNNs, which helps to discriminate object from background. However, this method is sensitive to similar distractors because its CNN model mainly concentrate on inter-class classification. Different from [36], we use RNNs to model self-structure of object and encode it into CNN, which is beneficial to distinguish distractors of intra-class.

RNNs on image processing: RNNs [10] have been first introduced to handle sequential prediction task [17], and then extended to multi-dimensional image processing tasks [18] such as image classification [58], scene labeling [3, 39], person re-identification [44] and so on. By capturing long-

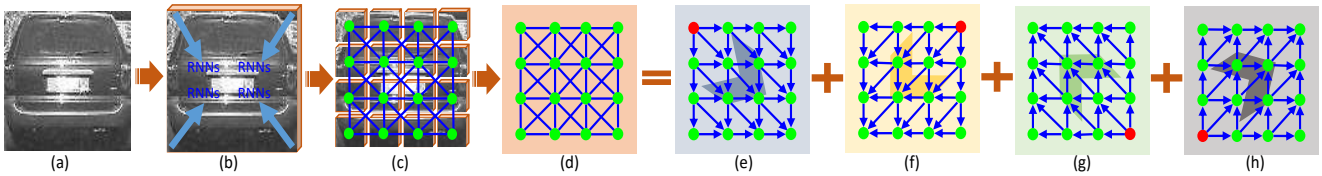

Figure 2. Decomposition of undirected cyclic graph into four directed acyclic graphs. Images (a) and (b) are inputs. Self-structure of object is encoded in an undirected cyclic graph in images (c) and (d). Images (e), (f), (g) and (h) are four directed acyclic graphs along southeast, southwest, northwest and northeast directions.

range dependencies among image units, RNNs are able to well model self-structure of object.

For visual tracking, one major challenge is to separate object from similar distractors of intra-class. CNNs cannot well deal with this situation because CNNs mainly focus on classifying objects belonging to different classes. One possible solution is to leverage the differences between intra-class objects to separate them. To make CNNs aware of the difference between objects of intra-class, we utilize RNNs to model self-structure of object and encode it into CNNs for classification. With this structural information, it is able to discriminate object from similar distractors via their even slight discrepancies.

We note that RNNs have been investigated in [5] for tracking, but it is different from ours. In [5], RNNs are used to model spatial-relationship between object and surrounding background, and obtain a confidence map to regularize correlation filters. However, in our work, we apply RNNs to modeling structure of object itself, and use such structure information to discriminate distractors of intra-class. Besides, the RNNs in this work are integrated with CNNs, and trained with enough video sequences. While in [5], RNNs are only trained with a few initial frames, and updated with each frame, which may not fully explore the advantages of RNNs.

3. RNNs for Object Self-Structure Modeling

RNNs [10] are developed for modeling dependencies in sequential data. Given an input sequence $\{x^{(t)}\}_{t=1,2,\cdots,T}$ of length T, the hidden layer $h^{(t)}$ and output layer $y^{(t)}$ at each time step t are calculated with

$$\begin{cases} h^{(t)} = \phi(Ux^{(t)} + Wh^{(t-1)} + b) \\ y^{(t)} = \sigma(Vh_t + c) \end{cases} \tag{1}$$

where U, W and V represent weight matrices between the input and the hidden layer, the previous hidden layer and the current hidden layer, and the hidden layer and the output layer respectively; b and c represent bias terms; and $\phi(\cdot)$ and $\sigma(\cdot)$ are non-linear activation functions. Since the inputs are progressively stored in hidden layers, RNNs can model long-range contextual dependencies among the sequence elements.

For two-dimensional image data, different from one-dimension sequential data, its self-structure is encoded in an undirected cyclic graph (see Figure 2(c)). Because of the loopy structure of undirected cyclic graph, the aforementioned RNNs cannot be directly applied to images. To handle this issue, we approximate the topology of an undirected cyclic graph by the combination of several directed acyclic graphs as in [39], and use variant RNNs to model self-structure of the target object as shown in Figure 2.

Assume that a directed acyclic graph is represented with $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i\}_{i=1,2,\cdots,N}$ denotes vertex set and $\mathcal{E} = \{e_{ij}\}$ is the edge set, in which e_{ij} represents a directed edge from v_i to v_j. The structure of RNNs follows the same topology as $\mathcal{G}$. A forward propagation sequence can be seen as traversing $\mathcal{G}$ from the start point, and each vertex relies on its all predecessors. For vertex v_i, therefore, the hidden layer $h^{(v_i)}$ is expressed as a non-linear function over current input $x^{(v_i)}$ at v_i and summation of hidden layers of all its predecessors. Specifically, the hidden layer $h^{(v_i)}$ and output layer $y^{(v_i)}$ at each v_i are computed with

$$\begin{cases} h^{(v_i)} = \phi(Ux^{(v_i)} + W \sum\limits_{v_j \in \mathcal{P}_{\mathcal{G}}(v_i)} h^{(v_j)} + b) \\ y^{(v_i)} = \sigma(Vh^{(v_i)} + c) \end{cases} \tag{2}$$

where $\mathcal{P}_{\mathcal{G}}(v_i)$ denotes the predecessor set of v_i in $\mathcal{G}$.

The forward pass of RNNs can be calculated with Eq. (2). For backward propagation, we need to calculate derivatives at each vertex in the RNNs. For each vertex in the directed acyclic graph, it is processed in the reverse order of forward propagation sequence. In details, to compute the derivatives at vertex v_i, we need to look at the forward passes of all its successors. Let $\mathcal{S}_{\mathcal{G}}(v_i)$ denote the direct successor set for v_i in $\mathcal{G}$. For each $v_k \in \mathcal{S}_{\mathcal{G}}(v_i)$, its hidden layer is computed by

$$\begin{cases} h^{(v_k)} = \phi(Ux^{(v_k)} + Wh^{(v_i)} + \sum\limits_{v_l \in \mathcal{Q}} Wh^{(v_l)} + b) \\ y^{(v_k)} = \sigma(Vh^{(v_k)} + c) \end{cases} \tag{3}$$

where $\mathcal{Q} = \mathcal{P}_{\mathcal{G}}(v_k) - \{v_i\}$. Combining Eq (2) and (3), we can see that the errors back-propagated to the hidden layer at v_i come from two sources: directed errors from v_i (i.e., $\frac{\partial y^{(v_i)}}{\partial h^{(v_i)}}$) and summation over indirected errors

from all its successors $v_k \in \mathcal{S}_{\mathcal{G}}(v_i)$ (i.e., $\sum_{v_k} \frac{\partial y^{(v_k)}}{\partial h^{(v_i)}} = \sum_{v_k} \frac{\partial y^{(v_k)}}{\partial h^{(v_k)}} \frac{\partial h^{(v_k)}}{\partial h^{(v_i)}}$). Therefore, the derivatives at vertex v_i can be obtained by

$$\begin{cases} \mathrm{d}h^{(v_i)} & = V^T \sigma'(y^{(v_i)}) + \sum\limits_{v_k \in \mathcal{S}_{\mathcal{G}}(v_i)} W^T \mathrm{d}h^{(v_k)} \circ \phi'(h^{(v_k)}) \\ \nabla W^{(v_i)} & = \sum\limits_{v_k \in \mathcal{S}_{\mathcal{G}}(v_i)} \mathrm{d}h^{(v_k)} \circ \phi'(h^{(v_k)})(h^{(v_i)})^T \\ \nabla U^{(v_i)} & = \mathrm{d}h^{(v_i)} \circ \phi'(h^{(v_i)})(x^{(v_i)})^T \\ \nabla b^{(v_i)} & = \mathrm{d}h^{(v_i)} \circ \phi'(h^{(v_i)}) \\ \nabla V^{(v_i)} & = \sigma'(y^{(v_i)})(h^{(v_i)})^T \\ \nabla c^{(v_i)} & = \sigma'(y^{(v_i)}) \end{cases} \tag{4}$$

where $\circ$ is the Hadamard product, $\sigma'(\cdot) = \frac{\partial L}{\partial y(\cdot)} \frac{\partial y(\cdot)}{\partial \sigma}$ is the derivative of loss function L with respect to output function σ, and $\phi'(\cdot) = \frac{\partial h}{\partial \phi}$. Note that the superscript T denotes transposition operation.

With Eq (2) and (4), we can perform forward and backward passes on one directed acyclic graph. In this paper, we decompose the undirected cyclic graph into four directed acyclic graphs along southeast, southwest, northwest and northeast directions. Figure 2 visualizes the decomposition. Let $\mathcal{G}^{\mathcal{U}} = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$ denote the undirected cyclic graph, where $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4$ represent the four directed acyclic graphs respectively. For each $\mathcal{G}_m$ ($m = 1, 2, 3, 4$), we can get the corresponding hidden layer h_m by performing RNNs. The summation of all hidden layers are fed to the output layer. We use Eq (5) to express this process

$$\begin{cases} h_m^{(v_i)} = & \phi(U_m x^{(v_i)} + \sum\limits_{v_j \in \mathcal{P}_{\mathcal{G}_m}(v_i)} W_m h_m^{(v_j)} + b_m) \\ y^{(v_i)} = & \sigma(\sum\limits_{\mathcal{G}_m \in \mathcal{G}^{\mathcal{U}}} V_m h_m^{(v_i)} + c) \end{cases} \tag{5}$$

where U_m, W_m, V_m, and b_m are matrix parameters and bias term for $\mathcal{G}_m$, c is the bias term for final output, and $\mathcal{P}_{\mathcal{G}_m}(v_i)$ denotes the predecessor set of v_i in $\mathcal{G}_m$. The error back-propagated to previous convolutional layer at v_i is computed by

$$\nabla x^{(v_i)} = \sum_{\mathcal{G}_m \in \mathcal{G}^{\mathcal{U}}} U_m^T \mathrm{d}h_m^{(v_i)} \circ \phi'(h_m^{(v_i)}) \tag{6}$$

4. Proposed Tracking Algorithm

4.1. Network architecture

The architecture of the proposed network is depicted in Figure 1, which receives a 107×107 (same in [36]) RGB input, and has three convolutional layers (each with ReLU and pooling layers), two fully connected layers and one fully connected classification layer. Each pooling layer is followed by a recurrent layer, which models the structure of object in this level. Besides, to provide the next convolutional layer with more information, we adopt a skip connection strategy to fuse the features from pooling and recurrent layers.

4.2. Training

Inspired by the success in [36], we utilize a set of annotated video sequences to train the whole network. For convolutional layer, it is trained by the Stochastic Gradient Descent (SGD) method, and the recurrent layer is trained by the method introduced in Section 3. Besides, we also adopt the multi-domain learning strategy as in [36]. In the training stage, the final layer has K branches, and only the k^{th} branch is handled in the k^{th} iteration. The whole training process ends when the network converges or a predefined max number of iteration is reached. In the testing stage, the K branches of the final layer are replaced with a single branch corresponding to the tracked object. By adopting the multi-domain strategy, the performance of the proposed tracker is further improved.

4.3. Tracking and update

Visual tracking is achieved within the particle filter framework. For each new frame, we sample N target candidates $\{c_i\}_{i=1}^N$ around the position of target in last frame, and evaluate them by their positive scores $p(c_i)$ obtained by the network. The positive score of each candidate indicates its probability belonging to target class. The candidate with the highest positive score is chosen to be the tracked result O as follows

$$O = \underset{c_i}{\operatorname{argmax}}\ p(c_i) \tag{7}$$

Due to object appearance variation caused by factors such as lighting change and deformation, update is essential during tracking. We adopt two strategies to update the network as in [36]: short-term and long-term updates. When the positive score $p(O)$ of the tracked result is smaller than a predefined threshold θ, the short-term update is performed. Otherwise, the long-term update is executed. For the long-term update, the whole network is updated with the collected positive samples for a long period of time and negative samples stored for a short period time. While for the short-term update, both positive and negative samples for update are collected from a short period of time.

4.4. Hard minibatch mining

In tracking, most negative samples are redundant, and only a few distracting negative samples are helpful in training a discriminative classifier. In this situation, the plain SGD method easily results in drift due insufficient effective negative samples. To address this problem, [36] leverages a popular solution, i.e., hard negative mining, in object detection [41]. In this paper, we utilize the same strategy to alleviate this problem.

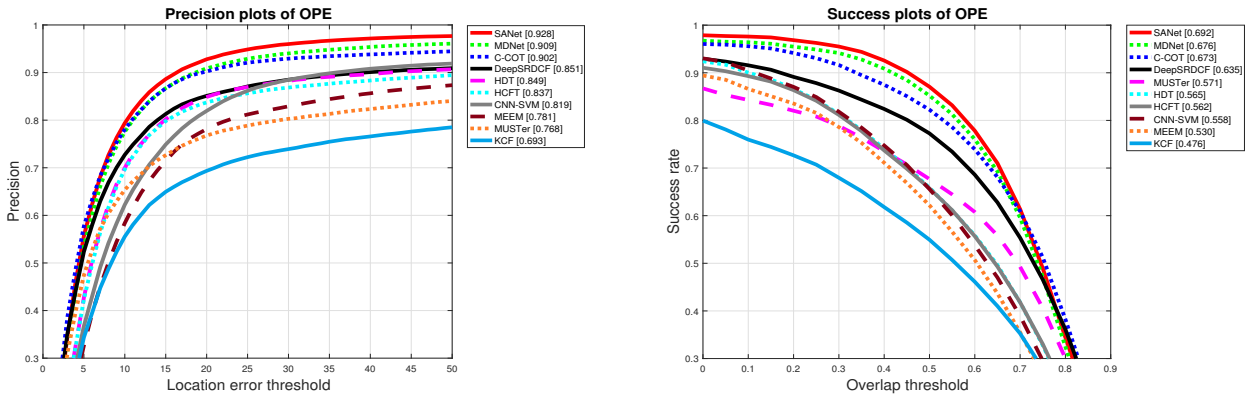

Figure 3. Precision and success plots on OTB100 [51]. The numbers in the legend indicate the representative precisions at 20 pixels for precision plots, and the area-under-curve scores for success plots.

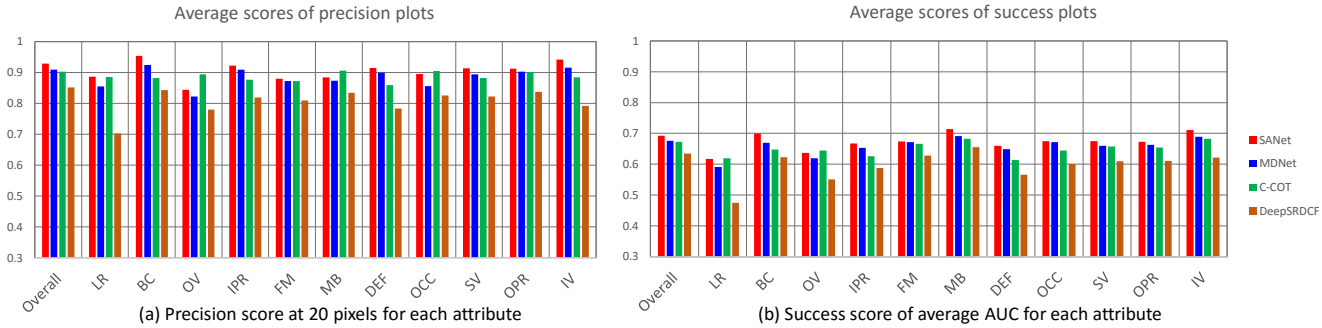

Figure 4. Precision scores at 20 pixels and success scores of average AUC of the four leading trackers under different attributes of test sequences in OPE on OTB [51], including illumination variation (IV), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-view (OV), background cluttered (BC) and low resolution (LR).

4.5. Box refinement

To locate the target object, we sample multiple positive samples around the target, which may result in failure to find the tight boxes enclosing the target. To handle this issue, [36, 42] adopt a refinement step in each frame to improve the predicted bounding box. In this paper, the same strategy is utilized. In the first frame, we train a simple linear regression model to predict the position of target. In subsequent frames, we use the regression model to adjust the target locations obtained by Eq. (3) if the positive score of the tracked result is larger than θ.

5. Experiments

5.1. Implementation details

The proposed method is implemented in Matlab based on MatConvNet [45], and runs at around 1 frames per second (FPS) with 3.7 GHz Intel i7 Core and a NVIDIA GTX TITAN Z GPU. In each new frame, we sample 300 ($N = 300$) target candidates in translation and scale dimension from a Gaussian distribution. Three independent RNNs are utilized to model image unit dependencies in multiple levels, i.e., the 1^{st}, 2^{nd} and 3^{rd} pooling layers. The dimension of hidden layers of RNNs are set to the same as the channels of the 1^{st}, 2^{nd} and 3^{rd} pooling layers. The learning rates of RNNs are initialized to be 10^{-3} and decay exponentially with the rate of 0.9. Other parameters of convolutional layers are set to the same as in [36].

5.2. Evaluation on OTB

OTB100 [51] is a popular tracking benchmark containing 100 fully annotated videos with various challenges. We employ the precision plots and success plots defined in [51] to evaluate the robustness of the tracking approaches. In addition to the trackers included in the benchmark [51], e.g., SCM [56] and Struck [19], we also compare our method with most recent state-of-the-art trackers including MEEM [53], TGPR [15], MDNet [36], MUSTer [22], CNN-SVM [21], DeepSRDCF [7], C-COT [8], HCFT [34], HDT [37] and KCF [20]. To train the network, we utilize image sequences collected from VOT2013 [26], VOT2014 [25] and VOT2015 [24], excluding the videos included in OTB [51].

Figure 3 shows the comparisons of our method with oth-

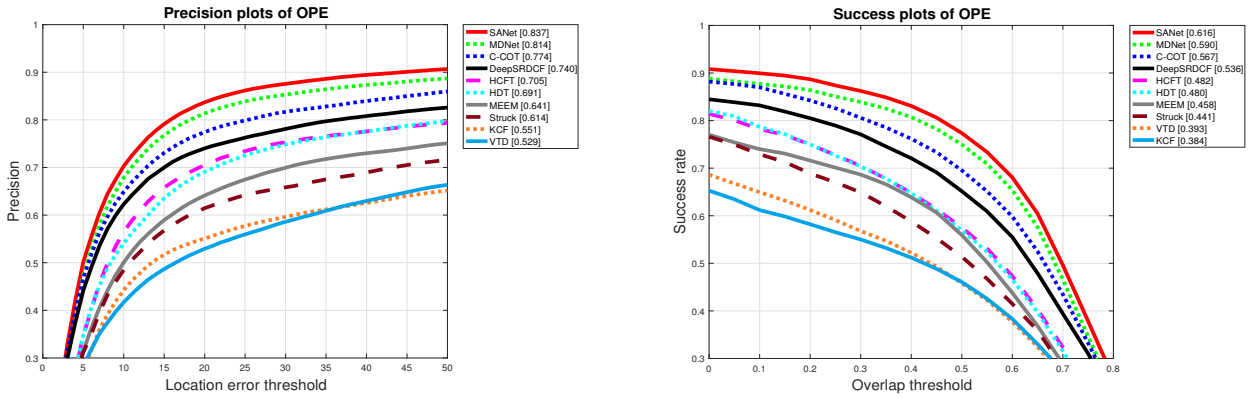

Figure 5. Precision and success plots on TC-128 [32]. The numbers in the legend indicate the representative precisions at 20 pixels for precision plots, and the area-under-curve scores for success plots.

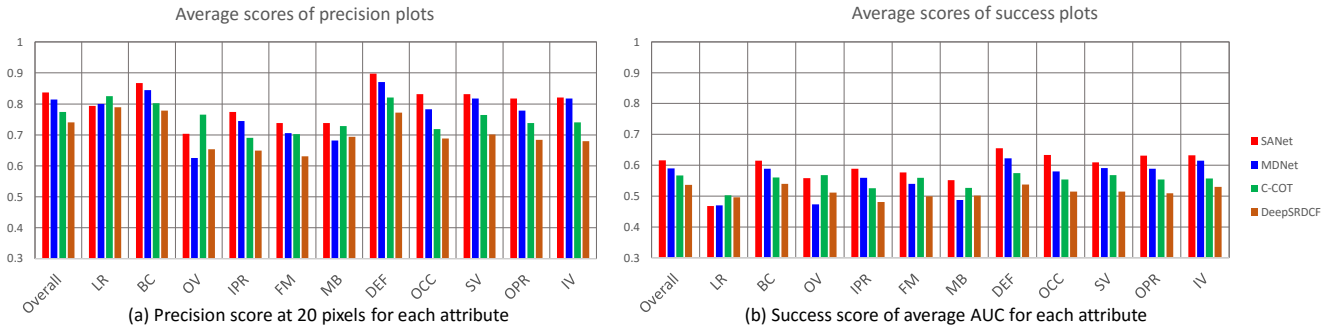

Figure 6. Precision scores at 20 pixels and success scores of average AUC of the four leading trackers under different attributes of test sequences in OPE on TC-128 [32], including: IV, OPR, SV, OCC, DEF, MB, FM, IPR, OV, BC and LR.

er state-of-the-art tracker in terms of precision and success plots, respectively. From Figure 3, we can see that the proposed approach outperforms other state-of-the-art trackers in both measures. The exceptional scores at mild thresholds means our tracker hardly misses targets while the competitive scores at strict thresholds implies that our algorithm also finds tight bounding boxes to targets. Among other trackers, [36] also utilizes deep convolutional neural networks to learn the object appearance representation. However, it does not take self-structure information of object into account. While our method considers structure of object during learning, and improves the ability of network to distinguish object from background. Figure 4 illustrates that our tracker is able to effectively deal with various challenging situations. It is worth noticing that, compared with the method in [36], our method improves performance of tracking in all 11 attributes.

In [36], an effective strategy, i.e., multi-domain learning, is adopted to train the networks. In this work, we also leverage this strategy to train the networks. To verify the impact of multi-domain learning, we conduct another experiments without multi-domain learning method to train the network, while keep other conditions the same. Without multi-domain learning, our method achieves 0.922 ranking score in precision plots and 0.688 ranking score in success plots. Compared with using multi-domain learning method, the tracking performance slightly degrades, which demonstrates the effectiveness of multi-domain learning strategy.

5.3. Evaluation on TC-128

TC-128 [32] contains 128 fully annotated color image sequences. We use the same metrics used in [51] and [32], i.e., precision and success plots, to evaluate the tracking methods. In addition to the trackers tested in the benchmark [32], we add some recent trackers including [7], C-COT [8], HCFT [34], HDT [37] and MDNet [36]. To train the network, we use sequences in VOT2013 [26], VOT2014 [25] and VOT2015 [24], excluding the videos in OTB [51].

Figure 5 illustrates the comparisons of our algorithm with other methods in terms of precision and success plots, respectively. From Figure 5, we can see that our approach outperforms other state-of-the-art trackers in both measures. Besides, to facilitate more detailed analysis, we also report the performance of four lead tracker on different attributes in Figure 6. Experimental results demonstrate that our method can well deal with various challenging factors

Table 1. The average scores and ranks of accuracy and robustness of different methods on VOT2015 [24]. The top three scores are highlighted in red, blue and green, respectively.

Trackers	Accuracy		Robustness		Expected
	Rank	Score	Rank	Score	overlap ratio
DSST	2.92	0.54	5.65	2.56	0.1719
DeepSRDCF	2.03	0.57	2.32	1.05	0.3181
LGT	5.75	0.42	4.72	2.21	0.1737
MEEM	3	0.5	4.32	1.85	0.2212
MUSTer	2.87	0.52	4.48	2	0.1950
SAMF	2.68	0.53	4.18	1.94	0.2021
TGPR	3.48	0.48	5.08	2.31	0.1938
MDNet	1.2	0.6	1.62	0.69	0.3783
SANet	1.17	0.61	1.58	0.69	0.3895

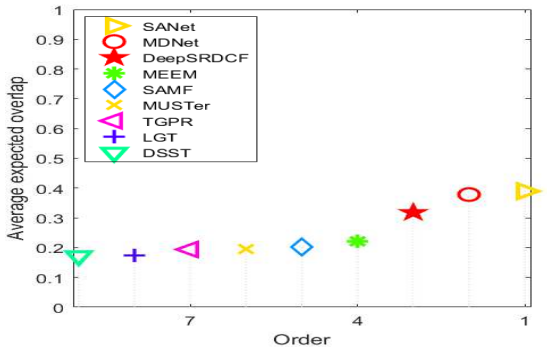

Figure 7. Expected average overlap ratio graph with trackers ranked from right to left.

and consistently outperform the other three trackers in most attributes.

5.4. Evaluation on VOT2015

VOT2015 [24] contains 60 image sequences with various challenges. According to VOT challenge protocol in [24], a tracker is re-initialized whenever failure happens. Two metrics, accuracy and robustness, are utilized to evaluate the performance of trackers. Besides, the VOT challenge also adopts the expected average overlap as a new evaluation metric, which estimates how accurate the estimated bounding box is after a certain number of frames are processed since initialization. We compare our method with eight state-of-the-art trackers, including DSST [6], DeepSRDCF [7], MDNet [36], TGPR [15], MEEM [53], MUSTer [22], SAMF [31], and LGT [4]. Our network is pre-trained using sequences from OTB100 [51], excluding the sequences in VOT2015 [24] dataset.

Table 1 summarizes the comparison of our tracker with other approaches. From the table we can see that the proposed method outperforms other trackers in all evaluation metrics. Especially, compared with MDNet [36], our tracker demonstrates advances in both accuracy and robustness, showing again the benefits of taking structure information into account. Figure 7 visualizes the ranks of trackers on VOT2015 [24] in term of expected overlap ratio.

6. Conclusion

We present a novel network architecture named SANet for visual tracking by taking into consideration self-structure information of a target object. Different from previous CNNs-based tracking methods, which mainly concentrate on inter-class classification and thus are prone to cause drift in presence of similar distractors, our SANet leverages RNNs to model the structure of target object and combines such structural information with CNNs to learn a discriminative appearance model, which is effective for distinguishing not only background objects of inter-class but also similar distractors of intra-class. Experimental results on three large-scale tracking benchmarks demonstrate the effectiveness of our method.

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, 2009.

[2] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *CVPR*, 2012.

[3] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015.

[4] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE TPAMI*, 35(4):941–953, 2013.

[5] Z. Cui, S. Xiao, J. Feng, and S. Yan. Recurrently target-attending tracking. In *CVPR*, 2016.

[6] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[7] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015.

[8] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[9] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, 2009.

[10] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[11] H. Fan, X. Mei, D. Prokhorov, and H. Ling. Multi-level contextual rnns with attention model for scene labeling. *arXiv:1607.02537*, 2016.

[12] H. Fan and J. Xiang. Robust visual tracking with multitask joint dictionary learning. *IEEE TCSVT*, 2016.

[13] H. Fan, J. Xiang, H. Liao, and X. Du. Robust tracking based on local structural cell graph. *JVCIR*, 31:54–63, 2015.

[14] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *IEEE TNN*, 21(10):1610–1623, 2010.

[15] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[17] A. Graves. Sequence transduction with recurrent neural networks. In *ICML*, 2014.

[18] A. Graves, S. Fernández, and J. Schmidhuber. Multi-dimensional recurrent neural networks. In *ICANN*, 2007.

[19] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[20] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE TPAMI*, 37(3):583–596, 2015.

[21] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015.

[22] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *CVPR*, 2015.

[23] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE TPAMI*, 34(7):1409–1422, 2012.

[24] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder. The visual object tracking vot2015 challenge results. In *ICCV Workshops*, 2015.

[25] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al. The visual object tracking vot2013 challenge results. In *ICCV Workshops*, 2013.

[26] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al. The visual object tracking vot2013 challenge results. In *ECCV Workshops*, 2014.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[28] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, 2010.

[29] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[30] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE TIP*, 25(4):1834–1848, 2016.

[31] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV Workshops*, 2014.

[32] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: algorithms and benchmark. *IEEE TIP*, 24(12):5630–5644, 2015.

[33] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[34] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[35] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE TPAMI*, 33(11):2259–2272, 2011.

[36] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[37] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, and J. L. M.-H. Yang. Hedged deep tracking. In *CVPR*, 2016.

[38] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[39] B. Shuai, Z. Zuo, G. Wang, and B. Wang. Dag-recurrent neural networks for scene labeling. In *CVPR*, 2016.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[41] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE TPAMI*, 20(1):39–51, 1998.

[42] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[43] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[44] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang. A siamese long short-term memory architecture for human re-identification. In *ECCV*, 2016.

[45] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *ACM MM*, 2015.

[46] D. Wang and H. Lu. Visual tracking via probability continuous outlier model. In *CVPR*, 2014.

[47] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[48] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan. Saliency detection with recurrent fully convolutional networks. In *ECCV*, 2016.

[49] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

[50] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *ICCV*, 2011.

[51] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE TPAMI*, 37(9):1834–1848, 2015.

[52] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM CSUR*, 38(4):13, 2006.

[53] J. Zhang, S. Ma, and S. Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014.

[54] K. Zhang, L. Zhang, and M.-H. Yang. Fast compressive tracking. *IEEE TPAMI*, 36(10):2002–2015, 2014.

[55] T. Zhang, A. Bibi, and B. Ghanem. In defense of sparse tracking: Circulant sparse tracker. In *CVPR*, 2016.

[56] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012.

[57] H. Zuo, H. Fan, E. Blasch, and H. Ling. Combining convolutional and recurrent neural networks for human skin detection. *IEEE SPL*, 24(3):289–293, 2017.

[58] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen. Learning contextual dependence with convolutional hierarchical recurrent neural networks. *IEEE TIP*, 25(7):2983–2996, 2016.