Optical Engineering

OpticalEngineering.SPIEDigitalLibrary.org

Robust visual tracking via multiscale deep sparse networks

Xin Wang
Zhiqiang Hou
Wangsheng Yu
Yang Xue
Zefenfen Jin
Bo Dai

SPIE.

Xin Wang, Zhiqiang Hou, Wangsheng Yu, Yang Xue, Zefenfen Jin, Bo Dai, "Robust visual tracking via multiscale deep sparse networks," *Opt. Eng.* **56**(4), 043107 (2017), doi: 10.1117/1.OE.56.4.043107.

Robust visual tracking via multiscale deep sparse networks

Xin Wang, Zhiqiang Hou,* Wangsheng Yu, Yang Xue, Zefenfen Jin, and Bo Dai
Air Force Engineering University, Information and Navigation College, Xi'an, China

Abstract. In visual tracking, deep learning with offline pretraining can extract more intrinsic and robust features. It has significant success solving the tracking drift in a complicated environment. However, offline pretraining requires numerous auxiliary training datasets and is considerably time-consuming for tracking tasks. To solve these problems, a multiscale sparse networks-based tracker (MSNT) under the particle filter framework is proposed. Based on the stacked sparse autoencoders and rectifier linear unit, the tracker has a flexible and adjustable architecture without the offline pretraining process and exploits the robust and powerful features effectively only through online training of limited labeled data. Meanwhile, the tracker builds four deep sparse networks of different scales, according to the target's profile type. During tracking, the tracker selects the matched tracking network adaptively in accordance with the initial target's profile type. It preserves the inherent structural information more efficiently than the single-scale networks. Additionally, a corresponding update strategy is proposed to improve the robustness of the tracker. Extensive experimental results on a large scale benchmark dataset show that the proposed method performs favorably against state-of-the-art methods in challenging environments. [DOI: 10.1117/1.OE.56.4.043107]

Keywords: visual tracking; deep learning; sparse network; particle filter; rectifier linear unit.

Paper 161997 received Jan. 7, 2017; accepted for publication Mar. 29, 2017; published online Apr. 21, 2017.

1 Introduction

Visual tracking is one of the current research hotspots in computer vision. It has been widely used in many fields, such as visual surveillance, human–computer interface, medical image analysis,[1,2] and so on. Given the initial state of the target (including position, scale, etc.), the classical visual trackers achieve the tracking by estimating its continuous states in following frames.

In recent years, a large number of tracking algorithms have been proposed. Existing tracking algorithms can be divided into two categories:[3] generative methods and discriminative methods. The former is a "model-driven" method that uses the target's information to establish the target model and determines the most similar sample as the tracking result. Some popular generative methods include incremental visual tracking (IVT),[4] multitask tracking (MTT),[5] and adaptive structural local appearance model (ASLA).[6] The latter is a "data-driven" method that deals with the tracking process as a binary classification problem between target and background. Some state-of-the-art trackers, such as compressive tracking (CT),[7] tracking-learning-detection (TLD),[8] and multiple instance learning (MIL),[9] are discriminative methods. These above trackers, which use hand-crafted features, achieve a good performance in simple and controllable environments, but there are always some problems of tracking drifting or a target missing in some practical and complicated environments, such as illumination variation, deformation, occlusion, motion blur, and so on. Therefore, there is still a challenging gap between a robust real-time tracker and the realistic application in extreme and complicated conditions.

The emergence and development of "deep learning" has gradually become the potential solution to the above problems.[10] Different from hand-crafted features, deep learning learns the high-level semantic features automatically. These features are effective in distinguishing the target from background due to the deep architectures of deep learning. Recently, the deep learning-based trackers have been gradually becoming the tendency in visual tracking fields due to their outperformance compared with traditional tracking methods.

However, the tracking methods based on deep learning still suffer from some difficulties.[11]

1. Numerous data are required to train a robust and stable deep network. However, there is limited number of labeled data in an actual tracking scene. The unsupervised pretraining method with numerous auxiliary training datasets[12] solved the problem to some extent, but it still requires high-performance hardware and is complicated and time-consuming. Moreover, the learned generic representation may not be suitable tracking a specific object.
2. The "gradient vanishing" problem easily occurs in the stochastic gradient descent[13] method during the training process of deep networks. It is caused by the property of saturation of the traditional nonlinear activation function and often results in a dilemma in training a robust deep network.
3. Traditional deep learning-based methods track the targets via a single-scale deep network. The samples are usually normalized into a unified pattern in the single tracking network. It will cause the deformation of the target and loss of some inner structure information of data. These factors are more likely to result in tracking drift to some degree.

*Address all correspondence to: Zhiqiang Hou, E-mail: hou-zhq@sohu.com

In this paper, we propose a multiscale deep sparse network (MDSN) and build a robust tracker: multiscale sparse networks-based tracker (MSNT). The main contributions of our work can be summarized as follows:

1. We propose an MDSN based on the stacked sparse autoencoders (SAE)[14] and rectifier linear unit (ReLU).[15,16] The combination of SAE and ReLU makes the deep network highly sparse and avoids the complex and time-consuming pretraining. The multiscale networks can retain the inner structure information of targets as much as possible. The architecture improves the robustness of deep networks for different shapes of targets.
2. Due to unsaturation and constancy of the gradient of ReLU, the "gradient vanishing" problem of training is effectively alleviated by MDSN. It also makes the online training of the deep networks easier and faster.
3. Combined with the particle filter framework, we built a simple but effective tracker named MSNT by the MDSN for overcoming the weakness of traditional trackers based on a single network. MSNT can automatically choose the corresponding tracking network according to different targets. It further improves the robustness of the trackers based on a single network.

A large number of experiments and analyses are carried out on the CVPR2013 tracking benchmark dataset[17] (including 51 challenging videos) with nine recent state-of-the-art trackers. The experimental results show that our tracker achieves outstanding performance in challenging environments and attains a practical tracking speed.

2 Related Work

The concept of "deep learning" was first proposed by Hinton and Salakhutdinov.[12] Since then, deep learning technology has been widely concerned and has been making great progress. With its robust and efficient features, deep learning has been applied in diverse fields, such as image classification,[14,18] automatic speech recognition,[19] face recognition,[20] and so on.

Recently, deep neural networks (DNNs) have been applied in the visual tracking field. Fan et al.[21] extracted specific features from convolutional neural networks (CNNs) with offline pretraining for human tracking and obtained acceptable tracking results in some complex situations. Through training a stacked denoising autoencoder on a large scale image dataset, deep learning tracker (DLT)[22] learned generic features and achieved a robust tracking performance. Li et al.[23] applied a single-CNN on visual tracking tasks without pretraining and combined it with multiple image cues to improve the tracking success rate. Wang et al.[24] used hierarchical features for tracking by training a two-layer CNN on an auxiliary dataset and gained a good result in complicated tracking situations. Zhang et al.[25] proposed a convolutional network-based tracker (CNT), which combined the local structure feature and global geometric information of tracking targets and attained a state-of-the-art performance.

The sparse distributed representation (SDR) is the key for learning powerful features in deep learning, while the activation function plays an important role in encouraging sparsity.[26] The performance of the activation function will directly influence the effectiveness and robustness of the extracted features. The most popular nonlinear activation functions are "sigmoid" and "tanh." They have been widely used in many deep networks, but they suffer from some drawbacks,[11] such as a slow training speed and a poor local solution with random initialization without good predictive performance. Recently, a sparse activation function called ReLU was proposed in Ref. 15. As illustrated in Fig. 1, different from traditional activation functions, such as sigmoid and tanh, the rectifier function $\mathrm{ReLU}(x) = \max(0, x)$ is a one-side activation function. It enforces hard zeros in the learned feature representation[26] and leads to the sparsity of hidden units by rectifying the negative output of the hidden units.[16] The sparsity of hidden units has the same effectiveness as the pretraining methods. The experimental results in Ref. 27 showed that pretraining will lead to more sparsity of the deep networks compared to DNNs without pretraining.

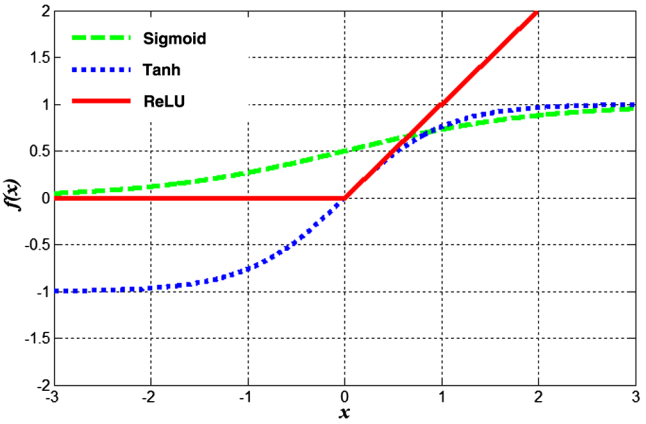


Fig. 1 The activation function curves.

Moreover, Glorot's experiments[16] proved that deep networks with ReLU can reach their best performance without any unsupervised pretraining due to the sparsity. More experiments further proved the conclusion in Ref. 27 and showed that there is no significant improvement for DNNs with ReLU using pretraining. Moreover, ReLU was used in a sparse deep stacking network (S-DSN) for image classification in Ref. 18. It avoided the expensive inference effectively and achieved higher sparsity and better classification performance than S-DSN with sigmoid. Furthermore, the active part of ReLU is an unsaturated linear function, which alleviates the "gradient vanishing" problem effectively in training and improves the speed of training. Therefore, ReLU is a practical activation function for quickly building sparse and powerful deep architectures without requiring pretraining process.

3 Deep Sparse Network Model

Different from sparse coding (SC), the sparsity of neural networks attempts to represent the features of the input data using the least amount of hidden neurons. The feature of objects in sparse neural networks is SDR,[26] which dictates that all representational units participate in data representation while very few units activate for a single data sample. It can exploit more powerful and robust feature representations from input data. Therefore, it is reasonable to build a model of deep sparse network for tracking.

The SAE is a basic unsupervised learning model and is often used in deep learning. In this paper, we use a structure of SAE that is similar to Ref. 14 and obtain a deep sparse network by training the stacked-SAEs using the "layer-by-layer greedy algorithm."[12] The cost function in the model is defined as

$$J(W,\mathbf{b}) = \sum_{i=1}^{m} \|x_i - \hat{x}_i\|_2^2 + \lambda(\|W\|_F^2 + \|W'\|_F^2) + \mu H(\rho||\hat{\rho}), \tag{1}$$

where $\hat{x}_i$ denotes the reconstruction of x_i, W and W' are the weight matrices of encoder and decoder, respectively, $\mathbf{b}$ is the bias vector of encoder included in $\hat{x}_i$, m is the number of samples, λ is a penalty factor, which balances the reconstruction loss and weights, μ is the sparsity penalty factor, and $\|\cdot\|_F$ denotes the Frobenius norm. The cross-entropy $H(\rho||\hat{\rho})$ is given as

$$H(\rho||\hat{\rho}) = -\sum_{j=1}^{n}[\rho_j \log(\hat{\rho}_j) + (1-\rho_j)\log(1-\hat{\rho}_j)], \tag{2}$$

$$\hat{\rho}_j = \frac{1}{k}\sum_{i=1}^{k}[h_j(x_i)], \tag{3}$$

where k and n are the number of neurons in the input layer and hidden layer, respectively. $h_j(x_i)$ denotes the activation value in the j'th hidden layer to the input x_i. The sparsity target ρ is close to 0, and it is set to 0.05 in our experiments.

In Refs. 16, 18, and 27, it is proven that ReLU will bring the inherent sparsity to DNNs, which let the pretraining become less effective for DNNs when using the ReLU activation function. Hence, we adopt ReLU as an activation function to the aforementioned deep sparse network to leave out the offline pretraining. Benefiting from the intrinsic sparsity of ReLU, around 50% of the hidden units' output values are real zeros once the deep network is built. This makes the basic stacked-SAEs transform into a variant, as shown in Fig. 2. Moreover, this percentage of inactive neurons (units that do not activate for any data sample) can easily increase with online training based on the sparsity constraints of SAE.[16]

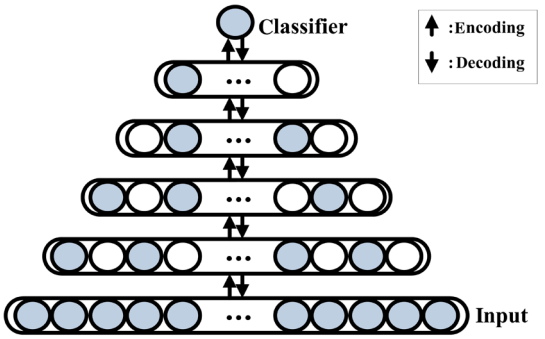


Fig. 3 The deep sparse neural network for tracking.

Based on the architecture of Fig. 2(b), a "softmax" classifier layer is added to the model as the last layer to classify the learned features. The logistic regression is included in the softmax classifier layer

$$l_\theta(t) = \frac{1}{1+e^{-\theta^T t}}, \tag{4}$$

where $l_\theta(t)$ is a value in [0, 1], i.e., it represents the probability of the sample t as the true target in the visual tracking problem and θ is the model parameter. The final model of the deep sparse neural network for tracking is shown in Fig. 3. During the tracking process, each sampling patch gets a value in [0, 1] through the softmax classifier in the tracking network.

4 Tracking Algorithm Based on Multiscale Deep Sparse Networks

A single deep sparse network for tracking is introduced in Sec. 3. However, this fixed architecture of deep network is too rigid in practical tracking tasks, and it cannot adapt to different situations effectively. Based on the single network model mentioned in Sec. 2, we propose an MDSN and combine it with a particle filter framework to cope with the complex tracking tasks.

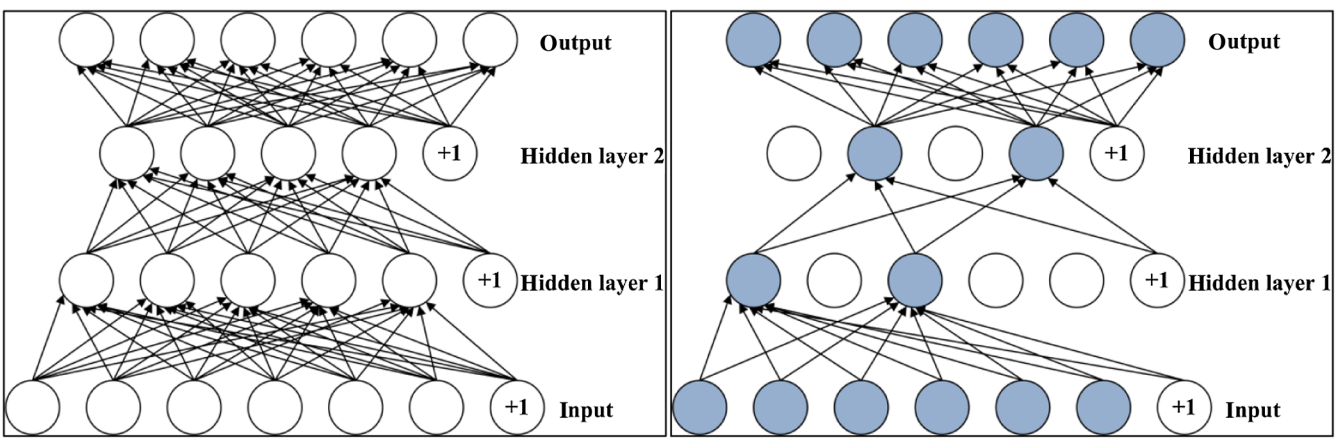


Fig. 2 The basic stacked-SAEs and its variant with ReLU: (a) the basic stacked-SAEs and (b) the variant of stacked-SAEs with ReLU.

4.1 *Multiscale Deep Sparse Networks*

The conventional neural network for tracking usually normalizes the initial target patch or sampling patches into the same size in the input layer, which can reduce the number of input neurons and the complexity of networks effectively. For example, the target patch in the first frame is normalized into a low-resolution (LR) image with 32×32 pixels in DLT.[22]

However, we observe in several experiments that the fixed normalization for different targets will cause various degrees of stretching or compressing for the targets. The deformation damages the inner structure information of the targets, reduces robustness of the extracted features, and increases tracking drifting. However, when different normalized method is used in different shapes of targets, it reserves more inner structure information and achieves a better tracking result due to the reduction of deformation of targets.

As shown in Fig. 4, the red bounding box and line represent the tracker based on a 32×16 normalization scale (normalization-2) while the green ones represent the 32×32 normalization (normalization-1). We clearly observe that the 32×16 normalization has better performance than 32×32 normalization in this case.

Based on the observations, we propose an MDSN to adapt to different targets and situations effectively. It is called "multiscale" because we build four different architectures of deep sparse networks aimed at four different kinds of situations. The four defined situations of tracked targets and corresponding architectures of deep network are as follows:

1. LR-target: The number of pixels inside the initial ground-truth bounding box is less than t_r ($t_r = 400$).[17] In this situation, we normalize the input image patches into a 16×16 pixels grayscale and build a six-layer deep network in which the amounts of neurons of each layer are [256 512 256 128 64 1]. The deep architecture has an overcomplete filter layer after the input layer. It will capture the image's structure more effectively[22] for LR-targets.
2. Square target (S-target): The target is not LR, and the aspect ratio $r \in [\frac{2}{3}, \frac{3}{2}]$, where $r = w/h$ and w and h are the width and height of the initial ground-truth bounding box of the target, respectively. In this situation, the width and height of the initial target are approximately equal, so we normalize the input image patches into 32×32 grayscale. Hence, a six-layer deep network with neurons of [1024 512 256 128 64 1] is built.
3. Vertical target (V-target): The target is not LR, and the aspect ratio $r < \frac{2}{3}$. In this situation, the height is 1.5 times greater than width of the initial target, so we consider the target a V-target and normalize the input image patches into 32×16 grayscales. A five-layer deep network is built, and the amounts of neurons in each layer are [512 256 128 64 1].
4. Horizontal target: The target is not LR, and the aspect ratio $r > \frac{2}{3}$. Contrary to the V-target, the width is 1.5 times greater than height of the initial target. We normalize the image patches into 16×32 grayscales and build a five-layer deep network of [512 256 128 64 1].

The entire architecture of the MDSN is shown in Fig. 5. With a new tracking task, MDSN first chooses a corresponding tracking network according to the above defined situations. The multiscale architecture reserves the inner structure information of targets as much as possible, so it will improve the robustness of the extracted features.

4.2 *Particle Filter Tracking Framework*

The particle filter algorithm[22,28] is a popular tracking framework used in the visual tracking field. Let s_t and z_t denote the state and observation of the target at time t, respectively. The tracking task can be considered the process of searching for the target's state of maximum probability at time t according to the observations $\{z_{1:t}\}$

$$s_t = \arg \max p(s_t|z_{1:t}), \tag{5}$$

where $p(s_t|z_{1:t})$ is the posterior distribution of the target at time t. According to Bayes criterion, we get that

$$p(s_t|z_{1:t}) = \frac{p(z_t|s_t)p(s_t|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \tag{6}$$

The particle filter algorithm estimates the posterior distribution through a set of random particles $\{s_t^i\}_{i=1}^N$ with corresponding weights $\{\omega_t^i\}_{i=1}^N$, where N denotes the numbers of

Fig. 4 Comparisons for two trackers based on different normalizations (red bounding box and line represent 32×16 normalization while green ones represent 32×32 normalization). (a) The tracking results of two trackers based on different normalizations and (b) CLEs of two trackers based on different normalizations.

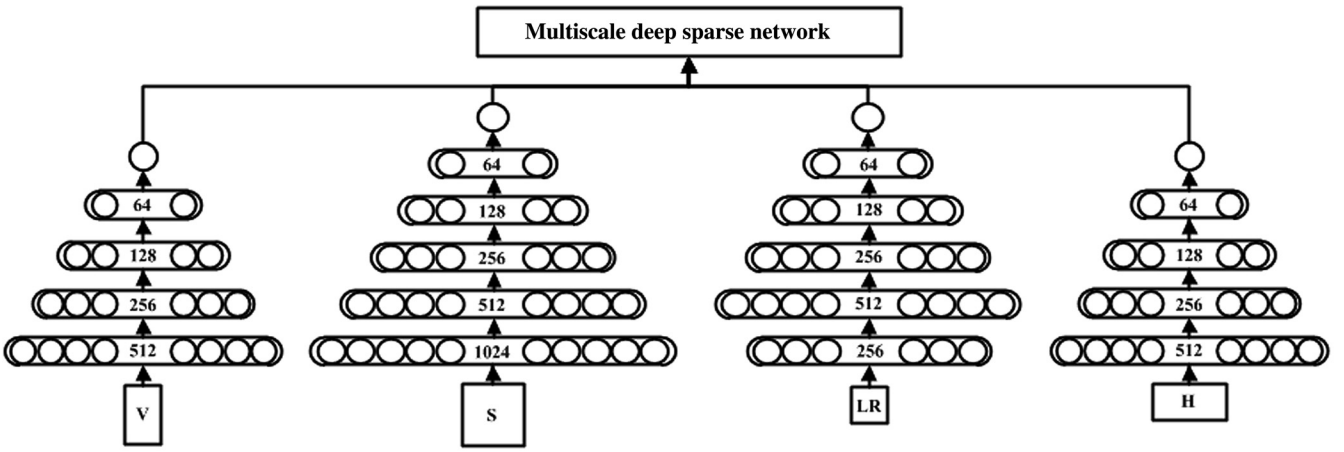


Fig. 5 The architecture of MDSN.

sampling particles and the initial weights are $1/N$. The weights of particles easily produce weight degeneracy, so the weights are updated as follows:

$$\omega_t^i = \omega_{t-1}^i \cdot \frac{p(z_t|s_t^i)p(s_t^i|s_{t-1}^i)}{q(s_t^i|s_{t-1}^i, z_t)}, \tag{7}$$

where $q(s_t^i|s_{t-1}^i, z_t)$ is the proposed distribution, which depends on the particle distribution at time $t-1$ and the observation at time t. Additionally, it is often simplified to a first-order Markov process $q(s_t^i|s_{t-1}^i)$, which is independent of the current observation. Thus, the update formulation can be simplified to

$$\omega_t^i = \omega_{t-1}^i \cdot p(z_t|s_t^i). \tag{8}$$

Meanwhile, the weights should be further normalized to satisfy the below equation

$$\sum_{i=1}^{N} \omega_t^i = 1. \tag{9}$$

In our proposed algorithm, we use the particle filter to randomly sample the candidate patches around the last tracking results; then, we send the sampling patches to the tracking network, which is proposed in Sec. 4.1. We get the confidence coefficient ς_i through the classifier layer, i.e., the posterior distribution $p(s_t|z_{1:t}) = \varsigma_i$, and then we choose the maximum ς_i to get the current target's state by Eq. (5). Meanwhile, to adapt to the changes of the target's scales during tracking, a random disturbance $r = (w_r, h_r)$ is added to the width and height of the candidate patches. In this paper, w_r and h_r obey normal distribution with zero mean and a variance of 0.01.

4.3 *Online Training and Updating Strategy for Tracking Network*

After determining the corresponding tracking network, the tracking network with random initialization cannot satisfy the requirements of the specific tracking task, so we adjust the network parameters using specific labeled samples by online training.

In specific tracking tasks, we need to collect enough positive and negative samples to train the network while only the initial state $s_0 = \{x_0, y_0, w_0, h_0\}$ is given. Here, (x_0, y_0) denotes the initial position of the target and w_0 and h_0 denote the width and height, respectively. In our proposed method, we randomly collect 10 positive samples close to the target's center and 100 negative samples far away from the target. Using these positive and negative samples to train the tracking network at the beginning of tracking, we get the adapted network for specific tasks.

A robust tracking algorithm should be able to consistently track the target without drifting or losing, which requires the tracker to have the capacity of adjusting parameters adaptively according to changes of environments. The condition to update the proposed method is as follows:

$$\max(\varsigma_i) < \tau || \text{fn} \geq \eta, \tag{10}$$

where τ is the threshold of updating, fn is the accumulative frames after the last update, and η is the maximum

Table 1 The main steps of MSNT algorithm.

Algorithm: The proposed MSNT algorithm

Input: Image sequences $I_1, I_2, \ldots, I_n$, initial target state $s_0 = \{x_0, y_0, w_0, h_0\}$.

Output: Tracking results, i.e., the estimated object state $\hat{s}_i$ for frame i.

Step 1 Determine the tracking network based on the target type with s_0 and initialize network.

Step 2 Collect positive sample patches and negative sample patches to train the network online.

Step 3 For $i = 1,2, \cdots, n$:

Step 3.1 Do particle sampling to get N sample patches in the neighborhood of (x_{i-1}, y_{i-1});

Step 3.2 Send the sample patches to the tracking network, to get the confidence coefficient ς_i;

Step 3.3 Choose the maximum ς_i to get the estimated state by Eq. (5);

Step 3.4 Update the network according to Eq. (10) and the updating strategy.

Step 4 End of the image sequences.

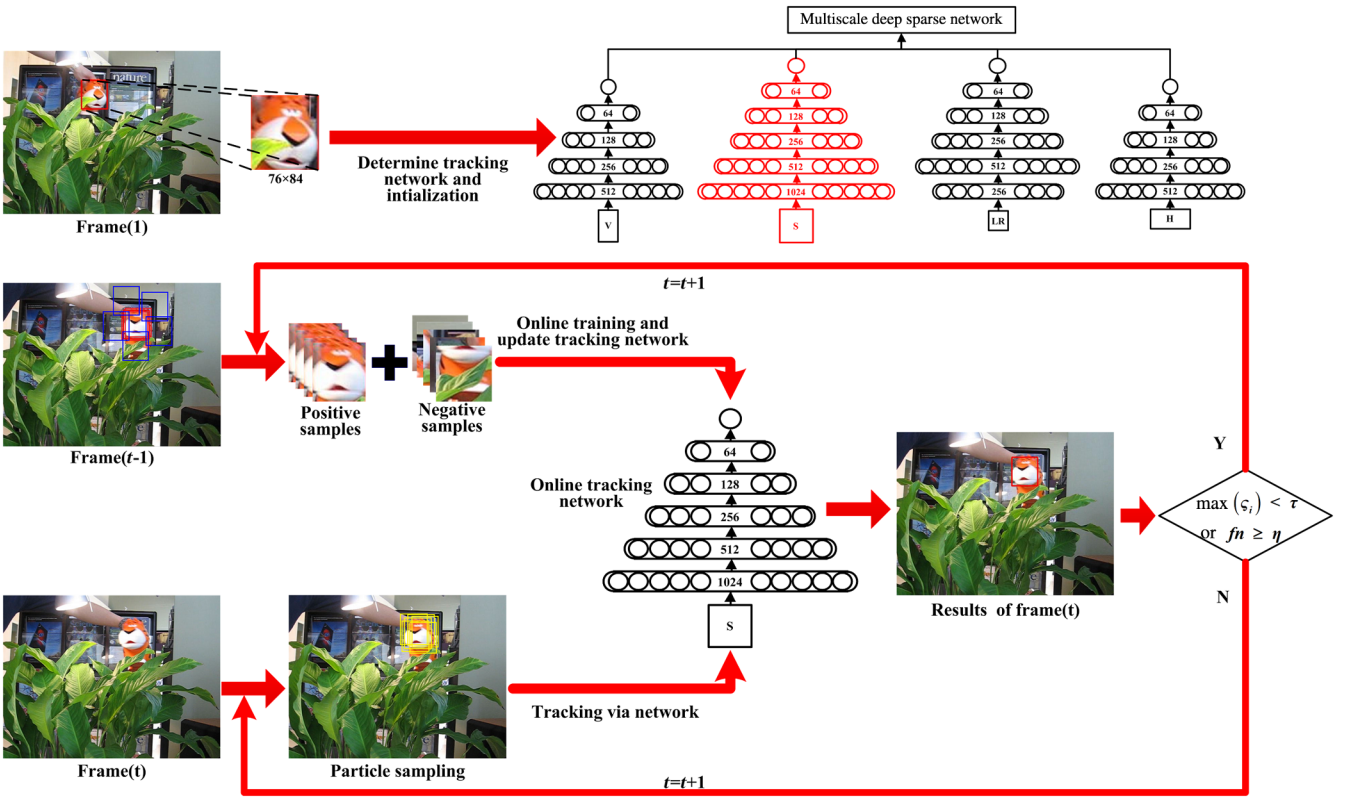


Fig. 6 Flow chart of MSNT algorithm.

accumulative frames. If Eq. (10) is satisfied, the current tracking result will be added to the positive samples set, and the negative samples will be randomly sampled again in the current frame. Then, it is retrained by utilizing the updated positive and negative samples to realize the updating of the tracking network.

4.4 *Overall Process of Algorithm*

Integrating the above description of the key components, we present a visual tracking method MSNT via the proposed MDSNs. The main steps of MSNT are shown in Table 1, and the flow chart of the overall algorithm is shown in Fig. 6.

5 Experiments

The proposed MSNT algorithm is realized in MATLAB® on the experimental platform of a CPU (Intel Xeon 2.4 GHz) and GPU (TITAN X). The initial parameters of the MSNT are as follows: $\lambda = 0.005$, $\rho = 0.05$, $\mu = 0.2$, $\eta = 50$, and $\tau = 0.9$. In addition, we set the learning rate ξ to 0.01 during the online training. The weight matrix W is randomly initialized

$$W_{i,i+1} = \frac{\text{rand}(n_i, n_{i+1}) - 0.5}{\sqrt{n_{i+1}}}, \tag{11}$$

where $W_{i,i+1} \in W$ denotes the weight matrix between the i'th layer and the $(i+1)$'th layer, n_i and n_{i+1} denote the number of the neurons of the i'th and the $(i+1)$'th layer, and $\text{rand}(n_i, n_{i+1})$ generates a random matrix of $n \times n$ with uniform distribution between 0 and 1. Therefore, W is randomly initialized into $[-0.5, 0.5]$ of the microweights, and the weights are different in different layers.

To verify the validity of our proposed method, the one-pass evaluation (OPE) as in Ref. 16 is used in our experiments. The MSNT algorithm is evaluated on the tracking benchmark dataset,[16] which includes 51 fully annotated videos. We compare the performance of our tracker with nine state-of-the-art trackers, including DLT,[22] CNT,[25] kernelized correlation filters (KCF),[29] tracking with Gaussian processes regression (TGPR),[30] sparsity-based collaborative model (SCM),[31] Struck,[32] structural sparse tracking (SST),[33] linearization to nonlinear learning tracker (LNLT),[34] and circulant sparse tracker (CST).[35] A brief introduction of these referenced trackers is shown in Table 2, and their tracking results are provided by their authors. Some qualitative and quantitative comparisons are implemented to evaluate the performance of our tracker. The detailed and color comparisons can be obtained in the online version of this paper.

5.1 *Qualitative Comparisons*

In qualitative comparisons, eight challenging sequences are selected to evaluate the MSNT intuitively. The results are shown in Fig. 7, and the different colors indicate different trackers. Then, we analysis the trackers qualitatively from the following aspects:

1. Illumination variation: Taking the video of "Coke" for an example, when the illumination changes dramatically, MSNT, TGPR, and Struck can always track the target correctly, but the others lose the target easily.
2. Scale variation: Taking the videos of "Car4" and "Singer1" for examples, MSNT can adapt to the scale variation of the target, but KCF, Struck, TGPR, and CST cannot adjust the size of the bounding

Table 2 Brief introduction to nine referenced trackers.

	CST	SST	LNLT	KCF	CNT	TGPR	DLT	SCM	Struck
Year	2016	2015	2015	2015	2015	2014	2013	2012	2011
Source	CVPR	CVPR	ICCV	TPAMI	TIP	LNCS	NIPS	CVPR	ICCV
Basic method	SC	SC	SC	KCF	DL (CNN)	GPR	DL (AE)	SC	SVM

Note: For basic method, SC, sparse coding; KCF, kernelized correlation filter; DL, deep learning; CNN, convolutional neural network; AE, autoencoder; GPR, Gaussian processes regression; and SVM, support vector machine.

box adaptively; the tracking drifting even appeared for TGPR in Singer1.

3. Occlusion: It indicates the full or partial occlusion of the target by background or other objects. For the #78 frame in "Jogging-1," when the full occlusion disappears gradually, only MSNT and LNLT can track the target immediately and accurately. For the partial occlusion in "Tiger1," only MSNT can track the target from beginning to the end.
4. Fast motion: For the target in "Deer," the motion of the target is fast and even causes the blur of the target region. The KCF, DLT, SST, SCM, and LNLT fail to track the target when the target moves too fast, but MSNT can always track the target very well.
5. Background clutter: In "Girl," the tracking drift arises in KCF, TGPR, CST, and LNLT when a background similar to the target appears in the tracking region, such as #441 and #470, while MSNT can successfully track the target.
6. LR: For targets of LR, such as "Freeman4," the information of the target is too small to extract enough features. Due to the overcomplete layer appended to the "LR-target" tracking network, MSNT captures more available features to track the target robustly.
7. Rotation: It is divided into in-plane and out-of-plane rotation. Both rotations are in Girl in which MSNT tracks the target consistently and the sizes of the bounding boxes match the target well.

5.2 *Quantitative Comparisons*

To evaluate our tracker comprehensively and reliably, we use four quantitative evaluation metrics, which are introduced in Ref. 17, to carry out quantitative analysis.

1. Overlap rate: Given the ground-truth bounding box S_G and the tracked bounding box S_T, the overlap rate is defined as $\alpha = \frac{|S_T \cap S_G|}{|S_T \cup S_G|}$, where $\cap$ and $\cup$ represent the intersection and union of two regions, respectively, and $|\cdot|$ denotes the area of the region. The larger value of the overlap rate indicates a better performance of the tracker.
2. Center location error (CLE): It is defined as the Euclidean distance between the center locations of the tracked results and the manually annotated ground truths. The smaller value of the CLE indicates a better performance of the tracker.
3. Success rate: Success rate is associated with the overlap rate α. Given a threshold t_0, the targets are considered to be tracked successfully if and only if $\alpha > t_0$. The success rate is defined as the percentage of the successful frames, and the larger value indicates a better performance of the tracker.
4. Precision: Precision shows the ratio of frames whose CLE is within a given threshold, and the larger value indicates a better performance of the tracker.

In our experiments, we quantitatively analyze our tracker from three aspects: the tracking performance for a single sequence, the overall performance, and the attribute-based performance for 51 sequences.[17]

5.2.1 *Tracking performance for a single sequence*

The above eight challenging sequences, which are introduced in Sec. 5.1, are used to compare the tracking performance of a single sequence quantitatively.

Figure 8 and Table 3 show the overlap rate plots and the success rate in the success threshold $t_0 = 0.5$, respectively, of the 10 trackers on eight challenging sequences. From Fig. 8, we see that the overlap rates of our tracker are always at a high level in these eight challenging sequences, and the success rates of our tracker in Table 3 are higher than most other trackers. These metrics prove that our tracker achieves a good tracking success rate for single sequences in different challenging scenes.

Figure 9 and Table 4 show the CLE plots and the average CLEs of the trackers, respectively, on eight challenging sequences. In the tracking process for a single sequence, our tracker maintains lower center errors compared to others and achieves a low tracking error for the whole sequence. These quantitative metrics show that our tracker possesses a higher precision during the tracking process.

5.2.2 *Overall performance for 51 sequences*

For evaluating our tracker's overall performance for 51 sequences in the benchmark,[17] we plot the success plots and the precision plots of the above 10 trackers. The success plot shows the success rates at a varied overlap threshold t_0 in the interval [0, 1], and the precision plot shows the precisions at a varied CLE threshold from 0 to 50 pixels. Furthermore, to verify the effectiveness of the multiscale tracking networks of our tracker, a single network tracker based on the S-target network, named single network-based tracker (SNT) algorithm, is used for comparison.

Figure 10 shows the overall performance comparisons of 11 trackers based on success plots and precision plots. These trackers are ranked according to the area under curve (AUC) values of success plots in Fig. 10(a) and

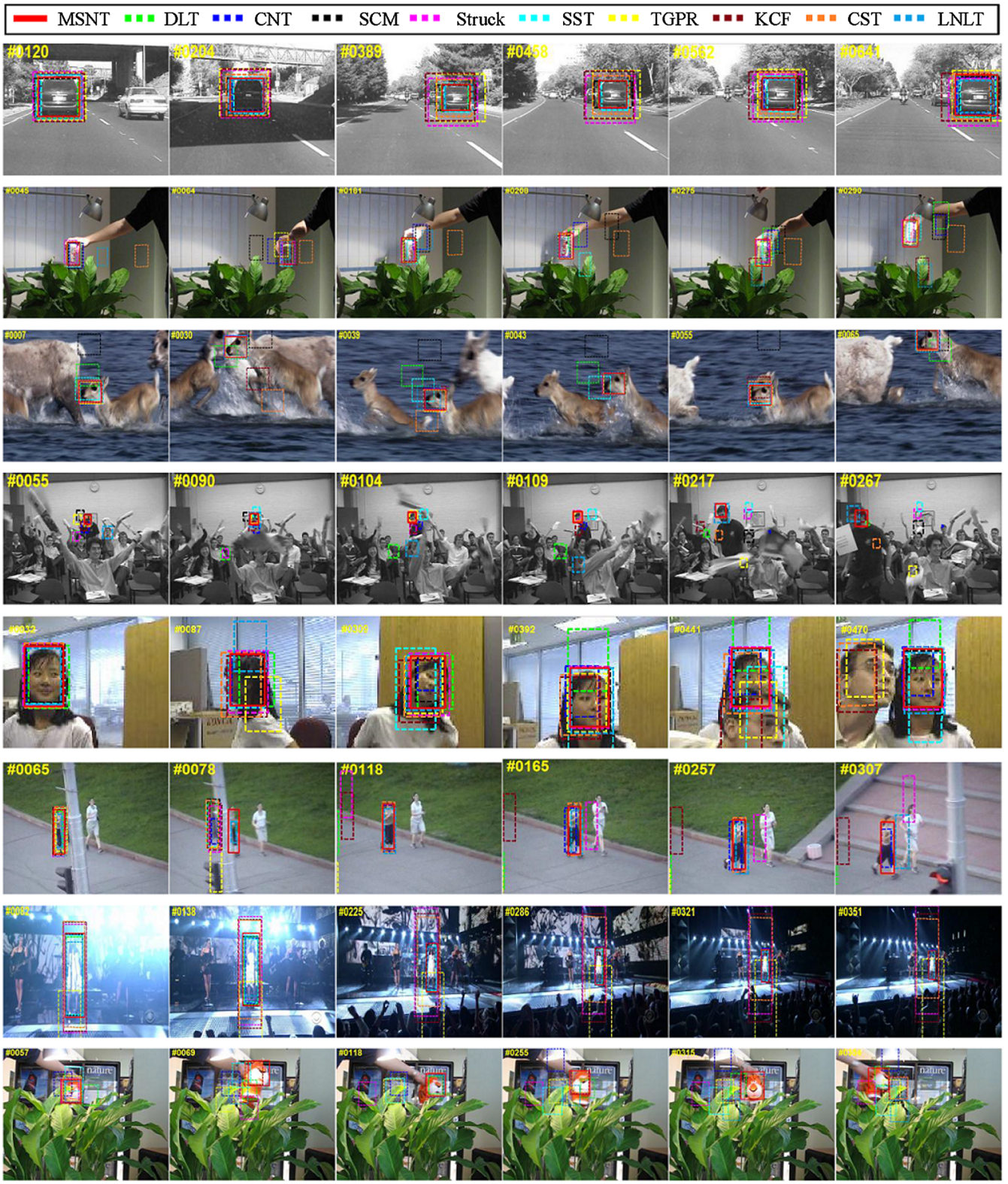


Fig. 7 Qualitative comparison of 10 trackers (denoted in different colors and lines) on eight challenging sequences (from top to bottom are Car4, Coke, Deer, Freeman4, Girl, Jogging-1, Singer1, and Tiger1).

the precision values at the threshold of 20 pixels in Fig. 10(b). For success plots, MSNT achieves the AUC value of 0.564 and ranks first of 11 trackers. Compared with DLT and CNT, which are also based on the deep learning method, the value of MSNT is improved by 29.4% and 4.0% over these, respectively. For precision plots, the precision of MSNT achieves 0.753 and ranks second, which is only after CST of 0.777. Similarly, the precision of MSNT is increased by 28.3% and 4.1% more than DLT and CNT, respectively.

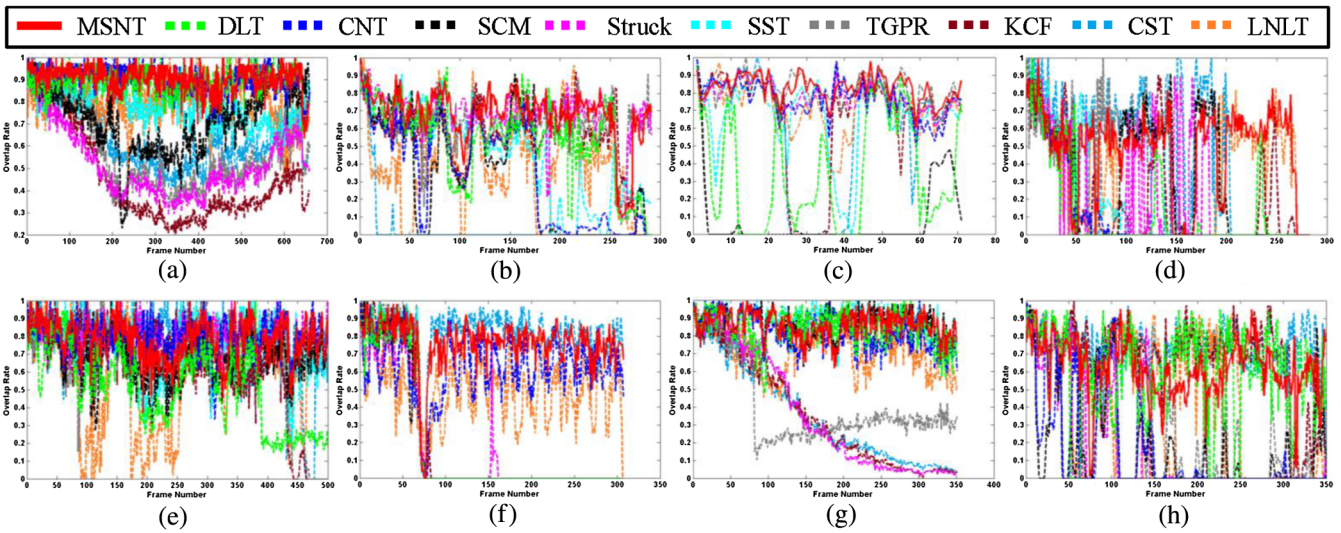


Fig. 8 Overlap rate plots of 10 trackers on eight challenging sequences. (a) to (h) Car4, Coke, Deer, Freeman4, Girl, Jogging-1, Singer1, and Tiger1, respectively.

Table 3 The success rates in the success threshold $t_0 = 0.5$ of the trackers on eight challenging sequences.

	MSNT	DLT	CNT	SCM	Struck	SST	TGPR	KCF	LNLT	CST
Car4	**100.00**	**100.00**	**100.00**	95.30	42.64.	**100.00**	51.75	26.25	*99.09*	86.19
Coke	*92.78*	67.35	43.30	35.05	**94.16**	50.86	89.35	72.16	28.52	4.12
Deer	**100.00**	38.03	*98.59*	2.82	**100.00**	85.92	**100.00**	81.69	94.37	71.83
Freeman4	**64.31**	14.13	13.07	39.93	26.86	20.14	35.34	19.43	37.81	*49.47*
Girl	*98.80*	52.60	98.60	90.00	**100.00**	90.00	86.60	82.90	67.60	92.40
Jogging-1	**97.07**	22.48	*79.80*	21.17	21.82	22.15	22.48	22.48	52.12	**97.07**
Singer1	**100.00**	*99.43*	**100.00**	**100.00**	36.18	**100.00**	23.08	35.04	95.73	30.77
Tiger1	80.23	67.05	15.76	13.47	20.63	13.47	27.51	*87.39*	42.98	**94.56**
Average	**91.65**	57.63	58.64	49.72	57.09	60.32	54.51	53.42	64.78	*65.80*

Note: %, the best results are in bold and the second best in italics.

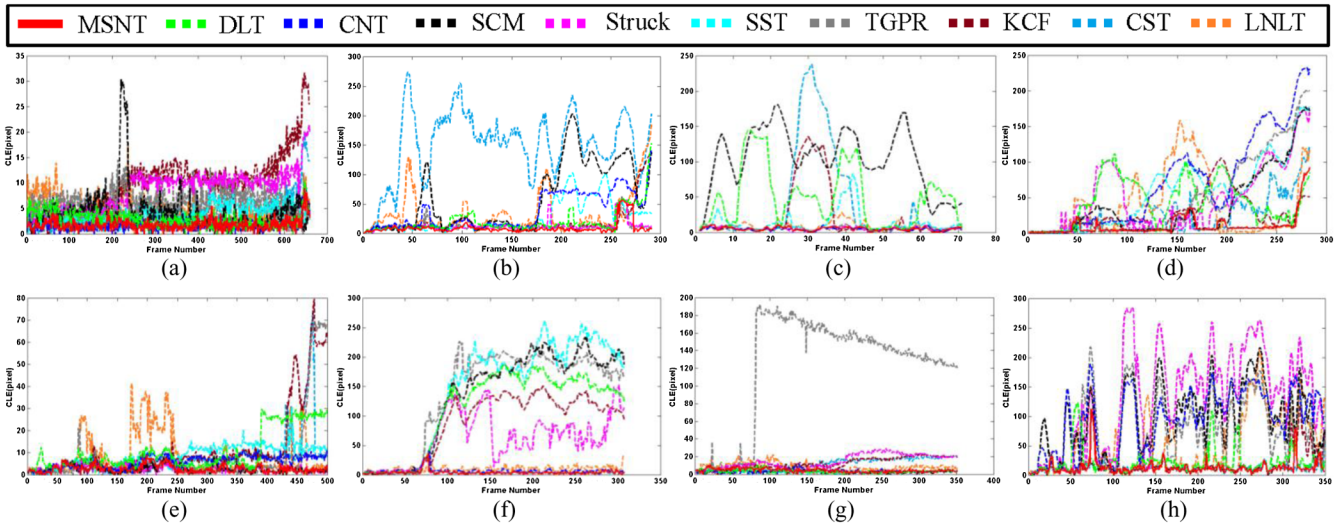


Fig. 9 CLE plots of the 10 trackers on eight challenging sequences. (a) to (h) Car4, Coke, Deer, Freeman4, Girl, Jogging-1, Singer1, and Tiger1, respectively.

Table 4 Average CLEs of the trackers on eight sequences.

	MSNT	DLT	CNT	SCM	Struck	SST	TGPR	KCF	LNLT	CST
Car4	*1.78*	2.78	**1.51**	4.27	8.63	3.75	6.11	9.47	4.33	3.73
Coke	**9.00**	20.13	36.67	56.81	12.08	25.94	*11.44*	18.65	32.84	148.66
Deer	*4.98*	49.13	**4.60**	103.54	5.17	13.84	5.85	21.27	8.56	39.58
Freeman4	**10.91**	45.12	70.37	37.67	48.63	56.20	48.06	26.89	38.12	*22.48*
Girl	2.63	10.51	5.74	*2.60*	**2.58**	8.45	7.70	11.92	8.31	7.43
Jogging-1	**3.77**	113.02	6.19	132.83	62.03	144.61	137.46	87.90	9.35	*3.92*
Singer1	4.37	3.37	3.45	**2.72**	14.53	*2.78*	120.29	12.59	8.02	10.90
Tiger1	*12.55*	23.22	94.17	93.49	128.70	93.49	72.68	15.66	53.61	**11.27**
Average	**6.25**	33.41	27.84	54.24	35.29	43.63	51.20	25.54	*20.39*	31.00

Note: Pixels, the best results are in bold and the second best in italics.

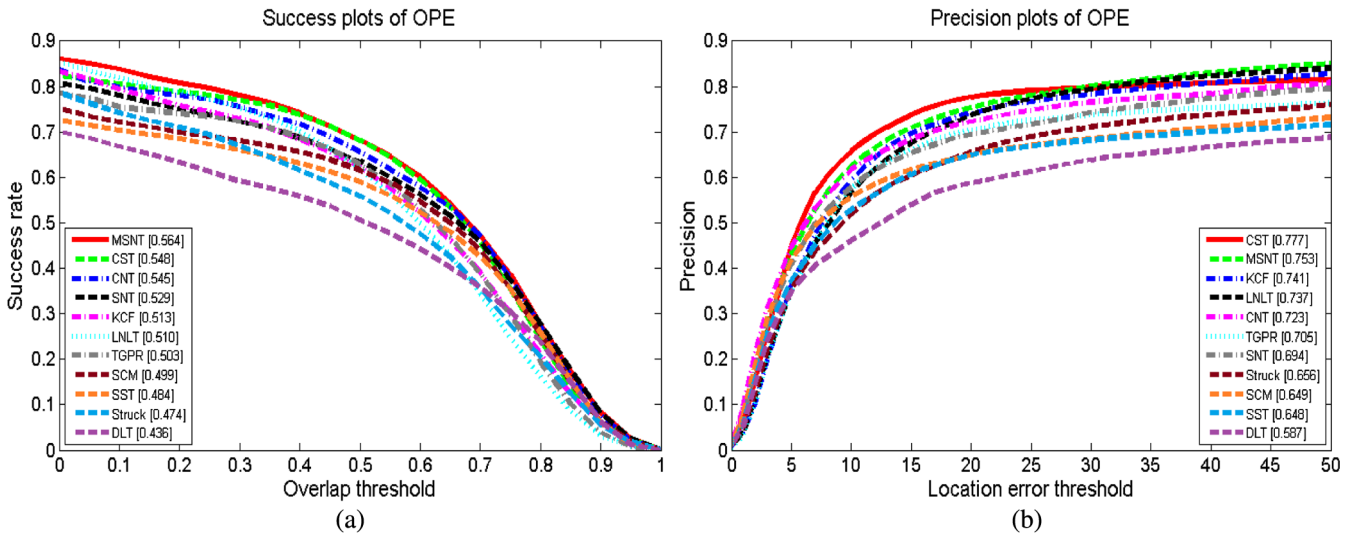


Fig. 10 The success plots and precision plots of OPE for the trackers: (a) success plots and (b) precision plots.

Analyzing the success plots and precision plots of MSNT and SNT, we find that MSNT improves the performance of SNT apparently in both of these metrics. The MSNT improves the value by 6.6% more than SNT in the success plots and improves the precision by 8.5% more than SNT. These results suggest that our proposed multiscale networks can extract more robust and effective features and have better performance than the single and fixed network.

These experimental data and the above analyses illustrate that our tracker outperforms these state-of-the-art trackers and achieves satisfactory tracking results in different challenging scenarios.

5.2.3 *Attribute-based performance for 51 sequences*

To further analyze the performance of our tracker under different tracking conditions, we evaluated these trackers on 11 attributes, which are defined in Ref. 17. The success plots and precision plots on different attributes are shown in Figs. 11 and 12, respectively. Among the 11 attributes, MSNT ranks first in eight attributes (including "illumination variation," "out-of-plane rotation," "scale variation," "occlusion," "fast motion," "in-plane rotation," "out of view," and "LR") and outperforms SNT in all attributes in Fig. 9. Only on the attributes of "deformation" and "background clutter" does MSNT not rank in the top 3 in the success plots.

For the precision plots in Fig. 12, MSNT ranks in the top 3 on eight attributes and outperforms SNT in all attributes. In particular, in the attributes of fast motion, out of view, and LR, MSNT has the best performance for the tracking precisions. However, MSNT has a worse performance on the attributes of illumination variation, deformation, and background clutter than some trackers, such as CST, KCF, and LNLT.

Some observations we obtained from these attribute-based data: first, our tracker achieves a good performance

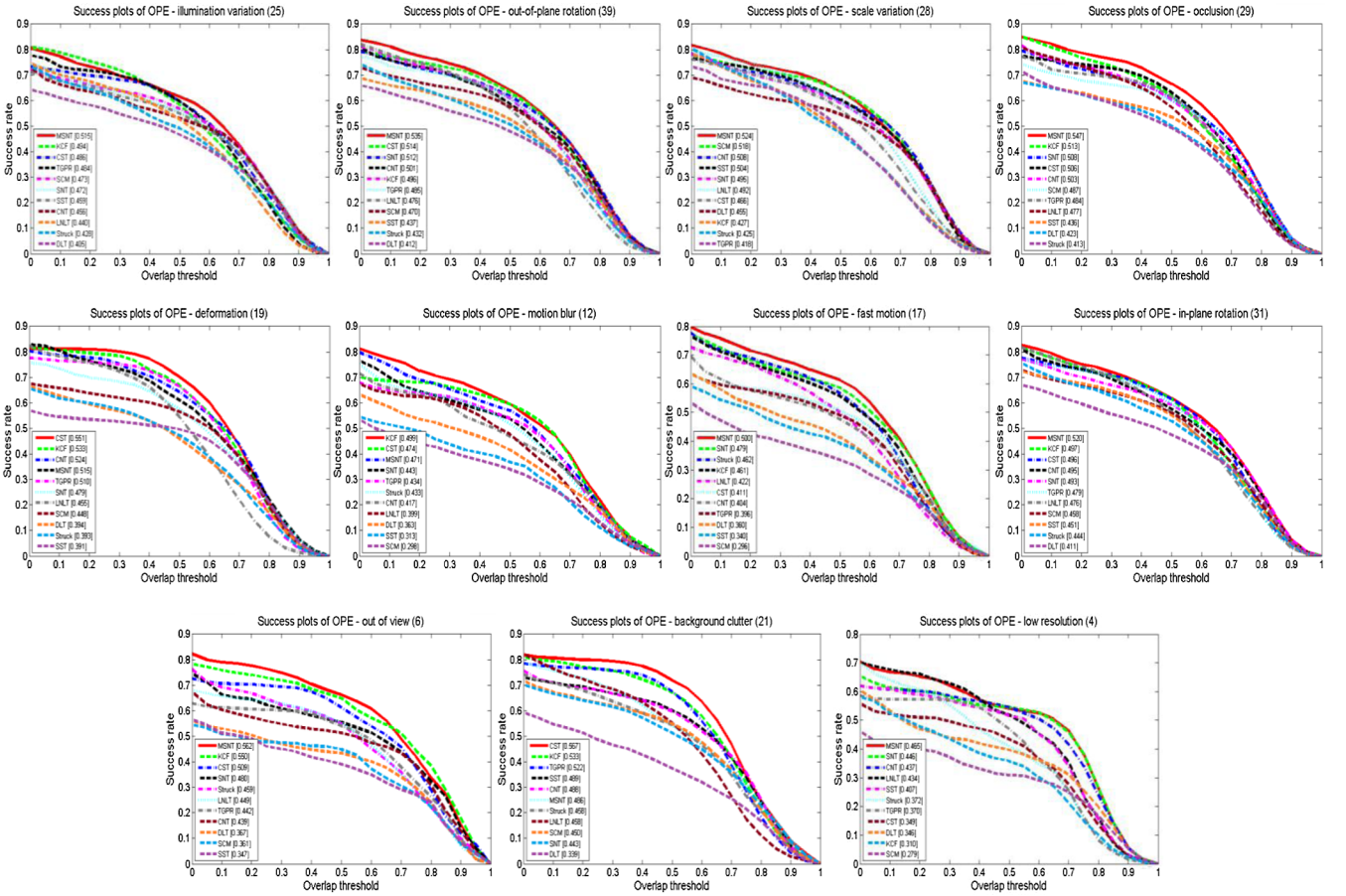


Fig. 11 The success plots of OPE for the trackers on different attributes.

in most attributions, especially in the attributes of fast motion, out of view, and LR. Second, our tracker cannot perform as well as CST and KCF on some attributes, such as deformation and background clutter, especially in the precision plots. These may be the next research areas for improving our tracker. Third, MSNT outperforms SNT in all attributes whether in success plots or precision plots. It further proves the availability of a multiscale tracking network.

5.3 *Tracking Speed of Tracker*

On our experimental platform, our tracker achieves a practical tracking speed of an average of 13.2 frame per second (FPS) for the 51 sequences. Table 5 shows the tracking speed of the above 10 trackers. All the data are published by the authors in their papers. The "—" indicates that the author does not give the tracking speed explicitly. From Table 5, we see that KCF has the highest tracking speed, and our tracker achieves a faster speed than CST, SST, TGPR, and SCM. Compared to DLT, which is also based on deep learning, our tracker has a slightly slower tracking speed, but it avoids the complex and time-consuming pretraining process. This property makes the establishment and adjustment of tracking networks more simple and flexible than DLT.

5.4 *Discussion*

For a more thorough evaluation, we also add the following recent trackers with their corresponding results (success rate, precision, and FPS) to the comparison: STCT (0.640, 0.780, 2.5),[36] RTT (0.588, 0.827, 3 to 4),[37] and DLRT (0.512, 0.694, 3).[38] Among these trackers, the proposed MSNT (0.564, 0.753, 13.2) achieves better performance than DLRT but is inferior to STCT and RTT. Nevertheless, our tracker has a faster processing speed than these trackers and achieves comparable performance as RTT in success rate and as STCT in precision. However, our tracker still has room to improve compared with the best tracker STCT. STCT regards CNN as an ensemble of base learners and trains the convolutional layers with random binary masks. These techniques reduce the correlation between the learned features and prevent overfitting effectively, although these lead to higher computation cost to some degree. Like the random binary masks in STCT, the similar trick, "Dropout,"[39] may be used in our tracker to further avoid overfitting.

In this paper, we propose a simple but effective MDSN for achieving real-time tracking. A robust tracker is built based on MDSN without offline pretraining with an auxiliary dataset, and the tracker alleviates the "gradient vanishing" in the training process due to the ReLU activation function. However, as shown in Fig. 13, there are some serious failed cases for our MSNT tracker. In "Bolt," the runners have similar appearances, so it is difficult to discriminate the correct target from the others. In "Ironman," the comprehensive factors (including intense lighting changes, similar background, fast motion, and rotation, etc.) make the tracker be unable to differentiate the dark target from the noisy background effectively. Finally, in "MotorRolling," the significant rotation

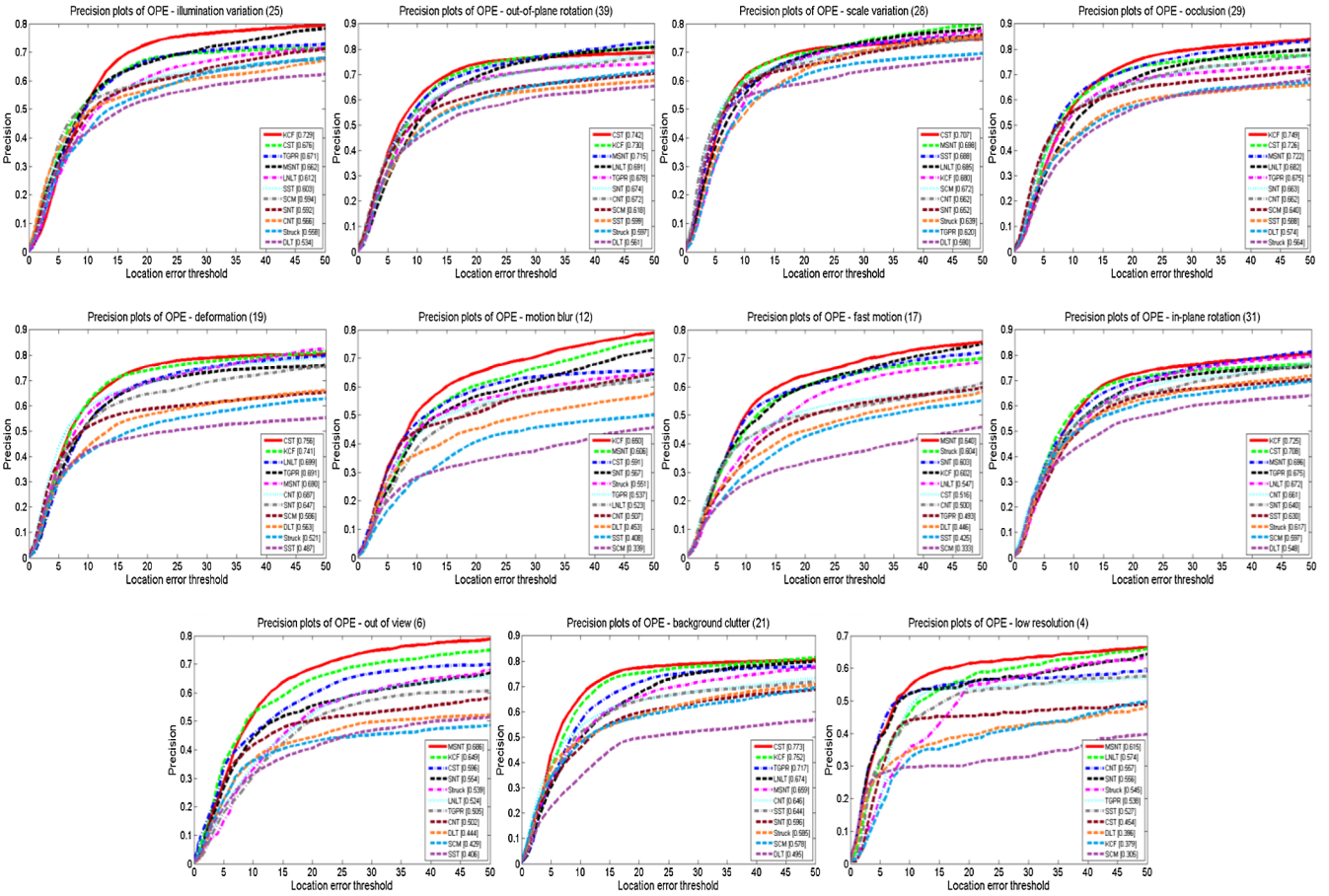
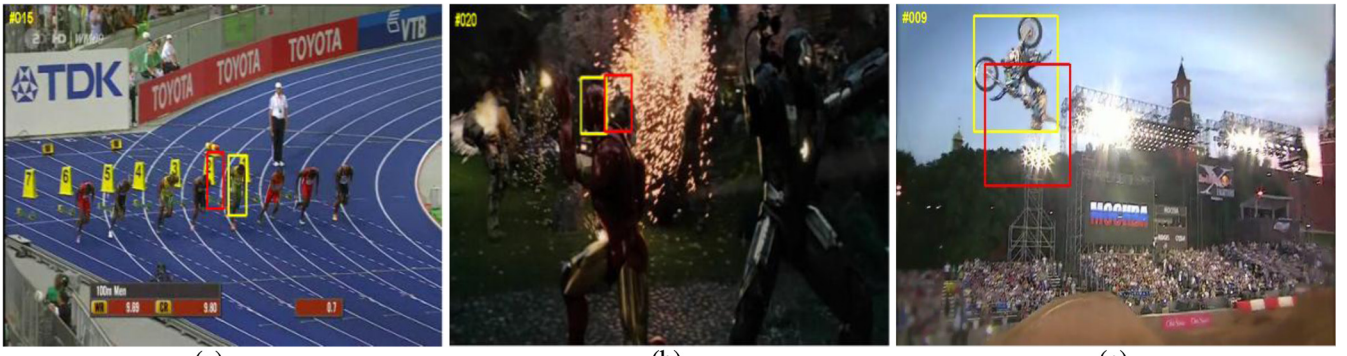


Fig. 12 The precision plots of OPE for the trackers on different attributes.

Table 5 The tracking speed comparison for the 10 trackers.

Tracker	MSNT	CST	SST	LNLT	KCF	CNT	TGPR	DLT	SCM	Struck
FPS	13.2	2.2	2.2	—	172	—	3 to 4	15	0.5	20.2



(a) (b) (c)

Fig. 13 Some failure cases for our tracker. Red boxes show our results and the yellow ones are the ground truth. (a) Bolt, (b) Ironman, and (c) MotorRolling.

and deformation of the target cause the tracking failure. In these cases, our tracker easily makes the biggest errors, i.e., the tracking drifting and target missing at the beginning of the tracking.

Analyzing these failed cases, the deformation and background clutter of targets may be the main factors to cause failure for our proposed MSNT. Moreover, the rankings of our tracker in Figs. 11 and 12 also indicate that MSNT has a relatively poor performance on the attributes of deformation and background clutter. In addition, the above trick for preventing overfitting, such as in Dropout, can improve the performance of our tracker to some degree;

a combination with more robust and semantic feature extractors, such as CNNs (in Ref. 36) or RNNs (in Ref. 37), may be potential solutions to our method on both challenging attributes. These problems will be the interesting research directions in our future work.

6 Conclusions

In this paper, we proposed an MDSN for extracting robust and powerful features for visual tracking. The intrinsic sparsity of the networks avoids offline pretraining with an auxiliary image dataset and exploits more sparse and robust feature representations. The multiscale networks can adaptively select the corresponding tracking networks based on the shapes of targets. It will capture more useful structural information of targets. Combined the MDSN with the particle filter tracking framework, the MSNT tracker is proposed to solve the tracking problems. Through quantitative and qualitative comparison with state-of-the-art trackers on the challenging tracking benchmark dataset, our proposed tracker achieves a satisfactory result and practicable tracking speed in experiments.

Furthermore, there are several possible directions to investigate in detail for this work. First, the technique of blocking, such as histograms of oriented gradients (HOG)[40] can be used in the proposed method to improve the performance on the attributes of deformation and background clutter. Second, CNNs and other feature extractors may be combined in our proposed method to exploit more robust and semantic features for tracking. Third, more effective classification methods, such as support vector machine, will be employed instead of softmax classifier, which may further improve the robustness of tracking.

Acknowledgments

This research has been supported by the National Natural Science Foundation of China (No. 61473309) and project supported by the Natural Science Basic Research Plan in Shaanxi Province (Nos. 2015JM6269 and 2016JM6050). We surely declare that no financial interests or conflicts are involved in this article.

References

1. A. W. M. Smeulders et al., "Visual tracking: an experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **36**(7), 1442–1468 (2014).
2. A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Comput. Surv.* **38**(4), 13 (2006).
3. X. Li et al., "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.* **4**(4), 58 (2013).
4. D. A. Ross, J. Lim, and R. S. Lin, "Incremental learning for robust visual tracking," *Int. J. Comput. Vision* **77**(1–3), 125–141 (2008).
5. T. Z. Zhang et al., "Robust visual tracking via multi-task sparse learning," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'12)*, Vol. 157, pp. 2042–2049 (2012).
6. X. Jia, H. Lu, and M. H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'12)*, Vol. 157, pp. 1822–1829 (2012).
7. K. H. Zhang, L. Zhang, and M. H. Yang, "Real-time compressive tracking," in *European Conf. on Computer Vision*, Vol. 7574, pp. 864–877, Springer (2012).
8. Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(7), 1409–1422 (2012).
9. B. Babenko, M. H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(8), 1619–1632 (2011).
10. Y. Lecun, Y. Bengo, and G. Hinton, "Deep learning," *Nature* **521**(7553), 436–444 (2015).
11. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics (AISTATS'10)*, Vol. 9, pp. 249–256 (2010).
12. G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science* **313**(5786), 504–507 (2006).
13. P. Vincent et al., "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.* **11**(12), 3371–3408 (2010).
14. Y. Zhang, E. H. Zhang, and W. J. Chen, "Deep neural network for halftone image classification based on sparse auto-encoder," *Eng. Appl. Artif. Intell.* **50**(1), 245–255 (2016).
15. V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. of the 27th Int. Conf. on Machine Learning (ICML'10)*, pp. 807–814 (2010).
16. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. of the 14th Int. Conf. on Artificial Intelligence and Statistics (AISTATS'11)*, Fort Lauderdale, Florida, Vol. 15, pp. 315–323 (2011).
17. Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: a benchmark," in *Proc. IEEE Computer Vision and Pattern Recognition*, Vol. 9, pp. 2411–2418 (2013).
18. J. Li, H. Chang, and J. Yang, "Sparse deep stacking network for image classification," in *Proc. Twenty-Ninth AAAI Conf. on Artificial Intelligence (AAAI'15)*, pp. 3804–3810 (2015)
19. M. Baccouche et al., "Deep learning of split temporal context for automatic speech recognition," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'14)*, pp. 5459–5463 (2014).
20. O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. British Machine Vision*, Vol. 41, pp. 1–12 (2015).
21. J. Fan et al., "Human tracking using convolutional neural networks," *IEEE Trans. Neural Networks* **21**(10), 1610–1623 (2010).
22. N. Y. Wang and D. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Advances in Neural Information Processing Systems (NIPS'13)*, pp. 809–817 (2013).
23. H. Li, Y. Li, and F. Porikli, "Robust online visual tracking with a single convolutional neural network," in *Asian Conf. on Computer Vision, Proc.*, Vol. 9007, pp. 194–209, Springer (2014).
24. L. Wang et al., "Video tracking using learned hierarchical features," *IEEE Trans. Image Process.* **24**(4), 1424–1435 (2015).
25. K. H. Zhang, Q. S. Liu, and Y. Wu, "Robust visual tracking via convolutional networks without training," *IEEE Trans. Image Process.* **25**(6), 1779–1792 (2016).
26. D. Arpit et al., "Why regularized auto-encoders learn sparse representation?" in *Proc. Int. Conf. on Machine Learning (ICML'15)*, pp. 134–144 (2015).
27. J. Li et al., "Sparseness analysis in the pretraining of deep neural networks," *IEEE Tran. Neural Networks Learn. Syst.* **PP**(99), 1–14 (2016).
28. F. S. Wang, "Particle filters for visual tracking," in *Proc. Advanced Research on Computer Science and Information Engineering*, Vol. 152, pp. 107–112, Springer (2011).
29. J. F. Henriques et al., "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.* **37**(3), 583–596 (2015).
30. J. Gao et al., "Transfer learning based visual tracking with Gaussian processes regression," *Lect. Notes Comput. Sci.* **8691**(1), 188–203 (2014).
31. W. Zhong, H. Lu, and M. H. Yang, "Robust object tracking via sparsity-based collaborative model," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'12)*, pp. 1838–1845 (2012).
32. S. Hare, A. Saffari, and P. H. Torr, "Struck: structured output tracking with kernels," in *IEEE Int. Conf. on Computer Vision (ICCV'11)*, pp. 263–270 (2011).
33. T. Z. Zhang et al., "Structural sparse tracking," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'15)*, pp. 150–158 (2015).
34. B. Ma et al., "Linearization to nonlinear learning for visual tracking," in *IEEE Int. Conf. on Computer Vision (ICCV'15)*, pp. 4400–4407 (2015).
35. T. Z. Zhang, A. Bibi, and B. Ghanem, "In defense of sparse tracking: circulant sparse tracker," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'16)*, Vol. 1, pp. 3880–3888 (2016).
36. L. J. Wang et al., "STCT: sequentially training convolutional networks for visual tracking," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'16)*, Vol. 1, pp. 1373–1381 (2016).
37. Z. Cui et al., "Recurrently target-attending tracking," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'16)*, Vol. 1, pp. 1449–1458 (2016).
38. Y. Sui, Y. F. Tang, and L. Zhang, "Discriminative low-rank tracking," in *Proc. IEEE Int. Conf. on Computer Vision*, pp. 3002–3010 (2015).
39. G. E. Hinton et al., "Improving neural networks by preventing co-adaptation of feature detectors," *Comput. Sci.* **3**(4), 212–223 (2012).
40. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1, pp. 886–893 (2005).

Xin Wang received his BS degree from Air Force Engineering University (AFEU), Xi'an, China, in 2015. He is currently a master's candidate at the Information and Navigation College, AFEU. His current research interests include pattern recognition, computer vision, and machine learning.

Zhiqiang Hou graduated from AFEU and received his MS degree in 1998. He received his PhD from Xi'an Jiaotong University in 2005. He was a visiting scholar at the University of Bristol, UK, in 2009. He is currently a professor at AFEU. His research interests include pattern recognition, computer vision, image processing, and information fusion.

Wangsheng Yu received his MS degree and PhD in communication and information systems from the AFEU in 2010 and 2014, respectively. He is currently a lecturer at the Information and Navigation College, AFEU. His research interests include computer vision and image processing.

Yang Xue received his BS degree in information engineering from AFEU, Xi'an, China, in 2015. He is currently working toward his graduate degree in the group of Professor L. H. Ma. His research interests include quantum information and machine learning.

Zefenfen Jin received her BS degree from Hunan University, Changsha, China, in 2015. She is currently a master's candidate at the Information and Navigation College, AFEU. Her current research interests include image processing, pattern recognition, and visual tracking.

Bo Dai received his BS and MS degrees from AFEU in 2014 and 2016, respectively. His current research interests include computer vision, image processing, and machine learning.