Hindawi
Journal of Sensors
Volume 2017, Article ID 6471250, 6 pages
https://doi.org/10.1155/2017/6471250

Research Article

Visual Vehicle Tracking Based on Deep Representation and Semisupervised Learning

Yingfeng Cai,[1] Hai Wang,[2] Xiao-qiang Sun,[1] and Long Chen[1]

[1]*Automotive Engineering Research Institution, Jiangsu University, Zhenjiang, Jiangsu 212013, China*
[2]*School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, China*

Correspondence should be addressed to Hai Wang; wanghai1019@163.com

Received 21 December 2016; Revised 4 February 2017; Accepted 15 February 2017; Published 9 March 2017

Academic Editor: Oleg Sergiyenko

Discriminative tracking methods use binary classification to discriminate between the foreground and background and have achieved some useful results. However, the use of labeled training samples is insufficient for them to achieve accurate tracking. Hence, discriminative classifiers must use their own classification results to update themselves, which may lead to feedback-induced tracking drift. To overcome these problems, we propose a semisupervised tracking algorithm that uses deep representation and transfer learning. Firstly, a 2D multilayer deep belief network is trained with a large amount of unlabeled samples. The nonlinear mapping point at the top of this network is subtracted as the feature dictionary. Then, this feature dictionary is utilized to transfer train and update a deep tracker. The positive samples for training are the tracked vehicles, and the negative samples are the background images. Finally, a particle filter is used to estimate vehicle position. We demonstrate experimentally that our proposed vehicle tracking algorithm can effectively restrain drift while also maintaining the adaption of vehicle appearance. Compared with similar algorithms, our method achieves a better tracking success rate and fewer average central-pixel errors.

1. Introduction

Visual vehicle tracking is one of the key technologies used in active vehicle safety applications. It is used in advanced driver assistance systems (ADAS) and in intelligent vehicles. However, in real traffic situation, the camera and the vehicles need to be tracked are all in motion. The visual tracking of target vehicles is often affected by complex backgrounds, changes in illumination, and occlusion by other vehicles or objects. These factors make vehicle tracking a challenging task in real traffic scenarios.

Existing tracking algorithms generally fall into one of two categories: generative models and discriminative models. Discriminative models convert tracking problems into binary classification problems of target and background. They have attracted much research interest in recent years. Many representative methods have been proposed, such as the online AdaBoost algorithm [1], the multiple instance learning algorithm [2], the TLD (tracking-learning-detection) algorithm [3], and the SVT (support vector tracking) algorithm [4].

The biggest challenge for discriminative model based tracking methods is the tracker drifting problem. This is because only a small amount of labeled samples (e.g., there is only one positive sample in most cases) can be used to train the classifier. Additionally, the tracking of subsequent image sequences depends on the classification results of the former frame. So, if the tracked area of former frame is not locked onto the optimal target, this self-training approach can lead to classifier drifting, which causes accumulated errors and further tracking failures.

To solve this shortcoming of the self-training approach, a new method, named semisupervised learning-based tracking, has been proposed [5–7]. In this method, a large amount of auxiliary images is used to maintain a feature dictionary which is able to describe images. Then, the dictionary will be used to update the tracker online. However, this kind of method prefers to use pixels from the original image or handcrafted features (such as the Haar feature, HOG feature, or DIFT feature), to generate the feature dictionary. This

cannot satisfy the requirements of the image classification that is needed for robust tracking.

The newly developed deep learning framework is a bioinspired architecture which describes data such as images, voices, and text by mimicking the human brain's mechanisms of learning and analysis. Through deep learning, features are transformed from their original space in a lower layer to a new space in a higher layer [8]. Compared with handcrafted features, the automatic features generated by deep learning are more capable of expressing the details of internal properties of the data. Inspired by this, a semisupervised vehicle tracking algorithm is proposed that uses deep representation. The overall structure of the proposed method is shown in Figure 1. Firstly, a large number of unlabeled images are selected as training samples, and a 2D deep belief network (2D-DBN) is used as a classifier structure. Then, a multilayer deep network will be trained with those unlabeled samples and the nonlinear mapping node in the top layer will be taken as the feature dictionary. In the tracking process, the subimage containing the vehicle to be tracked in the initial frame is set as the positive sample, and the surrounding background subimages are set as negative samples. After that, an online deep tracker will be learned from the generated samples and the feature dictionary. Finally, a partial filter will be used to estimate and provide a small area for the tracker. The deep learning-based tracking algorithm is able to fully exploit the deep-level feature information embedded in the image. This effectively suppresses drift situations while keeping the vehicle tracking system updated.

Generally, compared with other semisupervised learning-based tracking methods which use handcrafted features [5–7], this work introduces deep learning framework to the traditional semisupervised learning. This work applies 2D-DBN deep model to generate features automatically in unsupervised offline training and then uses small amount number of samples to adjust the tracker online.

In this article, in Section 2, the establishment of the deep model based semisupervised tracking framework is introduced in detail which contains both offline and online training steps. The vehicle position prediction method based on partial filter is given in Section 3. In Section 4, the experiment results and its analysis will be shown. Finally, a brief conclusion of this work is given in Section 5.

2. Establishment of Vehicle Tracking Based on Deep Modeling

The establishment of deep tracking requires two steps (Figure 2). (1) Offline training session: a large number of unlabeled images are selected as training samples and the 2D-DBN is trained unsupervised. (2) Online training session: the labeled samples are refined online using the parameters of the 2D-DBN with a backpropagation algorithm.

2.1. Unsupervised Offline Training. In the unsupervised offline training process, many road images containing vehicles are selected and a large amount of unlabeled samples with $M * M$ pixels is generated from them. Then, all sample images

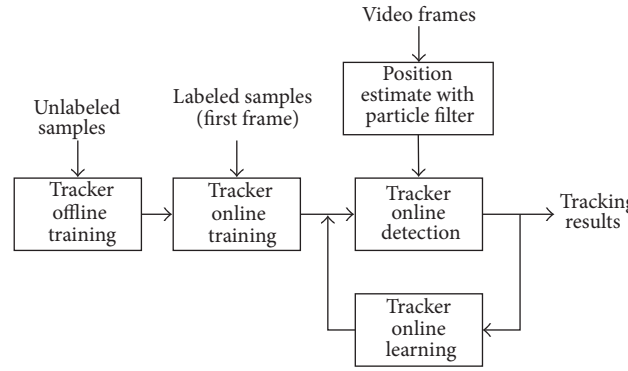

Figure 1: Overall structure of proposed vehicle tracking algorithm.

are converted to gray scale and normalized to the $[0, 1]$ interval. After that, the entire normalized matrix is input to the 2D-DBN to perform feature dictionary extraction.

The structure of the proposed 2D-DBN is shown in Figure 3. It is constructed with one visible layer, V^1, and two hidden layers, H^1 and H^2. The visible layer contains $M \times M$ units which is equal to the dimension of input samples. The hidden layers H^1 and H^2 contain $P \times P$ and $Q \times Q$ units, respectively. The visible layer and hidden layer are connected with a group of weights, θ. After unsupervised training to adjust the weights, the unit in hidden layer H^2 can be considered as the feature dictionary.

In unsupervised training, the greedy-wise reconstruction algorithm is used to adjust the weights between each two layers [9]. Taking visible layer V^1 and hidden layer H^1 as an example, V^1 and H^1 can be seen as a restricted Boltzmann machine (RBM). The state energy (v^1, h^1) of any two units of V^1 and H^1 can be written as

$$\begin{aligned} E\left(\mathbf{v}^1, \mathbf{h}^1, \theta^1\right) &= -\left(\mathbf{v}^1 \mathbf{A} \mathbf{h}^1 + \mathbf{b}^1 \mathbf{v}^1 + \mathbf{c}^1 \mathbf{h}^1\right) \\ &= -\sum_{i=1,j=1}^{i\le M, j\le M} \sum_{p=1,q=1}^{p\le P, q\le P} v_{ij}^1 A_{ij,pq}^1 h_{pq}^1 \\ &\quad - \sum_{i=1,j=1}^{i\le M, j\le M} b_{ij}^1 v_{ij}^1 - \sum_{p=1,q=1}^{p\le P, q\le P} c_{pq}^1 h_{pq}^1, \end{aligned} \tag{1}$$

where $\theta^1 = (\mathbf{A}^1, \mathbf{b}^1, \mathbf{c}^1)$ are the weight parameters of the units between visible layer V^1 and hidden layer H^1. $A_{ij,pq}^1$ is the connecting weight of unit (i, j) in visible layer V^1 and unit (p, q) in hidden layer H^1. b_{ij}^1 and c_{pq}^1 represent the bias between corresponding layers.

So, the RBM can be considered as a joint probability distribution:

$$P\left(\mathbf{v}^1, \mathbf{h}^1; \theta^1\right) = \frac{1}{Z} e^{-E(\mathbf{v}^1, \mathbf{h}^1; \theta^1)} = \frac{e^{-E(\mathbf{v}^1, \mathbf{h}^1; \theta^1)}}{\sum_{v^1} \sum_{h^1} e^{-E(\mathbf{v}^1, \mathbf{h}^1; \theta^1)}}, \tag{2}$$

where Z is a normalized parameter.

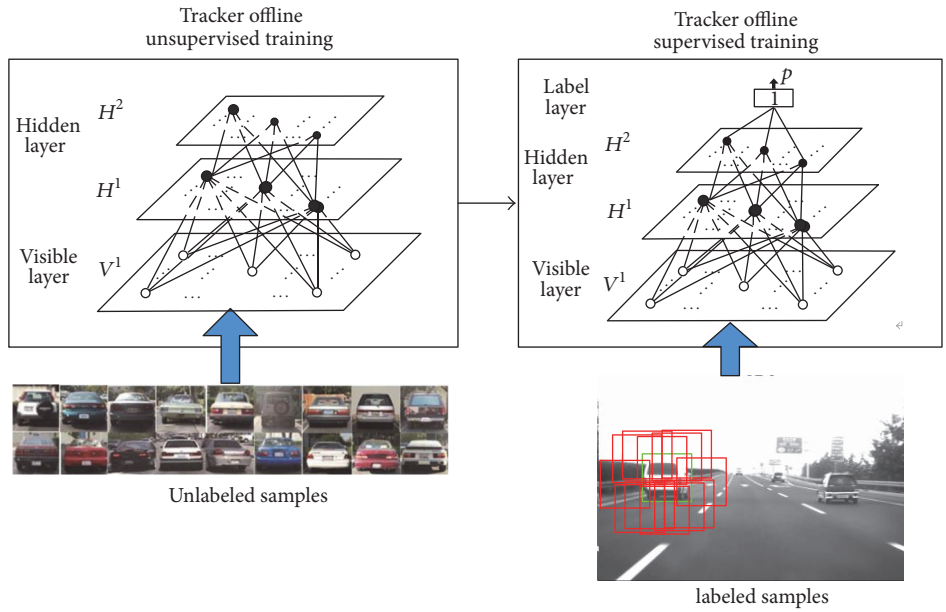

Figure 2: Overall structure of proposed semisupervised learning and deep model based method.

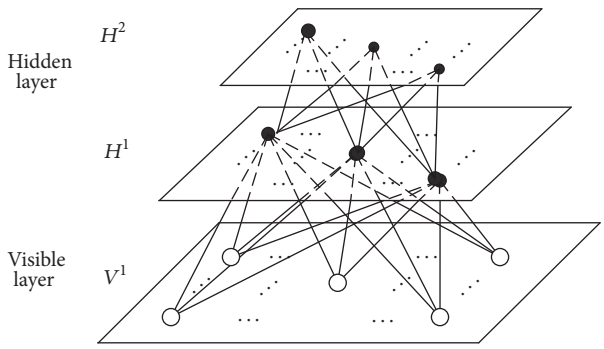

Figure 3: Structure of the deep belief network.

Then, the conditional probability distributions of input state $\mathbf{v}^1$ and hidden state $\mathbf{h}^1$ are able to be expressed with logistic functions:

$$p\left(\mathbf{h}^1 \mid \mathbf{v}^1\right) = \prod_{p,q} p\left(h_{pq}^1 \mid \mathbf{v}^1\right),$$
$$p\left(h_{pq}^1 \mid \mathbf{v}^1\right) = \sigma\left(\sum_{i=1,j=1}^{i\le M, j\le M} v_{ij}^1 A_{ij,pq}^1 + c_{pq}^1\right)$$
$$p\left(\mathbf{v}^1 \mid \mathbf{h}^1\right) = \prod_{i,j} p\left(\mathbf{v}_{ij}^1 \mid \mathbf{h}^1\right), \quad (3)$$
$$p\left(\mathbf{v}_{ij}^1 \mid \mathbf{h}^1\right) = \sigma\left(\sum_{p=1,q=1}^{p\le P, q\le P} h_{pq}^1 A_{ij,pq}^1 + b_{ij}^1\right),$$

where $\sigma(x) = 1/(1 + \exp(-x))$.

Based on the analysis above, the connecting weight and bias will be updated with a contrast divergence algorithm [9]:

$$A_{ij,pq}^1 = \vartheta A_{ij,pq}^1 + \varepsilon_A\left(\left\langle v_{ij}^1(0)\, h_{ij}^1(0)\right\rangle_{\text{data}} - \left\langle v_{ij}^1(t)\, h_{ij}^1(t)\right\rangle_{\text{recon}}\right) \quad (4)$$
$$b_{ij}^1 = \vartheta b_{ij}^1 + \varepsilon_b\left(v_{ij}^1(0) - v_{ij}^1(t)\right)$$
$$c_{pq}^1 = \vartheta c_{pq}^1 + \varepsilon_c\left(h_{pq}^1(0) - h_{pq}^1(t)\right),$$

where $\langle\cdot\rangle_{\text{data}}$ is the data's expected distribution, $\langle\cdot\rangle_{\text{recon}}$ is the reconstruction distribution, and $t = 1$ is the update step size.

By repeating these steps from lower layers to higher layers, the weights of (V^1, H^1) and (H^1, H^2) can be maintained. The weights of every unit in H^2 can then be considered as the nonlinear feature dictionary.

In real practice, many subimages captured from road videos that are vehicles will be input to the visible layer one by one. Then with the unsupervised training process given before, all the weights of the 2D-DBN will be adjusted and if we graphically display the weights in the top layer, the features are able to be viewed. Some of the features generated by the proposed method of unsupervised learning are shown in Figure 4. It can be seen from the feature that the features are more sensitive to the shape of edge, corner, and so on which exist more usually in vehicles.

2.2. Tracker Online Training. When the tracked area is identified in the first frame of a video, positive and negative samples are generated. Firstly, the tracked area will be chosen as

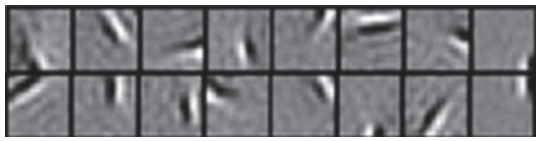

Figure 4: Example of features generated by unsupervised training.

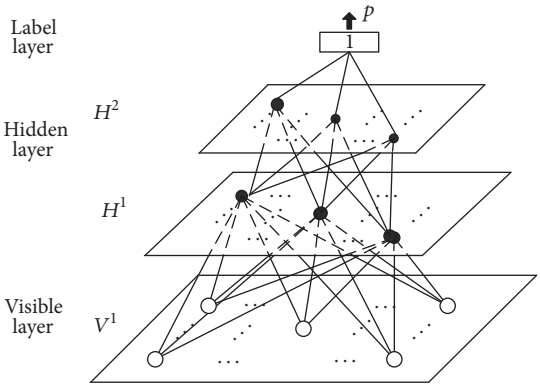

Figure 5: The structure of the proposed artificial neural network based on a DBN.

the positive sample and will then be rotated from −5 to 5 degrees with an intersection of 1 degree to generate more positive samples. Secondly, negative subimages are generated in a neighborhood ring area around the tracked area. The neighborhood area is defined as $\alpha < \|I_{\text{neg}} - I\| < \beta$, in which I_{neg} is the center of the negative samples, I is the center of the tracked area, and α and β are the inner and outer diameters of the ring area.

Based on the offline trained DBN, a label layer is introduced with a sigmoid function to generate a two-dimensional forward artificial neural network (Figure 5). In the online training process, the positive and negative samples and their labels are input to this neural network. The backpropagation algorithm is used to perform supervised training and adjust all the network weights. Finally, all the subimages in consequent frames are loaded to the newly updated network to make judgments.

3. Position Prediction Based on Particle Filtering

In the tracking process, the online trained classifier needs to verify a large number of subimages in consequent frames. To reduce processing time, a particle filter is used to estimate the target position.

The particle filter uses a Monte Carlo algorithm with Bayesian filtering and demonstrates good object tracking performance. Set $\mathbf{x}_t$ and $\mathbf{z}_t$ as the state and observation values of the target at time t. Then, the vehicle position estimate can be described as follows: in the condition of knowing observation value $\mathbf{Z}_t = \{\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t\}$ at time t, iteratively estimate the state of the system at time t.

Set the state space of system as

$$\begin{aligned} \mathbf{x}_t &= f(\mathbf{x}_{t-1}) + \mathbf{v}_{t-1} \\ \mathbf{z}_t &= h(\mathbf{x}_t) + \mathbf{n}_t, \end{aligned} \tag{5}$$

where f and h are the state transfer function and observation function, respectively, and $\mathbf{v}$ and $\mathbf{n}$ represent system noise and observation noise, respectively.

The filtering process contains two main steps: forecasting and updating. In forecasting process, the posterior probability density $P(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})$ at time $t-1$ is used to obtain the prior probability density at time t:

$$P(\mathbf{x}_t \mid \mathbf{z}_{t-1}) = \int P(\mathbf{x}_t \mid \mathbf{x}_{t-1}) P(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1}) \, d\mathbf{x}_{t-1}. \tag{6}$$

In the updating process, the newest system state observation value $\mathbf{z}$ at time t and the prior probability density $P(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})$ will be used to calculate the posterior probability density $P(\mathbf{x}_t \mid \mathbf{z}_t)$ at time t:

$$P(\mathbf{x}_t \mid \mathbf{z}_t) = \frac{\int P(\mathbf{z}_t \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{z}_{t-1})}{P(\mathbf{z}_t \mid \mathbf{z}_{t-1})}. \tag{7}$$

Assume that $\{\mathbf{x}_{0:t}^i, \omega_t^i\}_{i=1}^N$ is a group of samples with weights from the posterior probability density and $\sum \omega_t^i = 1$, $\mathbf{x}_{0:t} = \{\mathbf{x}_j,\ j = 0, 1, 2, \ldots, t\}$ is the sample set from time 0 to time t. Based on the Monte Carlo principle, the posterior probability density at time t is able to be approximated with a discrete weighting formula:

$$P(\mathbf{x}_{0:t} \mid \mathbf{z}_{1:t}) \approx \sum_{i=1}^{N} \omega_t^i \delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^i) \tag{8}$$

in which $\delta(\cdot)$ is a Dirac function.

Further, the estimated value of system state $\mathbf{x}_t$ at time t is

$$\widehat{\mathbf{x}}_t = \sum_{i=1}^{N} \omega_t^i \mathbf{x}_t^i. \tag{9}$$

4. Experiment and Analysis

The key parameters of the deep network and the particle filter are set out as follows: the number of units of the visible layer is 24×24 ($M = 24$); the number of units of the two hidden layers is 18×18 and 12×12 ($P = 18$, $Q = 12$); the initial training weights of offline training are $[0, 1]$, which satisfies the Gaussian distribution; the number of particles in the particle filter is $N = 1000$. For the unsupervised offline training, around 5000 subimages that only contain vehicle are selected for tracker training. Some of the typical images for training are shown in Figure 6.

In the experiments, different road scenarios were selected, including daytime, nighttime, and raining conditions. To evaluate performance, two criteria were used: the tracking success rate and center pixel offset. In these two criteria, tracking success is defined as $\text{area}(BB_T \cap BB_G)/\text{area}(BB_T \cup BB_G) \geq 50\%$ and center pixel offset is defined as the Euclidean

Figure 6: Some of the typical images for unsupervised offline training.

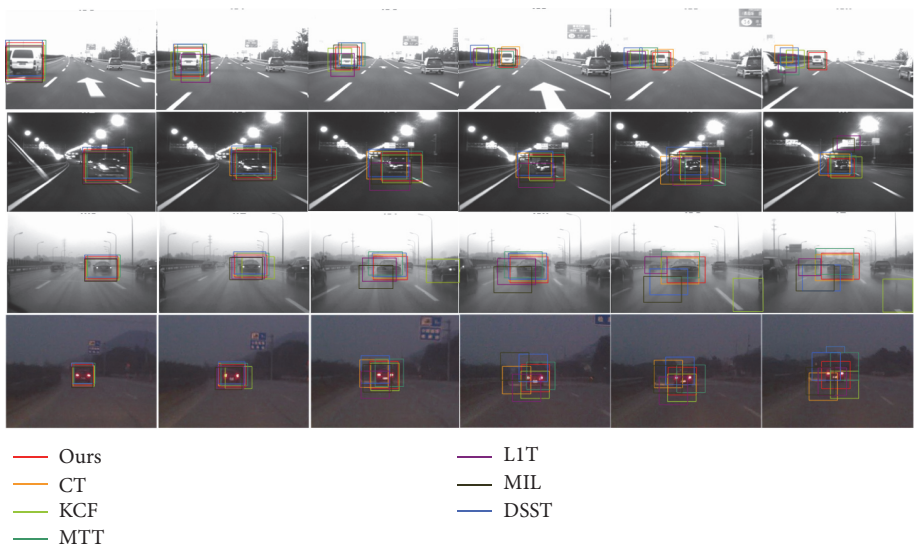

Figure 7: Vehicle tracking results of each algorithm in four typical scenarios.

distance between BB_T and BB_G, in which BB_T is the outline of the tracking box and BB_G is the ground truth of the real target outline box.

The proposed algorithm is comparable with many state-of-the-art algorithms such as MTT [10], CT [11], KCF [12], MIL [2], L1T [13], and DSST [14].

The experiment results are shown in Tables 1 and 2. It can be seen from the table that since the proposed tracking algorithm is able to generate richer deep features, the proposed method performs better than the existing state-of-the-art algorithms under the four different test conditions. The four typical scenarios that are chosen are daytime with good illumination, nighttime with road lamp, daytime with heavy raining, and twilight time. Examples of real tracking results are shown in Figure 7.

It should be mentioned that although the performance of our method is improved, the processing speed is relatively low. The average processing speed for one image is around 75 ms which is among the second worst of all the methods that are used in the experiments (Table 3). The possible reason for the big time cost is that the deep model contains a big amount of neurons and weight connections between each of neurons and the computing of weight in each connection may need much more processing time. The experimental platform was an Intel 2.67 GHz CPU with 4 GB RAM and the Windows 7 operating system.

5. Conclusion

To solve the problem of traditional tracking methods being not robust enough for vehicle tracking in ADAS, a deep representation and semisupervised onboard vehicle tracking algorithm was proposed. Relying on the strong feature extraction ability of deep modeling, this method dramatically inhibits drifting. Generally, the proposed semisupervised and deep model based tracking algorithm performs better than most of the existed methods in the merits of tracking success rate and average center pixel offset. However, the real-time performance of this work still needs to be further improved. There are two ways that are supposed to be used to solve this problem. First, concise deep model may be designed to reduce calculation burden. Second, parallel computing technology may be applied to speed the calculation process.

Competing Interests

The authors declare that they have no conflict of interests.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61601203, 61403172, and U1564201), China Postdoctoral Science Foundation (2015T80511 and

Table 1: Tracking success rate of each algorithm (%).

	MTT	CT	KCF	MIL	L1T	DSST	Our method
Video 1 (day 1)	92.4	64.7	83.2	69.7	75.8	50.2	**98.3**
Video 2 (night)	**89.6**	59.8	74.9	60.8	71.2	51.4	87.3
Video 3 (rain)	91.0	62.3	80.4	62.3	78.7	58.8	**93.5**
Video 4 (twilight)	80.4	59.8	69.9	56.6	70.5	60.7	**89.5**
Average	88.3	61.6	77.1	62.3	74.0	55.3	**92.1**

Table 2: The average center pixel offset of each algorithm (pixels).

	MTT	CT	KCF	MIL	L1T	DSST	Our method
Average center pixel offset	11.4	18.8	16.5	23.1	17.9	25.2	**8.3**

Table 3: The average processing time of each algorithm (pixels).

	MTT	CT	KCF	MIL	L1T	DSST	Our method
Average processing time (ms)	86	37	42	34	57	28	**75**

2014M561592), Key Research and Development Program of Jiangsu Province (BE2016149), Natural Science Foundation of Jiangsu Province (BK20140555), and Six Talent Peaks Project of Jiangsu Province (2015-JXQC-012 and 2014-DZXX-040).

References

[1] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proceedings of the 17th British Machine Vision Conference (BMVC '06)*, pp. 47–56, September 2006.

[2] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[3] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[4] S. Avidan, "Support vector tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.

[5] Q. Wang, F. Chen, J. Yang, W. Xu, and M.-H. Yang, "Transferring visual prior for online object tracking," *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3296–3305, 2012.

[6] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proceedings of the 10th European Conference on Computer Vision (ECCV '08)*, pp. 234–247, Springer, Marseille, France, October 2008.

[7] A. Teichman and S. Thrun, "Tracking-based semi-supervised learning," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 804–818, 2012.

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

[9] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[10] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 2042–2049, Providence, RI, USA, June 2012.

[11] K. Zhang, L. Zhang, and M. Yang, "Real-Time Compressive Tracking," in *Computer Vision—ECCV 2012*, vol. 7574 of *Lecture Notes in Computer Science*, pp. 864–877, Springer, Berlin, Germany, 2012.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[13] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 1830–1837, June 2012.

[14] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proceedings of the 25th British Machine Vision Conference (BMVC '14)*, Nottingham, UK, September 2014.