Convolutional Neural Network with Particle Filter Approach for Visual Tracking

Vladimir Tyan[1] and Doohyun Kim[1]
[1] Department of Software, Konkuk University,
120 Neungdong-ro, Gwangjin-gu, Seoul
[e-mail: auron@konkuk.ac.kr]
[e-mail: doohyun@konkuk.ac.kr]
*Corresponding author: Doohyun Kim

Received July 4, 2017; accepted August 11, 2017; published February 28, 2018

Abstract

In this paper, we propose a compact Convolutional Neural Network (CNN)-based tracker in conjunction with a particle filter architecture, in which the CNN model operates as an accurate candidates estimator, while the particle filter predicts the target motion dynamics, lowering the overall number of calculations and refines the resulting target bounding box. Experiments were conducted on the Online Object Tracking Benchmark (OTB) [34] dataset and comparison analysis in respect to other state-of-art has been performed based on accuracy and precision, indicating that the proposed algorithm outperforms all state-of-the-art trackers included in the OTB dataset, specifically, TLD [16], MIL [1], SCM [36] and ASLA [15]. Also, a comprehensive speed performance analysis showed average frames per second (FPS) among the top-10 trackers from the OTB dataset [34].

Keywords: Computer Vision, Object Tracking, Convolutional Neural Network, Particle Filter, GPU

https://doi.org/10.3837/tiis.2018.02.009 **ISSN : 1976-7277**

1. Introduction

The main goal of visual tracking is to track an object through a sequence of frames in a video stream with a given initial bounding box in the first frame. Originally the initial bounding box search is not a problem solved by tracking algorithms, usually it is marked by a human or received as the output of a detector. Further, tracker algorithms have no ability to redetect the target if the tracker fails or the target disappears and then reappears in the view area. Exceptions to these conventions are hybrid algorithms such as TLD [16] that include both tracker and detector parts inside.

The simplest example of a tracking algorithm is template matching [2], in which the scanning window estimates candidates nearby the target location in the previous frame by direct comparison of the previous frame target patch with candidate patches in the current frame, eventually choosing a candidate with the highest similarity score. However, it does not solve the main problem of visual tracking, which is the fact that target appearance is constantly changing owing to the influence of many factors such as illuminance, occlusion, motion blur, deformation, scale, and rotation. In order to overcome this problem, tracking algorithms need a tool that generalizes all possible target appearances and its changes. Typically, mathematical modeling is applied for this purpose. Thus, all trackers can be classified by a mathematical model used to represent a target and background during tracking.

Some algorithms use intensity histogram [25] or Haar primitives [3] representation to build a discriminative model that can be learned and used to distinguish good candidates from the background. TLD [16] algorithm uses three-stage candidates filtering: variance filter, ensemble of binary classifiers, and template matching based nearest neighbor search with discriminative modeling in the second and third stages. Another approach is to build a generative model via sparse representation [36] or by PCA [33]. In the Kernelized Correlation Filter (KCF) [13] algorithm, the object is represented by a kernel matrix.

Two major characteristics of tracking algorithms are currently utilized in visual tracker evaluation. The first is accuracy/robustness score that can be measured by Visual Object Tracking (VOT) [17] and Online Object Tracking (OTB) [34] metrics. In addition, these metrics contain an evaluation toolkit and a database of scores for all state-of-the-art trackers. This makes it easier for researchers to estimate and compare their own tracking algorithms. The second important parameter of trackers is the frames per second (FPS) rate, in other words the speed at which the algorithm processes data. An OTB survey on state-of-the-art trackers shows that only a small number (around 20%) of them has reached a real-time barrier. Tracker algorithm speed performance has become more important in multi-object trackers where the workload increases virtually proportionally with the number of objects tracked simultaneously.

Deep Learning (DL), especially Convolutional Neural Networks has attracted much attention in the last few years. At present, CNN has been successfully applied in various fields of Computer Vision, such as image classification [5, 18, 29], semantic segmentation [9, 22], object detection [5, 8, 22], and human pose estimation [30, 31]. In addition, some recent studies have tried to apply the discriminating power of CNN for visual tracking purposes [7, 14, 21, 23, 24, 35]. The model in this kind of trackers is a discriminative CNN with two outputs: target and background probabilities. Almost all CNN trackers outperform state-of-the-art tracking algorithms (like TLD and KCF) in terms of accuracy, but lose in terms

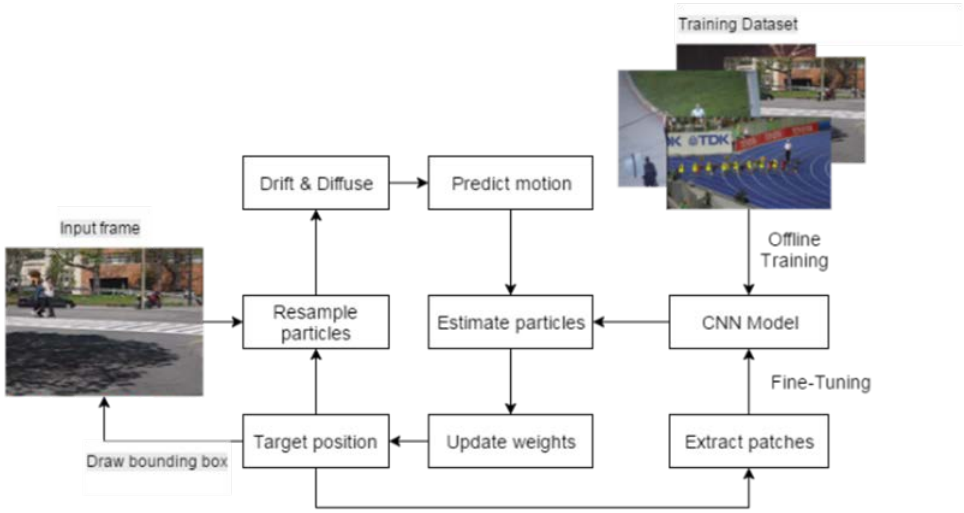

Fig. 1. Tracking algorithm workflow

of speed. One of the most recently successful trackers based on CNN is MDNet [24]. Although MDNet has been evaluated as a top tracker in terms of accuracy/robustness (first place on VOT2015 [17]), it suffers from low speed performance - around 1 FPS. There are two main reasons for this slowness of CNN-based trackers. The first is its online learning procedure: at present, the only optimal CNN training algorithm is Gradient Descent with Momentum [26], which works iteratively until convergence. In CNN trackers, the training procedure is usually limited to a maximum number of iterations that can influence tracking accuracy, but needed in order to achieve stable FPS. The second reason is that the candidates generation method is used in many trackers and all mentioned before CNN trackers is the Monte Carlo sampling [6] method. It requires an algorithm to estimate a huge number of candidates and slow it down significantly.

In our algorithm, we used a large video dataset in order to pretrain our CNN model so that the CNN can learn all common features of different objects. During tracking online learning, the algorithm produces a fine-tuning of the CNN weights, adjusting the neural network model to specific features of the tracked object. In order to reduce the number of candidates in every frame, we inserted a particle filter that models a target trajectory and scale changes. This allows the algorithm to significantly reduce the number of candidates estimated on every frame, improving the overall performance of the tracking algorithm.

2. Tracking algorithm

This section is devoted to the foundations on which our algorithm is based: model network architecture, pretraining, and tracking methods. The main workflow of our algorithm is presented in **Fig. 1**, it includes the following steps:

- Offline training
- Tracking with particle filter
- Online fine-tuning

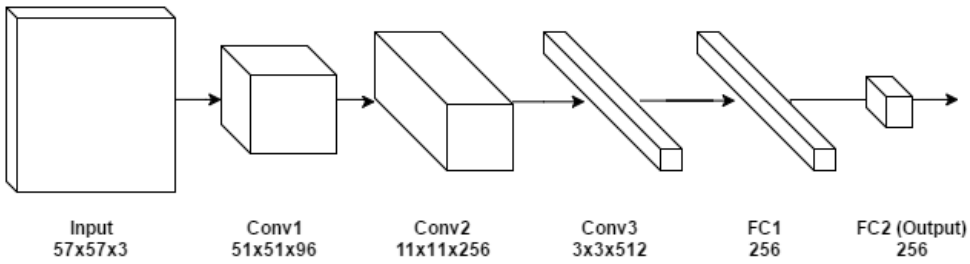

Fig. 2. CNN network architecture

2.1 Network architecture

This section describes in detail our CNN model architecture for generic object tracking. The input network takes three channels (R, G, B) of the target candidate patch resized to 57 × 57 pixels. These patches do not undergo any preprocessing, except mean subtraction - for more stable training process. Any features that are relevant for recognition, neural network is assumed to learn by itself during offline and online training.

Our CNN model comprises five main layers: The first three layers are convolutional layers that are initialized by the values of the network CNN-M [4], already pre-trained on the large image dataset, so that the offline training time can be reduced severely. Layers 4 and 5 are fully connected layers designed to classify high-level features calculated in layers 1 to 3 into two classes: target and background. The overall network architecture is reflected by **Fig. 2**, where filter dimensions are as follows:

$$Conv1(7x7) - Conv2(5x5) - Conv3(3x3) - FC1(256) - FC2(256) \quad (1)$$

The compact size of the neural network used in this study is dictated by limited resources and real-time requirements of tracker algorithms. Experiments showed that the size of the CNN used in this study is sufficient for robust and accurate tracking while providing a noticeable speed burst.

2.2 Offline training

Before tracking starts, the CNN model has to be trained in an offline manner. In order to make the model recognize a general high-level features of generic objects, offline training puts a network into a state close to the local minimum and subsequent fine-tuning will converge much faster, resulting in faster and more stable tracking.

At first all weights in the network are initialized (simply copied), as in the CNN-M network from [4] that was trained on ILSVRC 2014 images dataset. All parameters are also equivalent to CNN-M parameters, only the first convolutional layer's stride parameter was reduced to one in order to compensate the reduction of input patch size. Subsequently, our network is trained on patches extracted from video datasets, where every frame is labeled by a ground-truth bounding box.

For pre-training we used VOT2015 video dataset [17], in order to separate offline training data and test data from OTB [34]. Data generation for CNN training includes extraction of positive and negative samples, where the positive samples are ground-truth patches with random geometric transformations (translation, scaling and affine transformation) from the target, while the negative samples are background examples sampled all over the frame. From every labeled frame of the video, we extract 256 negative patches and 64 positive patches. Positive

patches are extracted randomly using a Gaussian distribution around ground-truth bounding box, with a condition: positive patch and ground-truth bounding box overlap is higher than 90%. In the same manner, we sample negative patches so that their overlap with ground truth is lower than 40%. For pretraining of Gradient Descent, the Momentum training algorithm was used. The algorithm runs iteratively with the following parameters:

Number of epochs: 1000
Learning rate: 0.0005
Weight decay: 0.001
Momentum: 0.9

2.3 Tracking with Particle Filter

Following completion of network pre-training and learning by the CNN model of the most efficient video features for all five layers, the CNN model serves as an initial model for generic object tracking.
In many state-of-the-art trackers, optimal candidate search is carried out via a simple N_{Gauss} candidates generation by Gaussian distribution around a previous frame target bounding box. Usually, this number is set around 1000 or more to ensure that Gaussian search density will cover all directions and translations tightly enough. After all candidates have been evaluated and assigned some similarity scores the candidate with the highest score becomes the new target location:

$$\begin{gathered} x_t^i = N(x_{t-1}; \sigma_x) \\ y_t^i = N(y_{t-1}; \sigma_y) \\ w_t^i = w_{t-1} * N(1; \sigma_w) \\ h_t^i = h_{t-1} * N(1; \sigma_h) \\ (x_t^*; y_t^*; w_t^*; h_t^*) = \underset{i=1..N}{\arg\max} F(x_t^i; y_t^i; w_t^i; h_t^i) \end{gathered} \tag{2}$$

where $F(x_t^i; y_t^i; w_t^i; h_t^i)$ is the candidate evaluation function towards the target from the previous frame. In our algorithm the Gaussian distribution vector of standard deviations is equal to $(\sigma_x; \sigma_y; \sigma_w; \sigma_h) = \left(\frac{w}{2}; \frac{h}{2}; 0.05; 0.05\right)$ and candidate evaluation is executed by our trained CNN model.
The method we used in our tracking algorithm is based on particle filter, or so-called Sequential Monte Carlo [6] method. The main idea underlying particle filter is that the posterior distribution of some stochastic process can be modeled by a large number N_{Part} of discrete samples. In order to update the posterior distribution of the model the weight of every particle is updated according to the measurements made on every frame by CNN model. Then, the filter removes all particles estimated as unreliable and replaces them with another set of particles in order to maintain the number of particles constant. This kind of algorithm has an apparent trade-off between the accuracy of the modeled posterior distribution and algorithm complexity.
In the tracking process, the particle filter carries out the following roles:

- Predicts motion dynamics of target
- Reduces a calculations, as the number of measures in particle filter is much lower than

in Gaussian search $(N_{Part} \ll N_{Gauss})$

- Refines a target bounding box using an ensemble of particles

This leads to more robust and smooth tracking than search based on Gaussian distribution.
For particles model we used a simple state model: $[x; y; \bar{x}; \bar{y}; w; h]$, where $[x; y]$ are bounding box coordinates, $[\bar{x}; \bar{y}]$ are motion vector components and $[w; h]$ are bounding box sizes. When tracking starts, all particles are initialized with random values generated around the initial bounding box (motion vector parameters are generated with zero mean value) using a Normal distribution with the following standard deviations:

$$\begin{gathered} \sigma_x = \frac{w_{init}}{7};\ \sigma_y = \frac{h_{init}}{7}; \\ \sigma_{\bar{x}} = \frac{w_{init}}{2};\ \sigma_{\bar{y}} = \frac{h_{init}}{2}; \\ \sigma_w = \frac{w_{init}}{20};\ \sigma_h = \frac{h_{init}}{20} \end{gathered} \tag{3}$$

After the initialization step, all particles are updated on every frame in accordance with the following steps (see **Fig. 1**): resampling, drift and diffuse, prediction and update.
In the resampling step, all particles form a density distribution that in turn generates new particles that replace the old ones. Usually, only a small number of particles have sufficiently high weights to give a rise the new particles, while the values of other particles are so low that they generate new particles with probability close to zero. This mechanism implements the main property of genetic algorithms in the particle filter.
On the next step, newly generated particles are drifted according to the corresponding motion model in every individual particle with some random Normal noise added to the motion vectors. Moreover, after the drift, all particles are translated in random directions with small values, ensuring that particles generated by the same parent will differ in future. This prevents PF model from having complete convergence of all particles in a small area (or even one point) and produces a more dense and accurate model. Drift and diffuse noise standard deviations were set to be equal $\left[\frac{w}{20}; \frac{h}{20}; \frac{w}{20}; \frac{h}{20}; \frac{w}{20}; \frac{h}{20};\right]$, where w and h are the sizes of the bounding box in the current frame.
Then, the CNN model calculates the similarity score of every particle via forward pass and, finally, at the last update step, the algorithm reassigns weight values according to the CNN model measurements on every particle in the filter. In order to enforce the selectivity ability of the particle filter we added an exponential transformation to the output scores of CNN, with the following normalization:

$$\begin{aligned} \bar{w}_i &= e^{-4.0*(1-z_i)} \\ w_i &= \frac{\bar{w}_i}{\sum_{k=1}^{N} \bar{w}_k} \end{aligned} \tag{4}$$

where z_i the CNN measured score and w_i is the resulting weight of i-th particle. In order to calculate the resulting target bounding box on every frame all particles are weighted together:

$$S = \sum_{i=1}^{N} (w_i * s_i) \tag{5}$$

Fig. 3 demonstrates the tracking process with a particle filter with $N_{part} = 128$. In the first frame, particles are scattered around the center of the target because of the random initialization values that models the uncertainty of the particle filter about the target motion dynamics. Eventually all particles will converge to the center of the object as the filter models the motion of the target more accurately. Frames 1, 6, 21, and 41 in **Fig. 3** show this clearly.

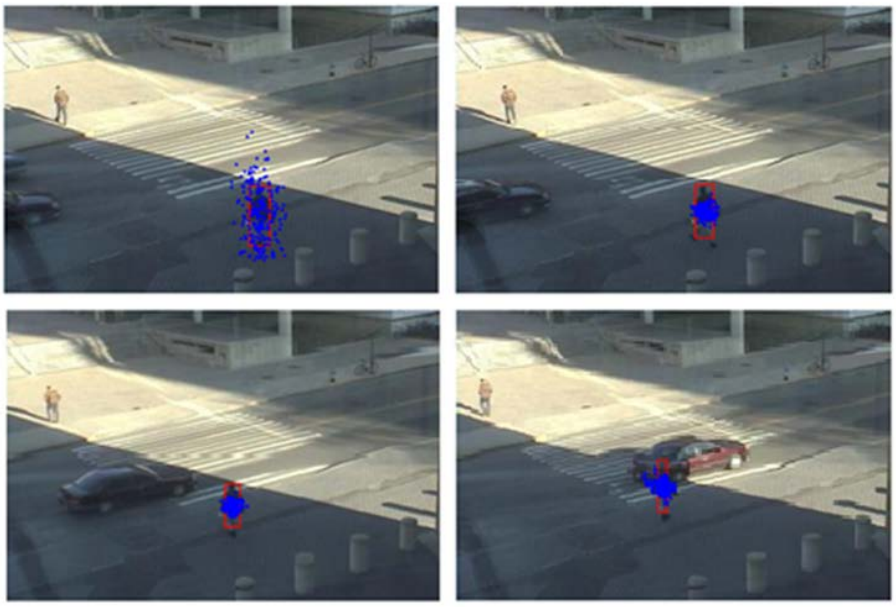

Fig. 3. Examples of particle filter tracking on frames 1, 6, 21, 41

2.4 Online fine-tuning

Finding a target bounding box in every frame allows us to use a new representation and location of the target so that it is possible to refine CNN model's discriminative ability. In addition, fine-tuning in conjunction with the particle filter dynamics model allow the algorithm to track even targets that are occluded for a short time or the targets for which view is mutable during the tracking process.

In essence, the fine-tuning process is very similar to offline training (Section 3.2) in terms of patches mining. The only difference is that the process of learning differs for different layers. At first, during fine-tuning all convolutional layers are assumed to have been learned during the offline training and the backpropagation algorithm does not modify their values during fine-tuning, that significantly reduces the time for online training. Fully connected layers are trained as usual with a learning rate multiplier value of 10. This value was chosen experimentally by examining the trade-off between performance and CNN adaptability speed. The number of positive patches extracted on every frame for online training was set 32 whereas the number of negative patches 128. The only exception being the first frame, on which the model should be approximated to target as much as possible. In our case, in the first frame, the model trained on 128 positive and 512 negative patches.

3. Experimental results

3.1 Experimental setup

The algorithm proposed in this paper was implemented in MATLAB with *MatConvnet* [34] – a convolutional neural network extension. All experiments were conducted using a processor with a desktop GPU and the following specifications:

CPU: *Intel Core i5 – 4670 @ 3.40*
RAM: *8 GB*
GPU: *NVidia Quadro K4000*
GPU Memory: *3GB*
Memory Bandwidth: *134.0GB/s*
CUDA Cores: *768*

3.2 Evaluation methodology

For offline training, we used a dataset from Visual Tracking Challenge 2015 [17]. The dataset contains 60 video sequences, with every sequence accompanied by a file containing ground-truth bounding box information for every frame.
For the accuracy experiments purpose we used the Object Tracking Benchmark 50 dataset [36] with 50 video sequences along with a toolkit containing the results of all state-of-art trackers for comparison. The toolkit provides additional functionality to evaluate the tracker for spatial and temporal robustness. Spatial robustness (SRE) is estimated by expanding the original One-Pass Evaluation (OPE) test with experiments in which the initial bounding box in the first frame is shifted from the center. The temporal robustness (TRE) test is carried out in a similar manner—the tracker starts from a different frame in every experiment. These robustness experiments provide a much more accurate overall tracker score in result.
In addition, OBT provides a list of interference attributes for every video, such as occlusion, scale-change, camera motion, and fast motion. Therefore, the tracker can be tested on subsets of videos filtered by any particular attribute. The tracker evaluation process is based on two well-known metrics: accuracy of the tracker, which is calculated by measuring the bounding box overlap rate, and location error threshold - Euclidian distance based error. Location error represents a precision of target center estimation in respect to the ground truth. The resulting score representing the metrics is defined by the Area Under Curve (AUC). These metrics are not correlated and together they provide a solid estimation of overall tracker performance.

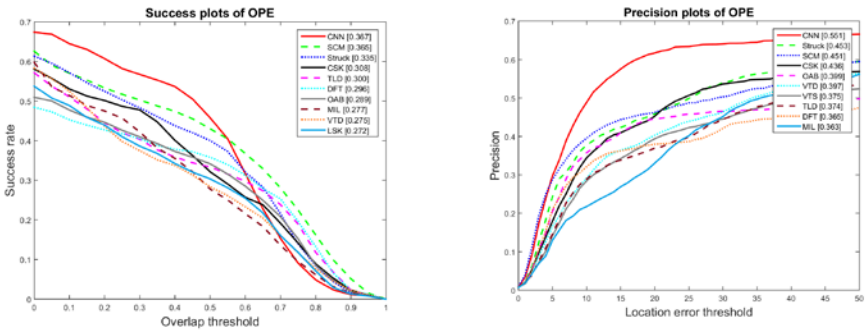

Fig. 4. Overall success and precision plots

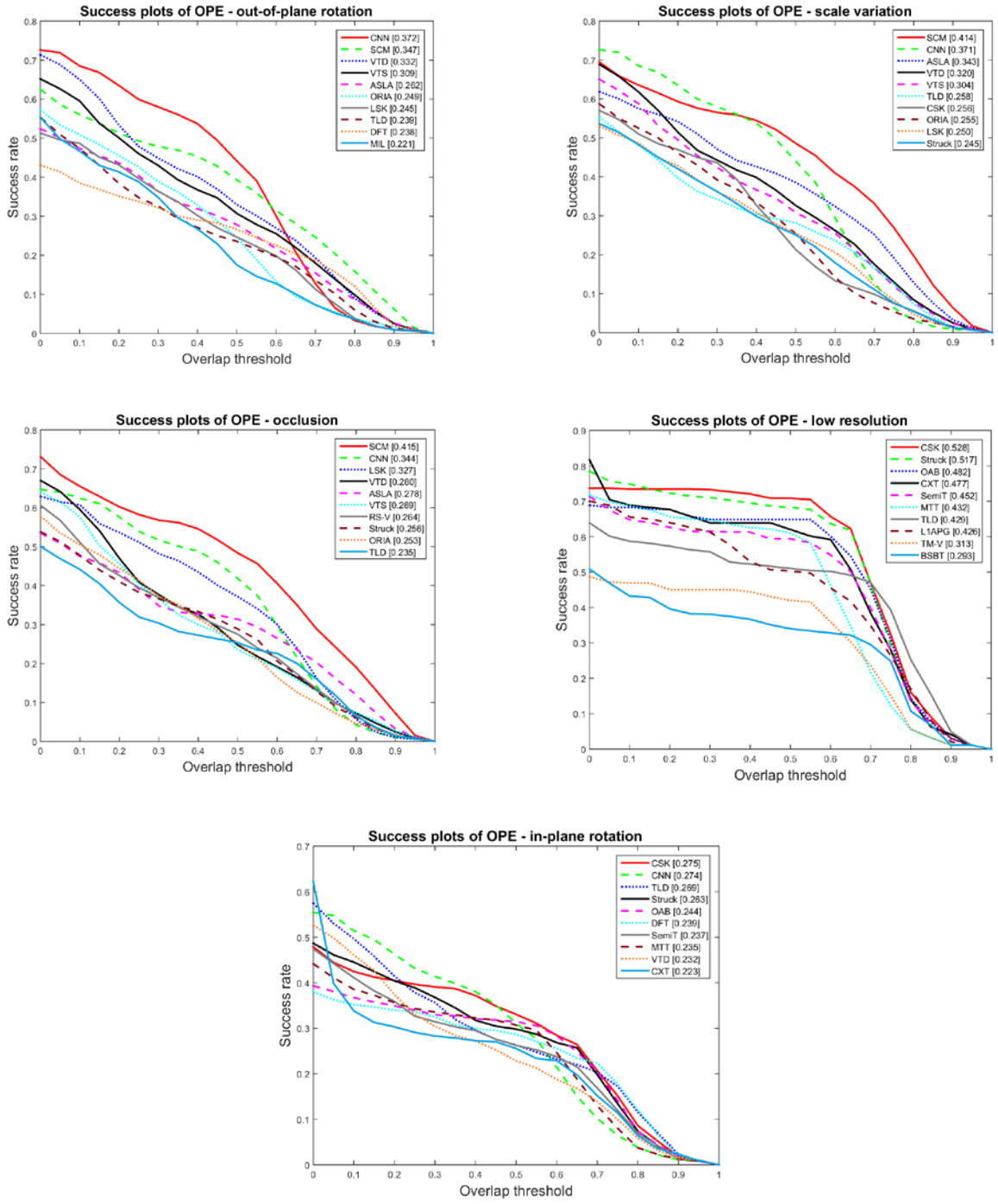

Fig. 5. Overlap success plots for attributes: out-of-plane rotation, scale variation, occlusion, low resolution and in-plane rotation

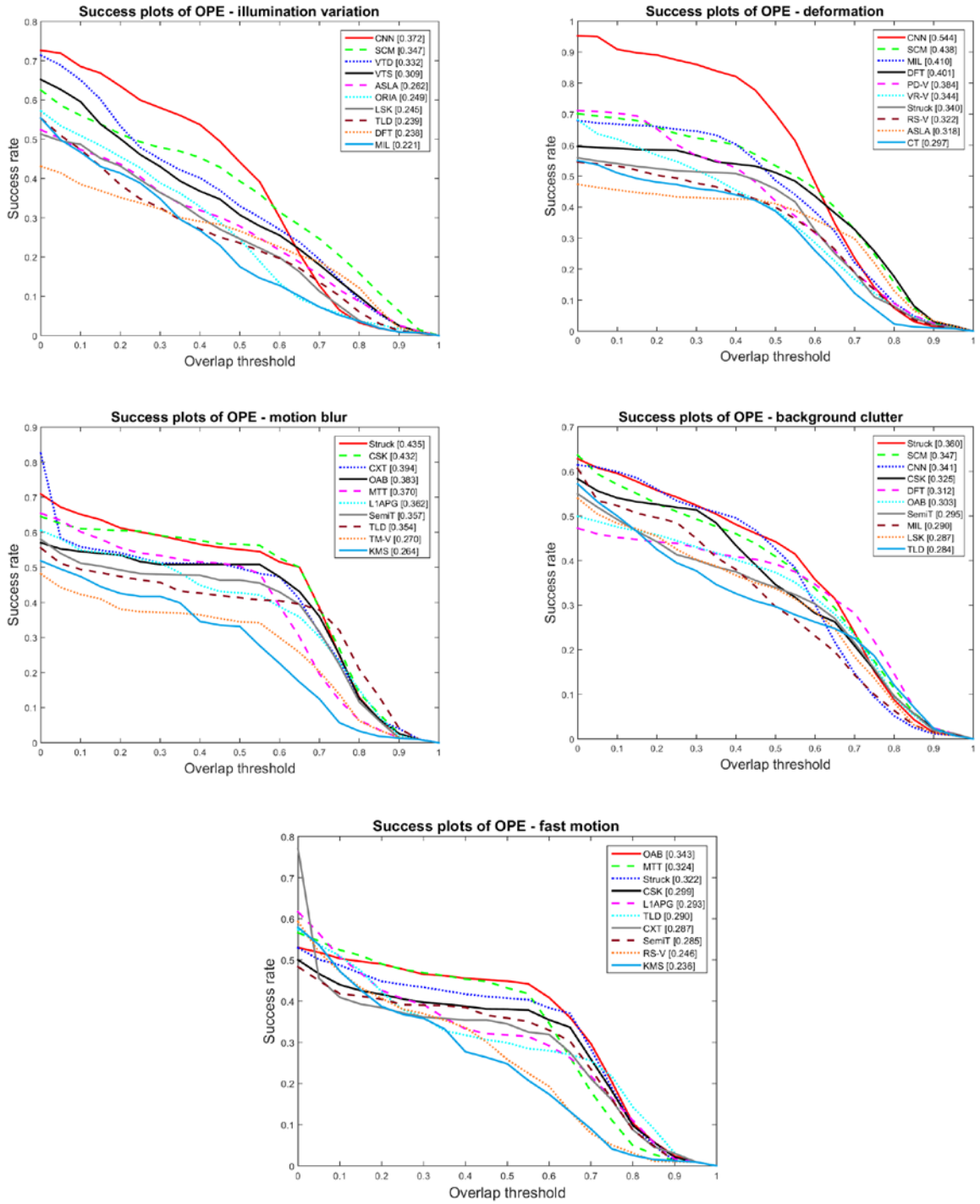

Fig. 6. Overlap success plots for attributes: illumination variation, deformation, motion blur, background clutter and fast motion

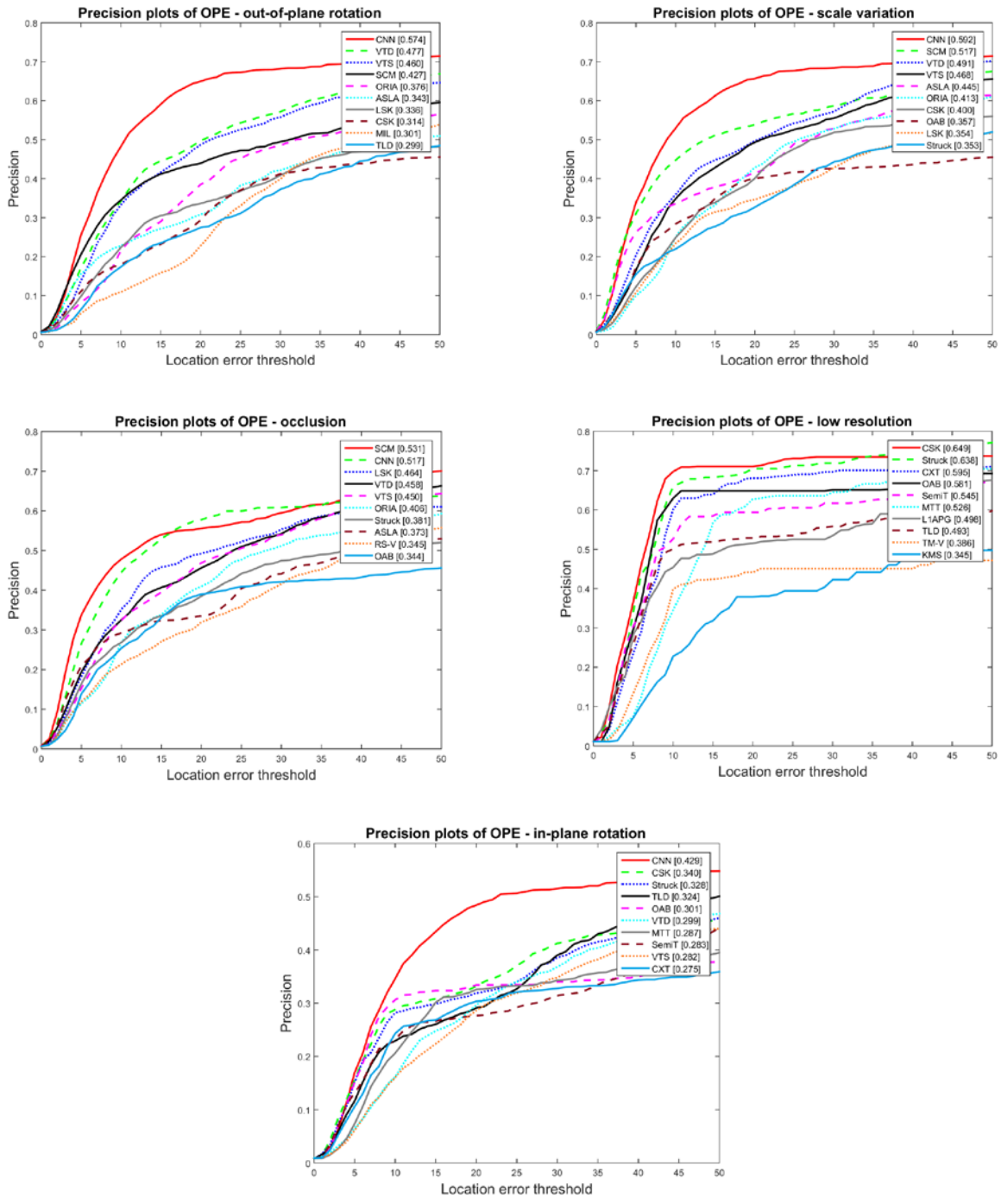

Fig. 7. Precision plots for attributes: out-of-plane rotation, scale variation, occlusion, low resolution and in-plane rotation

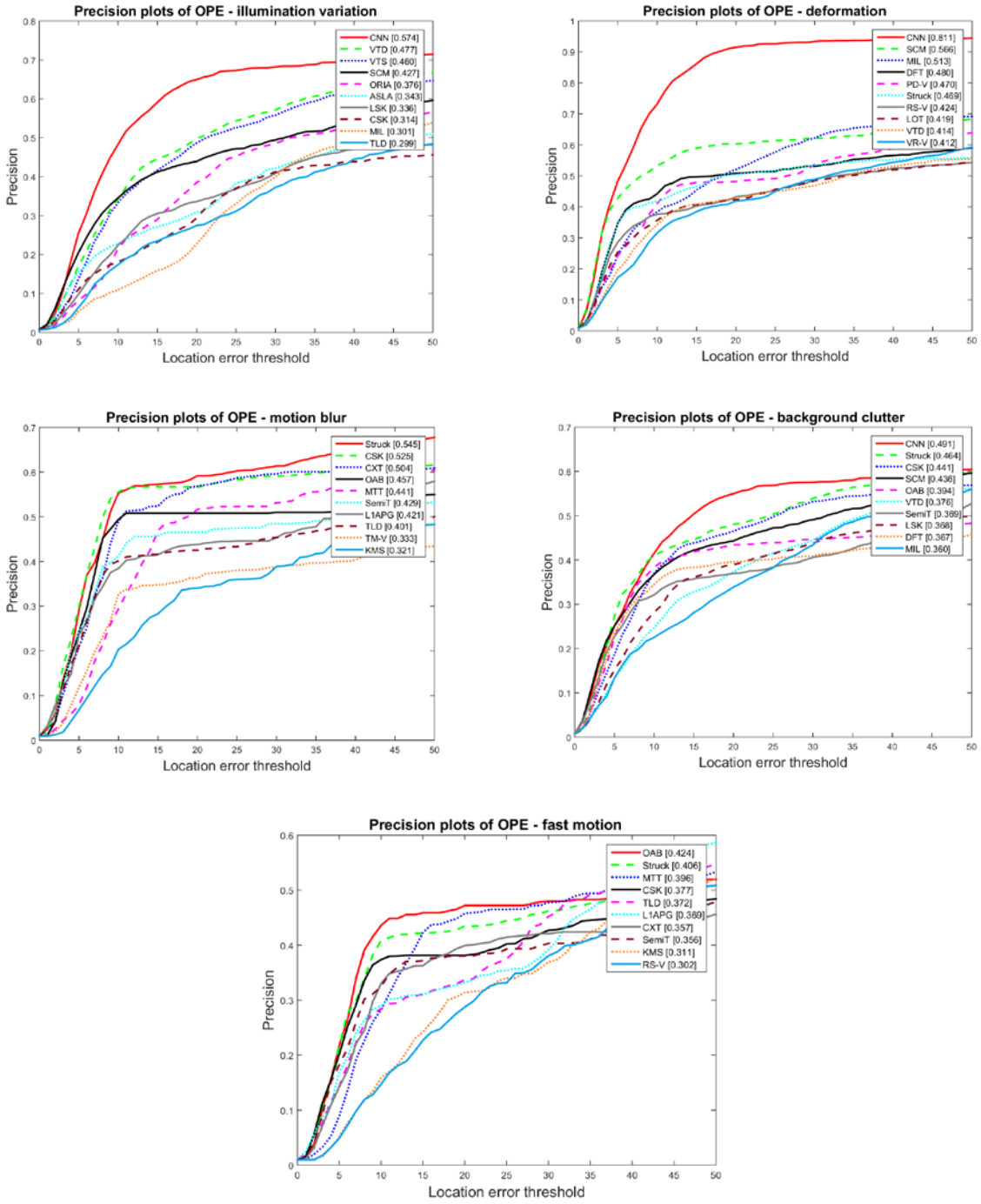

Fig. 8. Precision plots for attributes: illumination variation, deformation, motion blur, background clutter and fast motion

3.3 Results

Overall experiments results of our tracker on OTB [34] dataset are shown in **Fig. 4**. The results are displayed only for the top-10 trackers in every plot. Both, precision and success plots shows that our method outperforms all 29 trackers included in OTB toolkit.

Fig. 9. Tracking results for: CNN (Magenta), MIL (Black), TLD (Blue), SCM (Green), Struck (Red)

In addition to overall performance results, **Fig. 5, 6** shows success rate plots and **Fig. 7, 8** shows precision plots for experiments taken on different subsets filtered by specific video attribute. These resulting plots reveals that in general, the proposed algorithm shows better result than state-of-the-art trackers. The closest results in terms of accuracy and precision are: SCM [17] and Struck [11] trackers, our tracker outperformed them by 0.2%/10.0% and 3.2%/9.8% respectively.

In the attribute subsets experiments, our CNN tracker also obtained among the highest scores in most categories, especially in the subset labeled “deformation” attribute experiment, in which our algorithm substantially outperformed other algorithms with 10.6% accuracy and 24.5% precision.

On the other hand, our CNN tracker lost the target at the beginning of the video sequences marked by “fast motion”, “motion blur” and “low resolution” tags, and did not appear even in the top 10 corresponding plots. This happened because of limitations on the number of

particles in the particle filter and the drastic target appearance change when motion blur occurs.

In addition, it should be noted that the CNN accuracy curve dominates the first half of the plot, then gradually decays, losing to other trackers such as SCM [36] and Struck [11]. This means that the CNN tracker locates the target more often most of the time, but with lower accuracy of bounding box matching. This can be rectified by implementing additional bounding box refinement methods; for example, a method such as that used in R-CNN [9].

Fig. 9 shows the results of tracking for the sequences “crossing”, “skiing”, “singer1”, “singer2”, “skating1”, “shaking”, “subway”, “matrix”, “soccer”, “ironman”, and “deer”. For comparison visualization purposes, tracked bounding boxes for MIL [1], TLD [16], SCM [36], and Struck [11] are displayed as well. The first seven sequences represent successful tracking by our method, whereas last four sequences represent the cases where our tracker was not completely successful.

Table 1. Tracking algorithms speed performance comparison

Method	FPS
Our method	5.3
SCM [36]	0.51
ASLA [15]	8.5
TLD [16]	28.1
CSK [12]	362
DFT [28]	13.2
OAB [10]	22.4
VTD [19]	5.7
VTS [20]	5.7

For the sake of completeness of analysis of the proposed tracking algorithm, **Table 1** provides a speed performance data for the top 10 trackers, including our method. Noticeably, only one tracker from the top 10 - CSK [12], overcame the real-time barrier of 30 FPS. TLD [16] algorithm speed is also very close to real-time, performing on 28.1 FPS. Speed performance of the best tracker from OTB in terms of accuracy/precision is 0.51 FPS. Our CNN tracker is more than ten times faster, performing at 5.3 FPS, which places it on the same performance level as the VTD [19], VTS [20], and ASLA [15] trackers. The slowness of the CNN tracker originates from the CNN’s online fine-tuning sluggishness: for every iteration during online learning, the algorithm first makes a forward propagation, calculates the error function, and then fine-tunes the weights of the CNN via back-propagation.

4. Conclusions

In this paper, we proposed a compact CNN model with particle filter architecture for generic objects visual tracking tasks. Experiments showed that the proposed model is robust to object deformations, point of view, scale, and illumination changes, but suffers from low resolution and fast-motion tracking. Extensive experiments, conducted on the OTB dataset [36], shows that the proposed algorithm is superior to state-of-the-art tracking algorithms in terms of accuracy, while maintaining a reasonable speed performance that is comparable to other state-of-the-art trackers. Speed performance improvements for our CNN tracker will be undertaken in the future work.

Acknowledgements

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00465) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] B. Babenko, M.H. Yang and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8), 1619-1632, 2011. Article (CrossRef Link)

[2] J. S. Bae and T. L. Song, "Image tracking algorithm using template matching and psnf-m," *International Journal of Control Automation and Systems*, 6(3), p. 413, 2008. Article (CrossRef Link)

[3] R. R. Cabrera, T. Tuytelaars and L. Van Gool, "Efficient multi-camera detection, tracking, and identi_cation using a shared set of haar-features," in *Proc. of Computer Vision and Pattern Recognition* (CVPR), 2011 *IEEE Conference on*, pp. 65-71, 2011. Article (CrossRef Link)

[4] K. Chat_eld, K. Simonyan, A. Vedaldi and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. of British Machine Vision Conference*, 2014. Article (CrossRef Link)

[5] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition* (CVPR), 2012 *IEEE Conference on* (2012) pp. 3642-3649. Article (CrossRef Link)

[6] A. Doucet, N. De Freitas and N. Gordon, "An introduction to sequential monte carlo methods," in *Sequential Monte Carlo methods in practice*, pp. 3-14, Springer, 2011. Article (CrossRef Link)

[7] J. Fan, W. Xu, Y.Wu and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Transactions on Neural Networks*, 21(10), 1610-1623, 2010. Article (CrossRef Link)

[8] R. Girshick, "Fast R-CNN," in *Proc. of the IEEE International Conference on Computer Vision*, pp. 1440-1448, 2015. Article (CrossRef Link)

[9] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 580-587, 2014. Article (CrossRef Link)

[10] H. Grabner, M. Grabner and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC*, Vol. 1(5), p. 6, 2006. Article (CrossRef Link)

[11] S. Hare, A. Sa_ari and P. H. Torr, "Struck: Structured output tracking with kernels," in *Proc. of 2011 International Conference on Computer Vision*, pp. 263-270, 2011. Article (CrossRef Link)

[12] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. of European conference on computer vision* (2012), pp. 702-715, August 27, 2016. Article (CrossRef Link)

[13] 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*, 37(3), 583-596, 2015. Article (CrossRef Link)

[14] S. Hong, T. You, S. Kwak and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," *arXiv preprint arXiv:1502.06796*, 2015. Article (CrossRef Link)

[15] X. Jia, H. Lu and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. of Computer vision and pattern recognition* (CVPR), *2012 IEEE Conference on*, pp. 1822-1829, 2012. Article (CrossRef Link)

[16] Z. Kalal, K. Mikolajczyk and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, 34(7), 1409-1422, 2012. Article (CrossRef Link)

[17] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay and R. Pugfelder, "The visual object tracking VOT2015 challenge results," in *Proc. of the IEEE International Conference on Computer Vision Workshops*, pp. 1-23, 2015. Article (CrossRef Link)
[18] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, pp. 84-90, 2017. Article (CrossRef Link)
[19] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. of Computer Vision and Pattern Recognition* (CVPR), *2010 IEEE Conference on*, pp. 1269-1276, 2010. Article (CrossRef Link)
[20] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *Proc. of 2011 International Conference on Computer Vision*, pp. 1195-1202, 2011. Article (CrossRef Link)
[21] H. Li, Y. Li and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Transactions on Image Processing*, 25(4), 1834-1848, 2016. Article (CrossRef Link)
[22] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015. Article (CrossRef Link)
[23] C. Ma, J.-B. Huang, X. Yang and M.H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. of the IEEE International Conference on Computer Vision*, pp. 3074-3082, 2015. Article (CrossRef Link)
[24] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *arXiv preprint arXiv:1510.07945* (2015). Article (CrossRef Link)
[25] S. S. Nejhum, J. Ho and M.-H. Yang, "Visual tracking with histograms and articulating blocks, in Computer Vision and Pattern Recognition," *Proc. of CVPR 2008 IEEE Conference on*, pp. 1-8, 2008. Article (CrossRef Link)
[26] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, 12(1), 145-151, 1999. Article (CrossRef Link)
[27] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems* (2015), pp. 91-99. August 27, 2016. Article (CrossRef Link)
[28] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *Proc. of Computer Vision and Pattern Recognition* (CVPR), 2012 *IEEE Conference on*, pp. 1910-1917, 2012. Article (CrossRef Link)
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. Article (CrossRef Link)
[30] J. J. Tompson, A. Jain, Y. LeCun and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," *Advances in neural information processing systems*, pp. 1799-1807, 2014. Article (CrossRef Link)
[31] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653-1660, 2014. Article (CrossRef Link)
[32] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for MATLAB," in *Proc. of the 23rd ACM international conference on Multimedia*, pp. 689-692, 2015. Article (CrossRef Link)
[33] L. Wang, "Video image object tracking algorithm based on improved principal component analysis," *Journal of Multimedia*, 9(5), 722-728, 2014. Article (CrossRef Link)
[34] Y. Wu, J. Lim and M.H. Yang, "Online object tracking: A benchmark," in *CVPR (IEEE Computer Society)*, pp. 2411-2418, 2013. Article (CrossRef Link)
[35] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang, "Robust Visual Tracking via Convolutional Networks without Training," *IEEE Transactions on Image Processing*, pp. 1–1, 2016. Article (CrossRef Link)

[36] W. Zhong, H. Lu and M.H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. of Computer vision and pattern recognition* (CVPR), 2012 *IEEE Conference on*, pp. 1838-1845, 2012. Article (CrossRef Link)

Vladimir Tyan received the B.S. degree in Information Technologies from TUIT (Tashkent University of Information Technologies), Uzbekistan, in 2010 and M.S. degree in Internet and Multimedia Engineering from Konkuk University, Korea, in 2013. Currently he is working for Ph.D. degree on Internet and Multimedia Engineering in Konkuk University, Korea. His research interests include computer vision, deep learning, object tracking and GPU programming.

Doohyun Kim received the B.S. degree in Computer Engineering from Seoul National University, Korea, in 1985, M.S. and Ph.D. degrees in Computer Science from KAIST(Korea Advanced Institute of Science and Technology), Korea, in 1987 and 2002 respectively. He was a Principal Researcher at ETRI (Electronics and Telecommnications Research Institute) before he joined the Department of Software at Konkuk University in 2004. Currently he is a Professor at Konkuk University. His research interests include embedded intelligence and cloud computing.