Fast Fourier Transform Networks for Object Tracking Based on Correlation Filter

Zhangping He, Zhendong Zhang and Cheolkon Jung, *Member, IEEE*

***Abstract*—In this paper, we propose fast fourier transform networks for object tracking, called *FFTNet*. *FFTNet* is a correlation filter (CF)-based tracker that integrates two main components of CF, i.e. auto correlation and cross correlation between the features of two images. Thus, *FFTNet* takes full advantage of CF: (1) Auto correlation and cross correlation of CF are efficiently computed by FFT so that *FFTNet* achieves high computational efficiency; (2) *FFTNet* successfully performs displacement detection even in similar signals so that it achieves good tracking performance. Moreover, *FFTNet* combines the advantage of CF with convolutional neural networks (CNN) so that it has strong capabilities of learning feature representation and matching function. *FFTNet* is trained end-to-end in an off-line manner. First, we input two input patches of target object and search region into shared convolutional layers to get their features. Then, we calculate auto correlation and cross correlation from two features. Next, we concatenate the results of auto correlation and cross correlation, and put them into another subnetwork to learn a matching function. Finally, we get a response map whose values represent the probability of each pixel belonging to the target. Experimental results demonstrate that *FFTNet* outperforms state-of-the-arts in both tracking accuracy and computational efficiency.**

***Index Terms*—Object tracking, auto correlation, cross correlation, convolutional neural networks, correlation filter, fast fourier transform**

I. Introduction

OBJECT tracking aims at estimating the location and size of the target in an image sequence for a given query, which is one of the most challenging topics in computer vision. It has been applied in many applications such as robotic, video surveillance, human motion analysis, and human-computer interaction (HCI). Although much progress has been made in the past decade, many challenges still exist in designing a robust tracker to handle significant appearance changes, pose variations, severe occlusions, and background clutter. Tracking-by-detection is to build a discriminative classifier that distinguishes the target from the surrounding background. Typically, it captures the target position by detecting the best matching position using a classifier. In recent years, many tracking-by-detection methods have been proposed and demonstrated excellent tracking performance. Online boosting methods [2] were proposed to update the discriminative model in online manner. Multiple instance learning (MIL) [6] and tracking-learning-detection (TLD) [10] were proposed to update tracking models robust to the noise. Struck [5] minimized the structured output objective for localization, which achieved good tracking performance with an elegant formulation. However, their computation costs limit the number of features and training samples.

This work was supported by the National Natural Science Foundation of China (No. 61271298) and the International S&T Cooperation Program of China (No. 2014DFG12780).

Z. He, Z. Zhang and C. Jung (corresponding author) are with the School of Electronic Engineering, Xidian University, Xian, Shaanxi 710071, China e-mail: hezhangping@stu.xidian.edu.cn, zzd775089697@163.com, zhengzk@xidian.edu.cn

Manuscript received November 17, 2017.

A. Correlation Filter Trackers

Correlation filter (CF)-based trackers [7], [19], [13] have attracted much attention due to the computational efficiency and competitive performance. They learn CF in the Fourier domain to have low computational load. Bolme *et al.* [4] proposed the minimum output sum of squared error (MOSSE) filter, while Henriques *et al.* [19] proposed kernelized correlation filters (KCF) with multi-channel features. Danelljan *et al.* [15] added scale regression for accurate scale estimation. Staple [26] incorporated color statistics-based model to achieve complementary traits for CF tracking. Hong *et al.* [23] proposed MUlti-Store Tracker (MUSTer) based on short- and long-term memory to process target appearance memories. Choi *et al.* [27] proposed an integrated tracker with various CFs weighted by a spatially attentional weight map. Danelljan *et al.* [24] developed a regularized CF which extended the training region for CF by applying spatially regularized weights to the suppression of the background. However, there are two main drawbacks in CF tracking: One is to use hand-crafted features which are incapable of capturing semantic information of the target, and the other is the deficiency of training data and thus has inherently a limit in the generality. To overcome the insufficient representation of the hand-crafted features, deep convolutional features were utilized in CF [21], [22] which achieved the state-of-the-art performance. Ma *et al.* [21] estimated the position of the target by fusing the response maps obtained from the hierarchical convolution features of various resolutions in a coarse-to-fine scheme. Danelljan *et al.* [22] extended the regularized CF using deep convolution features. Danelljan *et al.* [28] also proposed a novel CF to find the target position in the continuous spatial domain, while incorporated features of various resolutions. Qi *et al.* [30] tracked the target based on adaptive hedge which was applied to the response maps from deep convolution features. Although CF trackers based on CNN features [21], [22] significantly improve the robustness against geometric and photometric variations, extracting CNN features from each frame and training/updating CF trackers over high dimensional deep features is computationally expensive. Thus, this makes it difficult to achieve real-time tracking performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2790942, IEEE Access

B. CNN-Based Trackers

Driven by the emergence of large-scale visual data sets and fast development of computation power, CNN with their strong capabilities of learning feature representations has shown an extraordinary performance in many computer vision tasks, *e.g.* image classification [8], object detection [14], and semantic segmentation [20]. CNN-based trackers [11], [17], [18], [25], [29], [31] have been proposed to learn the tracking models. Early attempts were suffering from the data deficiency for training networks. To solve the insufficient data, transferring methods were proposed by utilizing pre-trained CNN on a large-scale classification dataset such as ImageNet. Wang *et al.* [17] proposed a framework which fused shallow convolutional layers with deep convolutional ones to simultaneously consider detailed and contextual information of the target. Nam and Han [29] introduced a multi-domain CNN which determined the target location from a large set of candidate patches. Tao *et al.* [25] utilized a siamese network to estimate the similarities between the target and the candidate patches. Wang *et al.* [32] proposed a sequential training method of CNN for object tracking based on an ensemble strategy to prevent overfitting in the network. However, since these trackers tried to adapt the appearance change of the target, the networks need online fine-tuning which is slow and prevents real-time tracking. However, these methods still have a limitation due to the gap between the object classification and tracking. Recently, researchers have tried to overcome the gap between them by training their network with a large amount of tracking video datasets. Held *et al.* [33] proposed the tracking method which captured the target's location with deep regression networks. Bertinetto *et al.* [31] proposed a tracker based on a fully-convolutional Siamese network which was trained end-to-end on the ILSVRC15 dataset for object detection. They demonstrated competitive tracking accuracy over state-of-art methods while running at 58 frame/second (FPS). For reference, most CNN-based trackers run lower than 10 FPS.

C. Motivations

In this paper, we propose *FFTNet* for object tracking based on CF. *FFTNet* integrates two main components of CF, i.e. auto correlation and cross correlation. The reasons of introducing auto correlation and cross coreelation into *FFTNet* are as follows: (1) They are computed by fast fourier transform (FFT) so that *FFTNet* achieves high execution efficiency (nearly 90 FPS); (2) They are useful to detect displacement in similar signals so that *FFTNet* achieves good tracking performance. As shown in Fig. 1, there are two similar signals of x and y with small displacement in the left figure, while auto correlation of x and cross correlation between x and y are shown in the right figure. It can be observed that the displacement between two signals is consistent with the displacement between their auto correlation and cross correlation. Thus, we integrate auto correlation and cross correlation into *FFTNet* which are efficiently computed by FFT. Moreover, *FFTNet* is an end-to-end network, which does not need online training. First, we input the target object and the search region into shared convolutional layers to get feature maps that capture semantic information from inputs. Then, we calculate the cross correlation and auto correlation from two feature maps. Next, we concatenate the results of auto correlation and cross correlation, and put them into another subnetwork that consists convolutional layers and fully connected layers. Finally, we produce a response map whose values represent the probability of each pixel belonging to the target. *FFTNet* elaborately combines the advantage of CF with CNN, and thus both feature representation and matching function in *FFTNet* cooperate with each other to achieve outstanding performance in object tracking.

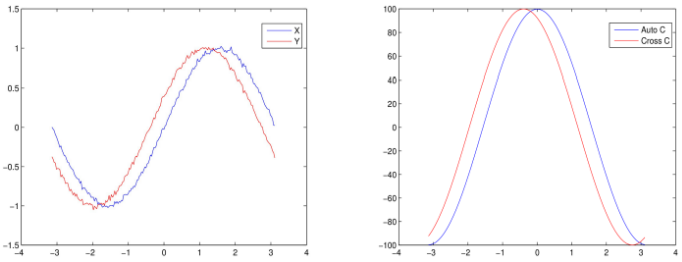

Fig. 1. Displacement between two original signals is in line with the displacement between their auto correlation and cross correlation.

The rest of this paper is as follows. In Section 2, we explain the proposed method in detail. Experimental results and their corresponding analysis are provided in Section 3, while conclusions are made in Section 4.

II. Proposed Method

A. Correlation Filter

A typical correlation tracker [4], [7], [19] learns a discriminative classifier and estimates the translation of target objects by searching for the maximum value of correlation response map. For notational simplicity, we use single-channel signals, which can be generalized for multi-channel data, i.e. images, in a straightforward way. Denote f as the training signal of size $M \times N$ (the current frame). We consider all circular shifts of f as training samples. Each shifted sample $f_{m,n}, (m,n) \in \{0,1,...,M-1\} \times \{0,1,...,N-1\}$ has a Gaussian function label $g(m,n) = e^{-\frac{(m-M/2)^2+(n-N/2)^2}{2\sigma^2}}$, where σ is the kernel scale. Then, a correlation filter h with the same size of f is learned by solving the following minimization problem:

$$\min_h \|h \otimes f - g\|_2 + \lambda \|h\|_2 \tag{1}$$

where $\otimes$ is the circular convolution operator and λ is a regularization parameter. This objective can be solved efficiently in Fourier domain by Fast Fourier Transform (FFT), and its learned CF is written as follows:

$$H^* = \frac{G \odot F^*}{F \odot F^* + \lambda} \tag{2}$$

where the capital letters are the corresponding Fourier transformed signals, i.e. $H = \mathcal{F}(h), F = \mathcal{F}(f), G = \mathcal{F}(g)$, and $*$ represents the conjugate operator. When a new test image z comes (the next frame), the response map y is calculated by:

$$y = \mathcal{F}^{-1}\left(H^* \odot Z\right) \tag{3}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2790942, IEEE Access

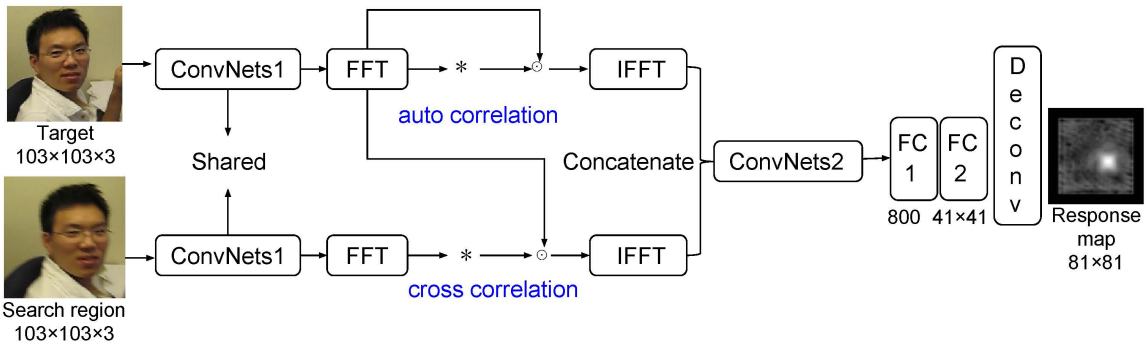

Fig. 2. Network architecture of the proposed *FFTNet*. ConvNets1: Shared convolutional layers. $*$: Conjugate operator. $\odot$: Element-wise product.

By substituting (2) into (3), we get:

$$\begin{aligned} y &= \mathcal{F}^{-1}\left(\frac{G \odot F^*}{F \odot F^* + \lambda} \odot Z\right) \\ &= \mathcal{F}^{-1}\left(\frac{G \odot (F^* \odot Z)}{F \odot F^* + \lambda}\right) \end{aligned} \tag{4}$$

where $F^* \odot Z$ and $F \odot F^*$ are the Fourier representation of the cross correlation between f and z and the auto correlation of f, respectively. Thus, it can be observed that the output is only determined by $F^* \odot Z$ and $F \odot F^*$. As shown in Fig. 1, they are very useful for displacement detection between two similar signals. Therefore, we introduce them into CNN to train an end-to-end network for object tracking.

B. FFTNet

We formulate object tracking as a matching problem by training a generic target matching network off-line for direct tracking without online training. By inputting two patches of target and search regions to this network, we produce a response map whose values represent the probability of each pixel belonging to the target. The network architecture of *FFTNet* is illustrated in Fig. 2. In *FFTNet*, we input the target object and the search region into shared convolutional layers of ConvNets1. The output of convolutional layers is a set of feature maps that capture semantic information from the image. After applying the same convolutional feature transformation to both input patches, we calculate the cross correlation and auto correlation of the two features maps. In the figure, $*$ represents the conjugate operator and $\odot$ is the element-wise product. Then, we concatenate them and add three convolutional layers and two fully connected layers. After that, we reshape the output of FC2 and add a deconvolution layer for upsampling. Finally, we get a response map with size 81×81. The leaky rectified linear unit (LReLU) is used after each convolutional layer as the non-linear activation function. After performing LReLU, there is a batch normalization layer. The detailed parameters of the network are shown in Table I.

C. Back Propagation Through FFT and IFFT

Neural networks are trained by back propagation, which requires that the forward function is differentiable or piece-wise differentiable. In this section, we show that FFT and IFFT are differentiable functions and provide their gradients. We analyze them in 2D case, which can be easily generalized into other dimensions. We formulate FFT in a matrix form, and define an $N \times N$ FFT matrix as follows:

TABLE I
DETAILS OF *FFTNet*

Layer		Input	kernel size	stride	filters
ConvNets1	conv1	$103 \times 103 \times 3$	3×3	1	64
	pool1	$103 \times 103 \times 64$	3×3	2	
	conv2	$51 \times 51 \times 64$	3×3	1	64
	conv3	$51 \times 51 \times 64$	3×3	1	32
ConvNets2	conv4	$51 \times 51 \times 64$	3×3	1	64
	pool5	$51 \times 51 \times 64$	3×3	2	
	conv6	$25 \times 25 \times 64$	3×3	1	64
	conv7	$25 \times 25 \times 64$	1×1	1	10
Deconv		$41 \times 41 \times 1$	5×5	2	1

$$\mathbf{W}_N = \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \dots & \omega_N^{0\cdot (N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \dots & \omega_N^{1\cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \dots & \omega_N^{(N-1)\cdot (N-1)} \end{bmatrix} \tag{5}$$

where $\omega_N = e^{-2\pi i/N}$. Then, based on the definition of FFT and IFFT, we get:

$$\hat{\mathbf{X}} = \mathbf{W}_N \mathbf{X} \mathbf{W}_M \tag{6}$$

$$\mathbf{X} = \mathbf{W}_N^* \hat{\mathbf{X}} \mathbf{W}_M^* \tag{7}$$

where $\mathbf{X}$ is an $N \times M$ image, and $\hat{\mathbf{X}}$ means $\mathcal{F}(\mathbf{X})$. It is obvious that (6) and (7) are differentiable. Their gradients are computed as follows:

$$\frac{\partial \hat{\mathbf{X}}_{i,j}}{\partial \mathbf{X}} = \mathbf{W}_{N[i,:]} \mathbf{W}_{M[:,j]}^T \tag{8}$$

$$\frac{\partial \mathbf{X}_{i,j}}{\partial \hat{\mathbf{X}}} = \mathbf{W}_{N[i,:]}^* \mathbf{W}_{M[:,j]}^{*T} \tag{9}$$

where $\mathbf{W}_{[i,:]}$ and $\mathbf{W}_{[:,j]}$ is ith row and jth column of $\mathbf{W}$ respectively. Since FFT and IFFT are differentiable, the whole network can be trained end-to-end.

D. Training on ALOV++ and VOC2012

We set the amount of context to be half of the mean dimension $p = (w+h)/4$. Then, we resize the cropped image

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2790942, IEEE Access

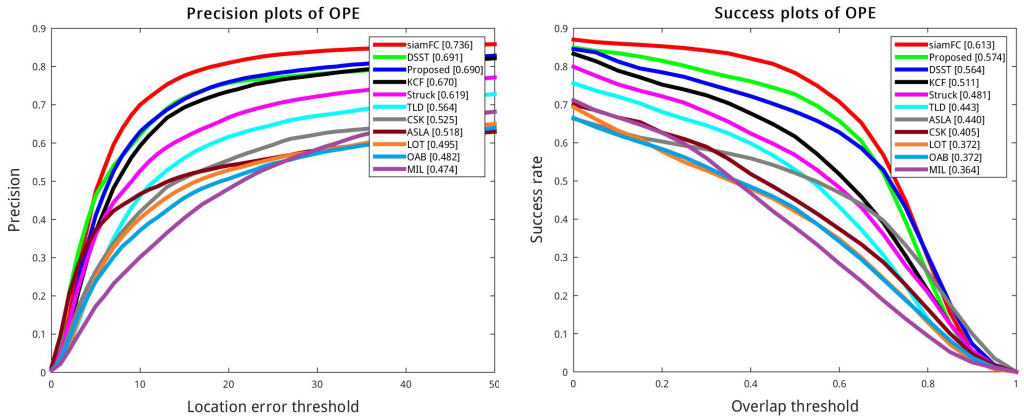

Fig. 3. Precision and success plots of one-pass evaluation (OPE) on OTB50 dataset. The performance score for each tracker is described in the legend. For precision plots the score is the precision value at the threshold 20, while for success plots the score is the area under the curve (AUC) score.

to the network input size (103×103). To find the target in the current frame, we assume that the movement of the target is smooth and thus crop a search region in a similar way to the current frame centered on the previous prediction. We train our network with a combination of videos from ALOV++ data set and still images from VOC2012 object detection dataset. ALOV++ is an object tracking dataset which collects a total of 314 videos. These videos cover diverse circumstances. In this data set, approximately every 5-th frame of each video has been labeled with the location of some objects being tracked. We remove 11 videos that overlap with the testing data set of OTB50[12] leaving 303 videos for training. Two successive annotated frames for each video are used to form a sample pair. To make the proposed network learn a more diverse set of objects and prevent overfitting to the objects in training videos, we employ still images from VOC2012 dataset for training. We also filter out these too big or too small objects. To train our tracker from an image, we first crop the target centered on the ground-truth bounding box, and then take random crops of the image by translation and scale change as search regions. We use these crops because they are taken from different frames of a video. Similar to [33], we use Laplace sampling to represent the smooth movement of the targets. To be specific, we model the center of the bounding box in the current frame $\left(c_x', c_y'\right)$ relative to the center of the bounding box in the previous frame (c_x, c_y) as follows:

$$\begin{aligned} c_x' &= c_x + w \cdot \Delta x \\ c_y' &= c_y + h \cdot \Delta y \end{aligned} \tag{10}$$

where w and h are the width and height of the bounding box of the previous frame. The terms Δx and Δy are random variables that capture the change in the position of the bounding box relative to its size. In our setting, Δx and Δy are modeled with a Laplace distribution with a mean of 0. Similarly, we changes the model size by:

$$\begin{aligned} w' &= w \cdot \gamma_w \\ h' &= h \cdot \gamma_h \end{aligned} \tag{11}$$

where w' and h' are the current width and height of the bounding box, respectively; and w and h are the previous width and height of the bounding box, respectively. The terms γ_w and γ_h are random variables that capture the size change of the bounding box. We model γ_w and γ_h by a Laplace distribution with a mean of 1. The scale parameters for Laplace distribution are $b_x = 1/5$ for the motion of the bounding box center and $b_s = 1/15$ for the change in bounding box size. We constrain the random crop such that it should contain at least half of the target object in each dimension. We also limit the size changes such that $\gamma_w, \gamma_h \in \{0.7, 1.3\}$ to avoid over-stretching or over-shrinking the bounding box in a way that would be difficult for the network to learn.

We adopt an element-wise logistic loss for the loss function as follows:

$$L(y, v) = \frac{1}{81 \times 81} \sum_{i=1}^{81} \sum_{j=1}^{81} \log\left(1 + e^{-y(i,j)v(i,j)}\right) \tag{12}$$

where v is network's prediction and $y \in \{1, -1\}$ is its ground-truth label which represents the pixel in the search image belongs to target or background. The elements of the response map are considered to be a positive example if they are within radius R of the center.

E. Tracking

In testing, we first initialize the target with the ground-truth of the first frame. When a new frame comes, we crop the search region at the previous estimated position, then feed the target image and search region into *FFTNet* and get a score map. Since the score map may contain noise, we process it with Gaussian filtering. To find the most probable location, we calculate the weighted average of all positions whose score is larger than a threshold, i.e. we define this threshold as $0.9\times$ the maximum value. The formula is described as follows:

$$\begin{aligned} i^* &= \frac{\sum_{i=1}^{81}\sum_{j=1}^{81} v(i,j) \cdot i \cdot I\{v(i,j) > 0.9 v_{\max}\}}{\sum_{i=1}^{81}\sum_{j=1}^{81} v(i,j) \cdot I\{v(i,j) > 0.9 v_{\max}\}} \\ j^* &= \frac{\sum_{i=1}^{81}\sum_{j=1}^{81} v(i,j) \cdot j \cdot I\{v(i,j) > 0.9 v_{\max}\}}{\sum_{i=1}^{81}\sum_{j=1}^{81} v(i,j) \cdot I\{v(i,j) > 0.9 v_{\max}\}} \end{aligned} \tag{13}$$

where $I\{\cdot\}$ is an indicator function. After estimating the optimal position, the position relative to the probability map's

center is the displacement. Finally, we need to convert this displacement to the original search region. Similar to [4], we also adopt Peak to Sidelobe Ratio (PSR) as a measure of the confidence of the tracking results. To compute PSR, the response map v is split into the peak which is the maximum value and the side-lobe which is the rest of the pixels excluding an 5×5 window around the peak. The PSR is then defined as $\frac{v_{\max}-u_{sl}}{\sigma_{sl}}$ where $v_{\max}$ is the peak values and u_{sl} and σ_{sl} are the mean and standard deviation of the side-lobe. For scale estimation, we search for the object over three scales $\{0.98, 1.0, 1.02\}$. We select the appropriate scale by PSR and update the scale by linear interpolation with a factor of 0.65. Since only the bounding box from the first frame is accurate, updating the target by the previous prediction may introduce errors, resulting in the tracker drifting. Hence, we do not update the target image.

III. Experimental Results

We perform experiments with a PC with ubuntu 14.04 LTS operating system and a single Titan X GPU. We implement *FFTNet* using Theano and Lasagne, and utilize FFT function from CUDA library for parallel computing. The parameters are optimized by an adaptive stochastic gradient descent (SGD) algorithm, i.e. Adam. The initial learning rate is set to be 0.001. We also use the weight decaying rate as a regularization term, and we set the weight decaying rate to 0.00005. The initial values of the parameters follow a Gaussian distribution, which are scaled according to the improved Xavier method [3]. Total 40 epochs are trained and 12,000 sample pairs are used at each epoch. The size of mini-batches is 128. For tests, we use the Visual Tracker Benchmark (OTB50) [12] which contains 50 videos with 51 fully annotated sequences and covers a variety of challenging scenarios such as illumination variation, scale variation, occlusion, and deformation. We use two performance measures: Success plot and precision plot. Specifically, for a given overlap threshold in [0, 1], a tracker is considered successful in a frame if its overlap rate exceeds the threshold. The success rate for a video measures the percentage of successful frames over the entire video. By varying the threshold gradually from 0 to 1, it gives a plot of the success rate against the overlap threshold for each tracker. A similar performance measure called precision plot is defined for the central pixel error which measures the distance in pixels between the centers of the bounding boxes for the groundtruth and the prediction. The difference is that the precision at threshold 20 is used for the performance score instead of the area under the curve (AUC) score as in the success plots. We compare *FFTNet* with 10 trackers: OAB [1], LOT [16], ALSA [9], CSK [7], TLD [10], MIL [6], Struck [5], KCF [19], DSST [15], siamFC [31]. The overall success and precision plots are shown in Fig. 3. For each tracker, we obtain the curve by averaging over those for all 51 test sequences.

A. Performance Evaluation

In terms of precision, *FFTNet* outperforms the tracking methods reported in [12], which is better than KCF, i.e. an advanced CF-based tracker. In terms of success rate, *FFTNet* outperforms most trackers, which is only a little worse than siamFC [31]. For visual comparison, we provide some tracking results in Fig. 4. From Fig. 4, it can be observed that the proposed tracker performs well in *Jogging-1* and *David3* with short-term occlusion, *Shaking* and *Sylvester* with illumination variation (IV), *Deer* and *Jumping* with fast motion (FM) and motion blur (MB), *carScale*, *Car4*, *David1* and *Dog1* with scale variation (SV). The experimental results verify that *FFTNet* is robust to short-term occlusion, illumination variation, fast motion, and scale variation. All videos in OTB50 are annotated with 11 different attributes.

TABLE II
Average runtime comparison between different methods on the OTB50 (Unit: FPS)

Methods	Language	GPU	AUC Score	Average
siamFC	Matlab	Titan X	0.613	58
DSST	Matlab	–	0.564	24
KCF	Matlab	–	0.511	172
Struck	C++	–	0.481	20
FFTNet	python	Titan X	0.521	88.8
FFTNet+S	python	Titan X	0.574	49

We also provide the experimental results under different attributes in Fig. 5. In most challenging conditions such as occlusion (OCC), motion blur (MB), fast motion (FM), out-of-plane rotation (OPR), background clutters (BC), siamFC [31] achieves the best performance, while DSST [15] is the most robust to illumination variation (IV). *FFTNet* is very effective in handling the deformation of the target objects (DEF). Compared to other discriminative methods, *e.g.* DSST, KCF, and Struck, *FFTNet* is trained end-to-end on large supervised datasets (VOC2012 and ALOV++) in an off-line manner, and generally outperforms them in most challenging conditions. This is because *FFTNet* is based on deep neural networks which have strong ability of feature representation.

B. Runtime Comparison

For runtime comparison, we provide average frame/second (FPS) of different trackers on all OTB50 videos in Table II. We use two versions of the proposed method: *FFTNet* and *FFTNet+S*, i.e. *FFTNet* with scale estimation. *FFTNet* runs at 88.8 FPS on average of all 50 videos, which much exceeds real-time requirements. Also, *FFTNet+S* achieves very high execution efficiency, which runs at average 49 FPS on all videos, even with scale estimation. This is because *FFTNet* does not need on-line fine-tuning and detection proposals. For tests, *FFTNet* only needs two input patches, i.e. target and search region, and finally produces the response map to determine the tracking result.

IV. Conclusion

In this paper, we have proposed *FFTNet* for object tracking based on CF. We have taken full advantage of CF, i.e. high computational efficiency and competitive performance, into *FFTNet*. Moreover, *FFTNet* is based on deep neural

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2790942, IEEE Access

Fig. 4. Tracking results on 10 test sequences: *Jogging-1, David3, Shaking, Sylvester, Jumping, Deer, CarScale, Car4, David1, Dog1*.

networks so that it has strong capabilities of learning feature representation and matching function. *FFTNet* is trained end-to-end in an off-line manner instead of online fine-tuning. Thus, *FFTNet* performs object tracking feed-forwardly from target-search pairs to generate the response map, and achieves high execution efficiency of nearly 89 FPS. Experimental results demonstrate that *FFTNet* is more robust to short-term occlusion, illumination variation, fast motion and scale variation than state-of-the-art real-time trackers.

REFERENCES

[1] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. BMVC*, vol. 1, pp. 6, 2006.

[2] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proceedings of the European Conference on Computer Vision*, pp. 234C247, 2008.

[3] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010.

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, , pp. 2544–2550, 2010.

[5] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *Proceedings of the IEEE Conference on Computer Vision (ICCV)*, pp. 263–270, 2011.

[6] 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.

[7] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proceedings of the European Conference on Computer Vision*, pp. 702–715, 2012.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[9] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1822–1829, 2012.

[10] 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.

[11] N. Wang and D. Y. Yeung, "Learning a deep compact image repre-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2790942, IEEE Access

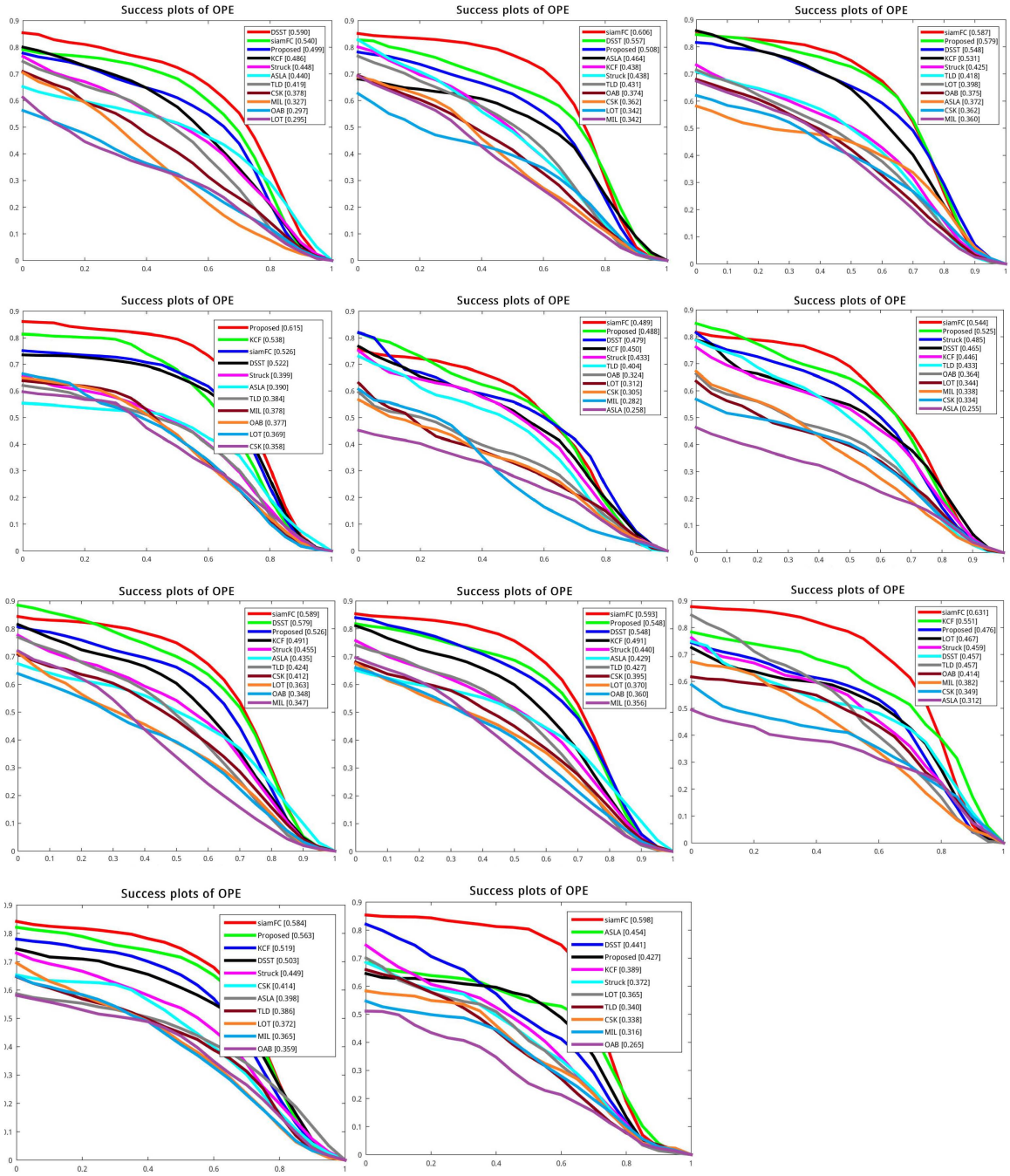

Fig. 5. Attribute-based analysis of different trackers on OTB50 dataset. Attributes from top to down, from left to right are illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane (OPR), out-of-view (OV), background clutter (BC), and low resolution (LR).

sentation for visual tracking," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 809–817, 2013.

[12] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411–2418, 2013.

[13] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual track- ing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090–1097, 2014.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[15] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. British Machine Vision Conference*, 2014

[16] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *International Journal of Computer Vision*, vol. 111, no. 2, pp. 213–228, 2015.

[17] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3119–3127, 2015.

[18] N. Wang, S. Li, A. Gupta, and D.Y. Yeung, "Transferringrich feature hierarchies for robust visual tracking," arXiv preprint arXiv:1501.04587, 2015.

[19] 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.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[21] C. Ma, J. B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3074–3082, 2015.

[22] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 58–66, 2015.

[23] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "Multi-store tracker(muster): Acognitive psychology inspired approach to object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 749–758, 2015.

[24] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4310–4318, 2015.

[25] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1420–1429, 2016.

[26] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1401–1409, 2016.

[27] J. Choi, H. Jin Chang, J. Jeong, Y. Demiris, and J. Young Choi, "Visual tracking using attention-modulated disintegration and integration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4321–4330, 2016

[28] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. European Conference on Computer Vision*, pp. 472–488, 2016.

[29] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2016.

[30] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged deep tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4303–4311, 2016.

[31] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. European Conference on Computer Vision*, pp. 850–865, 2016.

[32] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Stct: Sequentially training convolutional networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1373–1381, 2016.

[33] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Proc. European Conference on Computer Vision*, pp. 749–765, 2016.