LEARNING A REAL-TIME GENERIC TRACKER USING CONVOLUTIONAL NEURAL NETWORKS

Linnan Zhu[⋆]*, Lingxiao Yang*[⋆] *, David Zhang and Lei Zhang*[*]

Department of Computing
The Hong Kong Polytechnic University, Hong Kong, China
{cslzhu, cslyang, csdzhang, cslzhang}@comp.polyu.edu.hk

ABSTRACT

This paper presents a novel frame-pair based method for visual object tracking. Instead of adopting two-stream Convolutional Neural Networks (CNNs) to represent each frame, we stack frame pairs as the input, resulting in a single-stream CNN tracker with much fewer parameters. The proposed tracker can learn generic motion patterns of objects with much less annotated videos than previous methods. Besides, it is found that trackers trained using two successive frames tend to predict the centers of searching windows as the locations of tracked targets. To alleviate this problem, we propose a novel sampling strategy for off-line training. Specifically, we construct a pair by sampling two frames with a random offset. The offset controls the moving smoothness of objects. Experiments on the challenging VOT14 and OTB datasets show that the proposed tracker performs on par with recently developed generic trackers, but with much less memory. In addition, our tracker can run in a speed of over 100 (30) fps with a GPU (CPU), much faster than most deep neural network based trackers.

Index Terms— convolutional neural networks, single-target tracking, real-time tracking, generic object tracker

1. INTRODUCTION

Single-target visual tracking is a core computer vision problem which has been attracting significant attentions in the past decades [1, 2]. The tracking problem can be considered as a target detection problem within a search region, called *Tracking-by-Detection*. To learn a robust target detector, many researchers attempted to model the appearance of targets in an online fashion [3, 4, 5, 6]. Recent paradigms for this problem exploit the expressive representation power of Convolutional Neural Networks (CNNs) [7, 8]. Though, CNNs have been successfully applied for the problem of visual tracking [9, 10, 11, 12, 13, 14, 15, 16, 17, 18], it is still a challenging task due to the uncertain changes of objects online, such as illumination changes, shape deformation, occlusion, and fast motion *etc*.

Most existing methods focus on either using CNNs as a generic feature extractor [11, 12, 13] or directly training CNN trackers [14, 15]. Recently, inspired by the success of multiple inputs of CNN model in unsupervised learning [19] and action classification [20], some pair based CNN architectures [16, 17, 18] have been proposed for visual tracking.

Despite achieving promising performance, existing pair based CNN trackers still have many drawbacks. Most of these methods utilized two separated CNNs for each input. For example, one CNN stream stands for the target objects [16, 18] or the cropped regions in previous frame [17], and the other stream represents the search areas [16, 18] or the cropped regions in current frame [17]. This strategy results in a noticeable increasing of the number of model parameters, which consequently requires more labeled samples at the training stage. In addition, previous works cannot well model the temporal cues of tracked objects, as they rely much on the networks, pre-trained on static images [21]. Moreover, current CNN trackers can run at over 100 fps on a GPU device, but run at a very slow speed on a single CPU processer (around 5 fps in our computer) due to the complexity of their network structure. It is expected to develop a tracker which can run at fast speed in both CPU and GPU devices.

In this paper, we aim to learn a network in an off-line fashion and use it to track objects online. To make the tracker lighter and incorporate the temporal cues, we propose a novel frame-pair based CNN architecture. Specifically, we stack the cropped regions from two successive frames together as the input to a CNN stream. The fusion at early stage allows the tracker to directly learn many temporal features [22]. The output of our tracker is a probability map which indicates the location of target. Our design decreases the model size, and simultaneously increases the speed of online tracking. More importantly, we can train the proposed tracker with much less annotated videos as compared with other works [17, 18].

The rest of the paper is organized as follows. In Section 2, we review the related works on visual tracking using CNNs. In Section 3, we introduce the details of our tracker.

[⋆] Authors contributed equally

[*] Corresponding Author. This work is supported by the Hong Kong RGC GRF grant (PolyU 15224015).

978-1-5090-6067-2/17/$31.00

The implementation details and tracking results are presented in Section 4.2. Section 5 concludes the paper.

2. RELATED WORKS

CNNs have demonstrated their expressive representation power in high-level recognition problems [7, 8, 23, 24]. Li *et al.* [9] proposed a CNN architecture to learn the feature representation built upon multiple image cues. Guan *et al.* [10] tracked objects with an online CNN. The CNNs utilized in [9, 10] were trained from scratch and updated online, while they were likely to suffer from a lack of labeled training data. To address this problem, in [9], Li *et al.* sampled useful training data from the historical tracking results. In [10], Guan *et al.* utilized K-means to learn weights in CNN because K-means does not require any labeled data. Instead of training CNNs from scratch online, our tracker is built upon the pre-trained CNNs and trained in an off-line manner.

Many works have been proposed to exploit pre-trained CNN features for visual tracking [11, 12, 13]. Danelljan *et al.* [11] demonstrated that with the Kernelized Correlation Filters (KCF) [6], the shallow convolutional layers of the pre-trained CNNs are more appropriate for tracking since they preserve more spatial information than deeper layers. Ma *et al.* [12] independently trained KCF trackers on multiple layers of the CNNs. The target location is predicted by combining the decision of all trackers, while the combination weights are handcrafted. Qi *et al.* [13] presented an adaptive weighted method that pools KCF trackers from different CNN layers. Unlike these CNN-KCF structures, several works directly train networks for visual tracking [14, 15]. Wang *et al.* [14] tracked objects with two networks, called GNet and SNet. They attached GNet and SNet on top of the conv5-3 and conv4-3 layers of the VGG [8] model respectively, and demonstrated that these two networks are able to capture complementary information for visual tracking. Nam and Han [15] proposed a multi-domain CNN (MDNet) and achieved state-of-the-art results on several benchmarks. In this paper, however, we focus on the problem of learning a generic tracker in an off-line manner. When applying our tracker online, it can track objects at a significantly faster speed, which is significant faster than all trackers mentioned above.

More recently, several authors proposed to learn a similarity function for visual tracking [16, 18, 17]. Tao *et al.* [16] and Bertinetto *et al.* [18] considered tracking as a template matching task and utilized a siamese architecture to learn the matching function. Held *et al.* [17] treated the similarity learning as a regression problem and directly regressed the location of tracked objects. There are many differences between our method and aforementioned. While methods in [16, 18, 17] utilize a two-stream CNNs for visual tracking, our tracker uses a single stream. When using a similar CNN network, our tracker has less model parameters, which significantly reduces the size of training samples, and improves the speed of online tracking.

3. OUR METHOD

In this section, firstly we describe the general framework of our tracker, and then present the technical details. Finally, we illustrate our off-line training stage.

3.1. Tracker framework

Our work falls into the frame-pair based tracking framework [16, 17, 18]. Specifically, we are interested in searching a single target object in the current frame $\boldsymbol{F}_t$ given the object's location in previous frame $\boldsymbol{F}_{t-1}$, where t is the frame index. Introducing bounding box $\boldsymbol{B} = (x, y, w, h)$ to denote the object location (x, y) and size (w, h), we aim at learning a function Φ to predict $\boldsymbol{B}$:

$$\boldsymbol{B}_t = \Phi(\boldsymbol{F}_t, \boldsymbol{F}_{t-1}, \boldsymbol{B}_{t-1}). \quad (1)$$

Here, the function Φ is learned as a CNN considering the fact that CNNs have recently achieved state-of-the-art results in visual tracking.

Our proposed architecture of CNN-based tracker is shown in Figure 1, where the input is the concatenation of a pair of cropped regions. The output of our tracker is a probability map where the max score indicates the center of tracked object in the current frame. We crop a pair of frames $(\boldsymbol{F}_t, \boldsymbol{F}_{t-1})$ with target center $(x+w/2, y+h/2)_{t-1}$, and size $(kw, kh)_{t-1}$, where $t-1$ stands for the index of previous frame and k defines our search radius. The cropped regions are resized into $m_{in} \times m_{in}$ with scale factor (s_x, s_y), and fed into a CNN model to obtain the l-th convolutional activations. Notably, we adopt the early fusion technique to make our tracker lighter. Another merit of early fusion at pixel level is that the network can directly learn temporal cues and predict the motion patterns [22]. Finally, a deconvolutional layer is adopted to upsample the output to a finer probability map with size $m_{out} \times m_{out}$, leading to a more precise localization [12, 18]. Therefore, each pixel in the final probability map corresponds a $r \times r$ region in original network inputs, where $r = m_{in}/m_{out}$ is the sampling factor. However, instead of fixing the interpolation kernel for upsampling, in our experiment we find that the learned kernel in deconvolution layer can locate target better.

Let us briefly compare our tracker with the most similar work GOTURN [17]. Our tracker takes full advantage of early fusion (Figure 1) and thus largely reduces the number of model parameters (from 113M to 9M). Moreover, the output of our tracker is a discrete probability map compared with continuous regression output in GOTURN [17]. Thanks to above changes, we can train our model with fewer annotated videos, which significantly reduces the human efforts for annotation. Finally, our tracker can achieve comparable results with GOTURN and can run at about 30 fps with a single

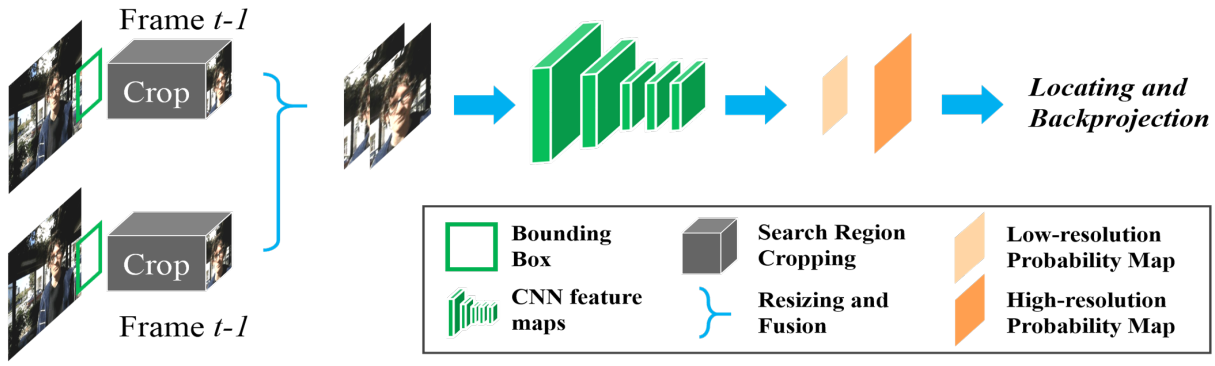

Fig. 1. Illustration of the proposed tracker. The input to our tracker is a pair of cropped regions and the output of our tracker is a probability map, where the max score indicates the coordinates of tracked target in current frame. We back-project the coordinates to un-cropped original frame to obtain real bounding box. (Best viewed in color and magnification).

cpu, which is much faster than GOTURN (around 5 fps in our computer).

In online tracking, usually only the initial bounding box is given, so that we start from $(\boldsymbol{F}_1, \boldsymbol{B}_1)$ and track target objects in successive frames. For each other frame $\{\boldsymbol{F}_t, t = 2, 3, ..., N\}$, we search the location of target from the output probability map, and project its coordinates back to the original un-cropped frame by:

$$\widetilde{\boldsymbol{c}} = (\boldsymbol{c} \cdot \boldsymbol{r})/(\boldsymbol{s}) + \boldsymbol{coord}_{crop}, \tag{2}$$

where $\widetilde{\boldsymbol{c}} = (x, y)$ is the center of target bounding box in un-cropped frame, $\boldsymbol{c} = (x, y)$ is coordinates on probability map, $\boldsymbol{r} = (r, r)$ is the sampling factor from the network inputs to outputs, $\boldsymbol{s} = (s_x, s_y)$ is the resizing factor, and $\boldsymbol{coord}_{crop} = (x, y)$ is the start coordinates of cropped regions in the original frame. All operations in Eq. (2) are *element-wise* operations. The $\boldsymbol{B}_t$ can be calculated from $\widetilde{\boldsymbol{c}}$ with bounding box size. For simplicity, we fix the size of tracked target across all frames in each video.

3.2. Tracker design

Our tracker is built upon existing CNNs, which were pre-trained on ImageNet [21] dataset. In order to adapt the pre-trained CNNs to our problem, we make several changes. First, we simply double copy the channels of filters in the first convolutional layer to accept stacked regions. The stacked regions are transformed into feature maps through 5 convolutional layers. Here, we remove all layers after the last pooling layer to preserve more spatial information, and to reduce the model size. We believe smaller network is more suitable for the problem of visual tracking because: 1) visual tracking is a binary classification task that requires much less model complexity than general recognition problems, and 2) less parameters make our tracker easy to train with less labeled data. Second, we create another 2 multilayer perceptron (MLP) layers on the top of output from the pre-trained CNNs. These two MLP layers help us to learn a more robust tracker. Third, in order to obtain a probability map, a typical operation [25] is to convolute the output of the second MLP layer with a $1 \times 1 \times chns$ filters, where $chns$ is the number of feature maps in the previous layer. In this paper, we adopt an *element-wise cross channel* (EWCC) classifier since it does improve the tracking performance than a typical convolutional layer.The proposed EWCC layer can preserve much spatial information than a typical convolutional operation in [25]. Forth, we utilize a deconvolutional layer to increase the size of probability map to better locate the target object, similar to [25]. We define the EWCC layer as follows:

$$\boldsymbol{P}(i, j) = \sum_{ch=1}^{chns} \boldsymbol{M}(i, pj, ch, n) \odot \boldsymbol{W}(i, j, ch), \tag{3}$$

where $\odot$ is the *element-wise multiplication*, $\boldsymbol{P}$ is the low-resolution probability map, $\boldsymbol{M}$ is the feature maps from the second MLP layer, and $\boldsymbol{W}$ is the classifier weight. $\boldsymbol{P}$, $\boldsymbol{M}$, and $\boldsymbol{W}$ are with the same spatial size. Here, i, j, ch and n are the index for spatial rows, columns, feature channels and samples, respectively. For online tracking, n always equals to 1. The Backpropagation procedures of the new EWCC layer are given below in Eq. (4) and Eq. (5).

$$\frac{\partial \boldsymbol{P}(i, j)}{\partial \boldsymbol{M}(i, j, ch, n)} = \boldsymbol{\Delta}(i, j, ch) \odot \boldsymbol{W}(i, j, ch), \tag{4}$$

$$\frac{\partial \boldsymbol{P}(i, j)}{\partial \boldsymbol{W}(i, j, ch)} = \sum_{n=1}^{N} \boldsymbol{\Delta}(i, j, ch) \odot \boldsymbol{M}(i, j, ch, n) \tag{5}$$

where $\boldsymbol{\Delta}$ is the gradients from successive deconvolutional layer, and N is the training batch size. With above formulations, our tracker can be trained in an end-to-end manner.

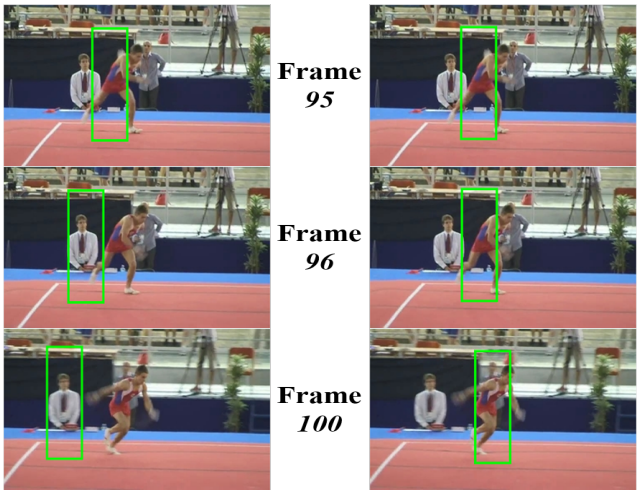

Fig. 2. Illustration of the tracker trained using different offsets. The results shown in left and right columns are obtained by using the tracker trained under the offset $o = 1$ and $o = 6$ respectively (Best viewed in color and magnification).

3.3. Tracker training

We employ logistic loss to train our tracker and define labels on the high-resolution probability map as:

$$y_n = \begin{cases} +1, & \text{if } ||\boldsymbol{u} - \boldsymbol{c}|| \leq \boldsymbol{R}/\boldsymbol{r} \\ -1, & \text{if otherwise} \end{cases} \quad (6)$$

where $\boldsymbol{u}$ is the center of bounding box on the probability map. Eq. (6) indicates that on this map, the samples are considered to be positive if they are within radius of $\boldsymbol{R}/\boldsymbol{r}$. $\boldsymbol{R}$ is set by user before training and fixed across all training epochs. Also, we weight the score map by the positive and negative samples to eliminate class imbalance issue as [18].

A set of videos from ALOV300++ [26] is used for tracker training. In this dataset, the ground truth bounding boxes are annotated in every 5th frames of each video. We augment the data by generating intermediate bounding boxes for the unannotated frames using KCF tracker [6]. The data augmentation can help us learn a more robust tracker. Note that even by this data augmentation, our training set is still half of the ones used in [17]. In the training phase, we observe that directly feeding two successive frames always leads tracker failure, especially in fast motion pattern. To handle this problem, we propose a simple, yet effective sampling strategy to train our tracker. To be specific, in each batch for training, we first sample $N/2$ reference frames $\{t_n, n = 1, 2, ..., N/2\}$, and then get another $N/2$ frames with random numbers ranged between $[t_n - o, t_n + o]$, where o is a hyper-parameter that controls the smoothness of tracking and it is defined by user. When o is set to 1, the training phase degrades into the one just using two successive frames. Figure 2 shows the different results obtained using $o = 1$ and $o = 6$ for training and demonstrates the effectiveness of our sampling method.

4. EXPERIMENTS

In this section, we first briefly introduce the employed datasets and the implementation details. Second, we conduct an in-depth studies on our tracker. Finally, we compare our tracker with other state-of-the-art methods.

4.1. Datasets and Implementations

Training Set. We train our tracker using a collection of annotated videos from ALOV300++ [26] dataset. We remove 7 videos which are overlapped with the VOT14 dataset, remaining 307 videos for tracker training. In this dataset, the ground truth bounding boxes are labeled for approximately every 5 frames of each video. We augment the dataset as described in Section 3.3. After data augmentation, our training set consists of 65,410 images, belonging to 251 different object categories. Note that, even with our augmentation, the total number of training images is still less than 147,903 as used in GOTURN [17]. Moreover, the ImageNet Detection [21] database was utilized in [17] to help the model capturing more diverse object appearance. Compared with [17], our tracker achieves competitive results on the public benchmarks. We split these videos into 250 for training and 57 for validation to fine-tune hyper-parameters. After choosing the hyper-parameters, we retrain our tracker using all 307 videos.

Testing Set. Our goal is to learn a generic tracker like GOTURN [17]. We firstly conduct experiments on the VOT 2014 Challenge [2] database which is a popular benchmark that consists of 25 videos in total. We also compare our work with GOTURN on the OTB [1] dataset, where 9 videos overlapped with the training set are also removed. The results of GOTURN are obtained by their published code [1]. We report two popular metrics [1] for evaluations: the Precision and the Area Under the Curve (AUC) for the success plot. We use a score 20 pixels as threshold for Precision [1].

Implementation Details. We build our tracker based on VGG-F network [27]. The representation power of VGG-F is similar to AlexNet [7], utilized in GOTURN [17]. For the network, we remove all fully-connected layers and add two MLP layers on the top of *Conv5* activations, followed by an EWCC classifier and a deconvolutional layer as described in Section 3.2. For tracker training, to construct a batch for each iteration, we first randomly sample a video, and then select 16 frames using our sampling strategy. The frame offset is 6 for tracker learning. The weights of two MLP and EWCC are generated using a Gaussian distribution, and the weight of deconvolutional (DeConv) layer is initialized with a bilinear kernel. The upsampling factor for DeConv is 4. We execute 50 epochs in total and 1000 iterations per epoch. The optimization is achieved by Stochastic Gradient Descent (SGD) with momentum technique. The learning rate is logarithmically decreased from $log(-2)$ to $log(-4)$. Table 1 presents

[1] https://github.com/davheld/GOTURN

Table 1. Illustration of additional layers used in our tracker. All shapes are formulated as $[h, w, chns, num]$, where h, w, $chns$ and num represent the spatial rows, columns, the channels and the number of filters, respectively.

Layer Name	Input Size	Weight Size	Output Size
MLP1	$[13, 13]$	$[1, 1, 256, 128]$	$[13, 13, 128]$
MLP2	$[13, 13]$	$[1, 1, 128, 128]$	$[13, 13, 128]$
EWCC	$[13, 13]$	$[1, 1, 128, 1]$	$[13, 13]$
DeConv	$[13, 13]$	$[8, 8]$	$[56, 56]$

Table 2. The parameters for the off-line training. All parameters are selected by the method described in Section 4.1

Frame Offset (o)	Batch Size (N)	Search Radius (k)
6	16	2.5
Positive Radius (R)	Epoch	Iteration Per Epoch
16	50	1000
Learning Rate Range	Momentum Rate	Weight Decay
$[log(-2), log(-4)]$	0.9	5e-6

additional layer settings in our tracker, and Table 2 shows all parameters used in the training phase. After learning, we employed our tracker online without any model updating.

4.2. Results

A series of experiments were carried out to investigate the performance of the proposed tracker. We first present the results obtained by training our tracker under different settings, and then show the comparisons between our method and other related works. Examples of visual tracking results can be found in the supplementary file.

In-depth studies. Table 3 shows the results of different model variants. For fairly comparisons, we fix all other settings the same, including the additional layers and the hyper-parameters used in optimization. We can find that our full model achieves the best performance among all our variant models. As presented in Table 3, the model initialized from a pre-trained CNN contributes most to our performance, which is also demonstrated in other high-level recognition problems [19, 20]. Besides, weight updating in Deconv layer also improves the tracker's performance. In addition, Table 3 illustrates that our specially designed classifier leads to a big improvement and demonstrates that the *element-wise cross channel* classifier does preserve more spatial information than traditional convolutional layers, which share all spatial cues in its weights [25]. Moreover, the results obtained by our full model with two different frame offset settings are listed in Table 3 demonstrating the importance of our sampling strategy. According to above analysis, we adopt the following implementations in the rest of comparisons, including model initialized with pre-trained CNNs, DeConv weight updating, EEWC layer, and sampling with frame offset 6.

Table 3. Qualitative results of the comparisons on VOT14 dataset. SR and Prec stand for the success rate and precision, respectively. Results from compared methods are obtained using the authors's published codes. Speeds achieved by CPU and GPU are marked with C and G.

Compared Models	AUC (SR)	Prec@20	Speed (fps)
KCF [6]	0.421	0.547	C392
DSST [28]	0.401	0.553	C43
GOTURN [17]	0.461	0.610	C5 G153
No Pre-trained CNNs	0.304	0.369	
No Deconv Updating	0.387	0.501	
No EEWC	0.403	0.511	
Full Model ($o = 1$)	0.401	0.481	
Full Model ($o = 6$)	0.427	0.563	C38 G148

Comparison with other methods. In the upper part of Table 3, we also show the comparisons to other related works. Our tracker achieves competitive results with other methods. Our tracker is slightly worse than GOTURN [17] in precision and success rate. A main reason may be that our tracker does not model the object's sizes as GOTURN, which is useful for visual tracking. We leave this issue for further studies. On the other hand, our tracker only has about 9 million parameters, which is significantly lighter than the 113 million parameters in GOTURN. The speed of our tracker is approaching the DSST method when running on a CPU device, which is much faster than GOTURN. To the best of our knowledge, our proposed tracker is the fastest deep architecture based tracker when running on a single CPU. In Table 3, we do not include any other deep architecture based trackers like [11, 12, 13, 15] because they were not aiming at learning a generic object tracker. In comparisons to other popular trackers like KCF [6] and DSST [28], our tracker achieves better performance. Note that, even equipped with deep CNNs, our tracker is trained in an off-line fashion and it does not update online as KCF and DSST. A possible improvement of our proposed tracker is to explore an efficient online update method to adapt the weights to tracked objects.

We also conduct another experiment on the OTB50 [1] dataset. The 9 videos overlapped with the training set are removed. In this dataset, our tracker achieves similar behaviors as compared with GOTURN (SR: 0.389 *v.s.* 0.445 and Prec@20: 0.539 *v.s.* 0.646), and it runs in a faster speed.

5. CONCLUSIONS

In this paper, we presented a novel generic tracker that can run at a high speed in both GPU and CPU devices. Our tracker is adopts pair-based inputs for CNN frameworks. The proposed tracker is light-weight, achieved by early fusion at image pixel domain. This strategy allows our network to directly

learn the temporal appearance of tracked objects. The experiments conducted on public tracking dataset demonstrated the effectiveness of our proposed tracker. Another merits of our method is that the training stage needs less annotated videos. We hope our findings could arouse further researches in general object trackers.

6. REFERENCES

[1] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *CVPR*, 2013, pp. 2411–2418.

[2] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomáš Vojíř, and Gustavo et al. Fernández, "The visual object tracking vot2014 challenge results," in *ECCVW*, 2015, pp. 191–217.

[3] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas, "Tracking-learning-detection," *PAMI*, vol. 34, no. 7, pp. 1409–1422, 2012.

[4] Sam Hare, Amir Saffari, and Philip HS Torr, "Struck: Structured output tracking with kernels," in *ICCV*. IEEE, 2011, pp. 263–270.

[5] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang, "Real-time compressive tracking," in *ECCV*. Springer, 2012, pp. 864–877.

[6] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-speed tracking with kernelized correlation filters," *PAMI*, vol. 37, no. 3, pp. 583–596, 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

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

[9] Hanxi Li, Yi Li, and Fatih Porikli, "Robust online visual tracking with a single convolutional neural network," in *ACCV*. Springer, 2014, pp. 194–209.

[10] Hao Guan, Xiangyang Xue, and An Zhiyong, "Online video tracking using collaborative convolutional networks," in *ICME*. IEEE, 2016, pp. 1–6.

[11] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg, "Convolutional features for correlation filter based visual tracking," in *ICCVW*, 2015, pp. 58–66.

[12] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang, "Hierarchical convolutional features for visual tracking," in *ICCV*, 2015, pp. 3074–3082.

[13] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, and Jongwoo Lim Ming-Hsuan Yang, "Hedged deep tracking," in *CVPR*, 2016.

[14] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu, "Visual tracking with fully convolutional networks," in *ICCV*, 2015, pp. 3119–3127.

[15] Hyeonseob Nam and Bohyung Han, "Learning multi-domain convolutional neural networks for visual tracking," in *CVPR*, 2016.

[16] Ran Tao, Efstratios Gavves, and Arnold W M Smeulders, "Siamese instance search for tracking," in *CVPR*, 2016.

[17] David Held, Sebastian Thrun, and Silvio Savarese, "Learning to track at 100 fps with deep regression networks," in *ECCV*, 2016.

[18] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV*. Springer, 2016, pp. 850–865.

[19] Carl Doersch, Abhinav Gupta, and Alexei A Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015, pp. 1422–1430.

[20] Karen Simonyan and Andrew Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014, pp. 568–576.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[22] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014, pp. 1725–1732.

[23] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition.," in *ICML*, 2014, pp. 647–655.

[24] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *CVPRW*, 2014, pp. 806–813.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[26] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah, "Visual tracking: An experimental survey," *PAMI*, vol. 36, no. 7, pp. 1442–1468, 2014.

[27] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *BMVC*, 2014.

[28] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.