Manipulating Template Pixels for Model Adaptation of Siamese Visual Tracking

Zhenbang Li, Bing Li, Jin Gao, Liang Li, Weiming Hu

***Abstract*—In this letter, we show that the challenging model adaptation task in visual object tracking can be handled by simply manipulating pixels of the template image in Siamese networks. For a target that is not included in the offline training set, a slight modification of the template image pixels will improve the prediction result of the offline trained Siamese network. The popular adversarial example generation methods can be used to perform template pixel manipulation for model adaptation. Different from current template update methods, which aim to combine the target features from previous frames, we focus on the initial adaptation using target ground-truth in the first frame. Our model adaptation method is pluggable, in the sense that it does not alter the overall architecture of its base tracker. To our knowledge, this work is the first attempt to directly manipulating template pixels for model adaptation in Siamese-based trackers. Extensive experiments on recent benchmarks demonstrate that our method achieves better performance than some other state-of-the-art trackers. Our code is available at https://github.com/lizhenbang56/MTP.**

***Index Terms*—Model adaptation, siamese networks, visual tracking.**

I. Introduction

OBJECT tracking refers to the task of sequentially locating a specified moving object in a video, given only its initial state. Recently, Siamese networks [1], [2] have demonstrated a significant improvement in object tracking performances. Siamese trackers formulate the visual object tracking problem as learning cross-correlation similarities between a target template and a search region. Tracking is then performed by finding the target object from the search image region by computing the highest visual similarity. Despite its recent success, the learned similarity measure of the Siamese network is not necessarily reliable for objects that are not included in the offline training set, leading to poor generalization [3]. Several recent works aim to adapt the model to the current target appearance. For example, TADT [4] identifies the importance of each convolutional filter according to the back-propagated gradients and selects the target-aware features based on activations for representing the targets. However, the feature extractor of TADT is pre-trained on ImageNet [5], not on large-scale visual tracking datasets. This limits the representation ability of its features on the object tracking task. GradNet [6] exploits the discriminative information in gradients and updates the template in the Siamese network through feed-forward and backward operations. However, the extra sub-network increases the computational cost and is prone to overfitting. UpdateNet [7] learns to combine the target features from previous frames. However, it does not use ground truth information to adaptively adjust the template features of the first frame.

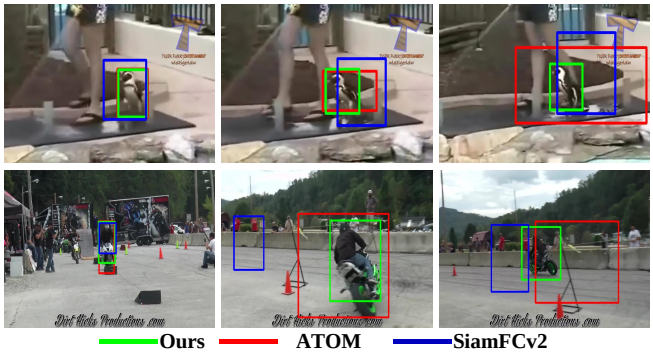

Fig. 1. A comparison of our method with the state-of-the-art trackers ATOM [1] and SiamFCv2 [2] in challenging situations. The example frames are from the GOT-10k [8] testing set.

In this work, we show that the challenging model adaptation task in visual object tracking can be handled by simply manipulating pixels of the template image in Siamese networks. Given an object tracker, our algorithm modifies template pixels in only a few gradient-descent iterations using the target ground-truth in the first frame. For a target that is not included in the offline training set, we believe that a slight modification of the template image pixels can improve the prediction result of the offline trained Siamese network. We use the adversarial example generation method to achieve this, because it is commonly used to slightly modify the input image, and thereby impose an impact on the prediction result of the network. We depart from the purpose of adversarial sample generating in that the latter is aimed to make the prediction of the network worse, while we hope the prediction of the Siamese network is better. The proposed model adaptation method can

This work is supported by the National Key R&D Program of China (No. 2018AAA0102802, No. 2018AAA0102803, No. 2018AAA0102800), the NSFC-General Technology Collaborative Fund for Basic Research (Grant No. U1636218), the Natural Science Foundation of China (Grant No. 61751212, 61721004, 61772225, 61972394), the Key Research Program of Frontier Sciences, CAS (Grant No. QYZDJ-SSW-JSC040), and the National Natural Science Foundation of Guangdong (No. 2018B030311046). *(Corresponding author: Jin Gao.)*

Z. Li, B. Li, J. Gao are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China, and also with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 100190, China (e-mail: zhenbang.li@nlpr.ia.ac.cn; bli@nlpr.ia.ac.cn; jin.gao@nlpr.ia.ac.cn).

L. Li is with the Brain Science Center, Beijing Institute of Basic Medical Sciences, Beijing, China (e-mail: liang.li.brain@aliyun.com).

W. Hu is with CAS Center for Excellence in Brain Science and Intelligence Technology, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China, and also with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 100190, China (e-mail: wmhu@nlpr.ia.ac.cn).

be integrated with varieties of Siamese trackers like SiamFC++ [9]. Note that the parameters of the Siamese network remain intact to preserve the generative ability of offline-trained embedding space. We perform comprehensive experiments on 4 tracking benchmarks: VOT2018[10], TrackingNet[11], GOT-10k[8], and OTB2015[12]. Our approach achieves state-of-the-art results while running at over 80 FPS (see Fig. 1).

II. Proposed Algorithm

In this section, we present a new model adaptation approach for Siamese trackers via directly manipulating template pixels. We start by revisiting the tracking process of popular template matching-based trackers, which is closely related to the proposed approach. Following [13], we formulate object tracking as a confidence-based regression problem, which learns a function $s_\theta : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$, and predicts a scalar confidence score $s_\theta(y, x) \in \mathbb{R}$ given an output-input pair (y, x). The final estimate $f(x) = y^*$ is predicted as follows:

$$f(x) = \arg\max_{y \in \mathcal{Y}} s_\theta(y, x), \tag{1}$$

where x is an input image. y usually represents the center 2D image coordinate of the target object. Currently, there are two prevalent template matching-based paradigms: discriminative correlation filter (DCF) approaches and Siamese tracking methods.

In DCF-based methods, a circular correlation filter w_θ is trained during tracking to predict a target confidence score:

$$s_\theta(y, x) = (w_\theta * \phi(x))(y), \tag{2}$$

where $\phi(x)$ is the features extracted from the search image x.

In contrast to DCF, Siamese trackers exploit a two-stream architecture. One stream extracts the target's features $\phi_\theta(z)$ based on the template image z which is cropped from the first frame according to the ground truth bounding box. The other stream receives as input a large search image x and outputs the search features $\phi_\theta(x)$. The two outputs are cross-correlated to predict a target confidence score:

$$s_\theta(y, x) = (\phi_\theta(z) * \phi_\theta(x))(y). \tag{3}$$

DCF-based trackers and Siamese trackers both have the advantage of utilizing large-scale visual tracking datasets to train the feature extractor $\phi(\cdot)$ or the embedding network $\phi_\theta(\cdot)$ by their own right. This way the representation ability of features on the object tracking task can be enhanced.

In contrast to Siamese trackers, DCF learns the filter w_θ from example patches of the target appearance to discriminate it from the background. Despite the improved tracking efficiency using the circular correlation operation, its boundary effect and sophisticated optimization prevent itself from making a good trade-off between computational speed and tracking performance. Siamese trackers do better in this aspect, although the learned similarity measure in the cross-correlation is not necessarily reliable for objects that are not included in the offline training set, leading to poor generalization.

In this letter, we aim to design a new Siamese tracking method that has the ability to make full use of the specific information of the current video for model adaptation like the DCF-based trackers, although the object in it is not included in the offline training set. This is achieved by utilizing the annotation information in the first frame to perform model adaptation.

Note that Equ. (2) and Equ. (3) bear some similarities with each other, and the main difference lies in the kernel term for correlation: the kernel of DCF is the online-learned w_θ, and the kernel of the Siamese network is $\phi_\theta(z)$. In order for the Siamese network to have the model adaptation ability, we need to adapt $\phi_\theta(z)$ using the first frame ground truth annotation of the current video. There are two design choices to adapt $\phi_\theta(z)$: changing $\phi_\theta(\cdot)$ or changing z. However, changing $\phi_\theta(\cdot)$ may cause too much tedious meta-learning settings to ensure the generative ability of offline-trained embedding space [14], [15]. In contrast, our solution is in a simple way to perform model adaptation of Siamese trackers by changing z, i.e., modifying template pixels in only a few gradient-descent iterations using the target ground-truth at the first frame. Compared with current model adaptation methods of Siamese trackers, the proposed method has the following advantages. First, we never modify parameters of the Siamese network to preserve the representation ability of offline-trained embedding space. Second, different from current template update methods [16], [7], which aim to combine the target features from previous frames, we focus on the initial adaptation using target ground-truth at the first frame. Finally, our model adaptation method is pluggable, in the sense that it does not alter the overall architecture of the base tracker. In the next subsection, we will show how the popular adversarial example generation methods can be used to perform template pixel manipulation for model adaptation.

A. Manipulating Template Pixels for Model Adaptation

At first glance, there may be a contradiction between the model adaptation task and the adversarial example generation task, because these two tasks have different purposes. An adversarial example [17] is a sample of input data which has been modified very slightly in a way that is intended to perform an attack on machine learning systems, which means to cause a machine learning model to make wrong predictions. However, the purpose of model adaptation in our work is to make full use of the annotation information in the first frame to improve the tracking performance for the current video. We will point out in the following that there are some similarities between these two tasks and we can utilize adversarial example generation methods to perform the model adaptation task.

Before introducing the proposed method, we first revisit the popular adversarial example generation methods. One of the simplest methods to generate adversarial images I^{adv} works by linearizing loss function in L_∞ neighbourhood of a clean image and finds exact maximum of linearized function using following closed-form equation [18]:

$$I^{adv} = I + \epsilon \ \text{sign}\big(\nabla_I L(I, y_{true})\big), \tag{4}$$

where I is the input image, and the values of the pixels are integer numbers in the range [0, 255]. y_{true} is the true label for the image I. $L(I, y)$ is the cost function of the neural network for the attack purpose, given image I and label y. ϵ is a hyper-parameter to be chosen. A straightforward way to extend the

TABLE I
STATE-OF-THE-ART COMPARISON ON THE POPULAR TRACKING BENCHMARK OTB2015 WITH THE RUNNING SPEED. FPS: FRAME PER SECOND. OUR SPEED IS TESTED ON AN NVIDIA RTX 2080TI GPU.

Trackers	ECO	MDNet	SiamRPN++	ATOM	SiamFC++_G	Ours
Success	70.0	67.8	69.6	66.9	68.3	69.7
FPS	8	1	35	30	90	82

TABLE II
STATE-OF-THE-ART COMPARISON ON THE TRACKINGNET TEST DATASET IN TERMS OF PRECISION, NORMALIZED PRECISION AND SUCCESS.

Method	Prec.	Norm. Prec.	Succ.
Ours	70.6	81.7	74.9
SiamFC++_G	70.5	80.0	75.4
SiamFC++_A	64.6	75.8	71.2
ATOM	64.8	77.1	70.3
SiamRPN++	69.4	80.0	73.3
MDNet	56.5	70.5	60.6
ECO	49.2	61.8	55.4
SiamFC	51.8	65.2	55.9

above method is applying it multiple times with small step size, and clipping pixel values of intermediate results after each step to ensure that they are in an ϵ-neighbourhood of the original image. This leads to the Basic Iterative Method (BIM) introduced in [17]:

$$\begin{aligned} I_0^{adv} &= I, \\ I_{N+1}^{adv} &= Clip_{I,\epsilon}\{I_N^{adv} + \alpha \ \text{sign}(\nabla_I L(I_N^{adv}, y_{true}))\}, \end{aligned} \tag{5}$$

where $Clip_{I,\epsilon}\{I'\}$ is the function which performs per-pixel clipping of the image I', so that the result will be in L_∞ ϵ-neighbourhood of the source image I. The BIM can be easily made into an attacker for a specific desired target class, called the Iterative Target Class Method [17]:

$$\begin{aligned} I_0^{adv} &= I, \\ I_{N+1}^{adv} &= Clip_{I,\epsilon}\{I_N^{adv} - \alpha \ \text{sign}(\nabla_I L(I_N^{adv}, y_{target}))\}. \end{aligned} \tag{6}$$

Equ. (6) shows that manipulating pixels of the input image in only a few gradient-descent iterations is able to change the network's prediction to the target class y_{target}. Note that our purpose is manipulating pixels of the template image in the first frame, so that the prediction is closer to the ground truth bounding box. So we can perform model adaptation for Siamese networks using Equ. (6) with some modifications:

$$\begin{aligned} z_0 &= z, \\ z_{N+1} &= Clip_{z,\epsilon}\{z_N - \alpha \ \text{sign}(\nabla_z L(z_N, y_{bb}))\}, \end{aligned} \tag{7}$$

where z is the template image in the first frame, and y_{bb} is the label for Siamese trackers generated from the ground truth bounding box. In the next subsection, we will introduce the overall tracking process equipped with the proposed model adaptation module.

B. Tracking Framework

The proposed model adaptation method is integrated with the SiamFC++ tracker [9] in a plug-and-play manner. SiamFC++ is based on SiamFC [2] and progressively refined according to several guidelines proposed in [9]. The input of the original SiamFC++ network is composed of the template image z_0 cropped from the first frame and search image x_i cropped from the i-th frame. However, we expect to perform

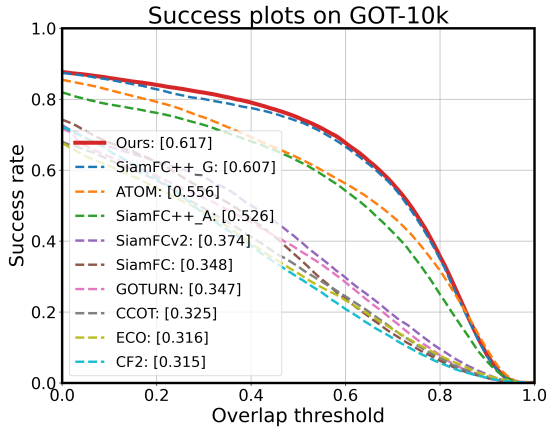

Fig. 2. Comparing the results of our approach against other approaches over the GOT-10k test set. The trackers are ranked by their average overlap scores.

template manipulating such that, after N steps of pixel update on the input pair (z_0, x_0) to obtain z', the tracker performs well on (z', x_i). To achieve this, we start by cropping the initial template image $z_0 \in \mathbb{R}^{3\times128\times128}$ and the initial search image $x_0 \in \mathbb{R}^{3\times289\times289}$ from the first frame using the ground truth bounding box. Then, z_0 and x_0 are sent to the SiamFC++ network to obtain the tracking prediction of the first frame. The tracking loss in SiamFC++ [9] is calculated as follows:

$$L = L_{\text{cls}} + L_{\text{quality}} + L_{\text{reg}}, \tag{8}$$

where L_{cls} is the focal loss [19]. L_{quality} is the binary cross entropy (BCE) loss for quality assessment. L_{reg} is the IoU loss [20] for bounding box regression. The gradient with respect to the template z_0 is used to generate z_1 according to Equ. (7). The updated template z' is obtained by applying a very small number of iterations of Equ. (7). Note that the template image is only updated with the first frame of given sequences and is kept fixed during the whole tracking process to ensure stability. The subsequent tracking procedure remains the same as SiamFC++.

III. EXPERIMENTS

In this section, we first present the implementation details. Then we compare our method with the state-of-the-art trackers on four tracking datasets: OTB2015 [12], VOT2018 [10], GOT-10k [8] and TrackingNet [11]. Specifically, OTB2015 [12] includes 100 sequences, which are labeled with different attributes for in-depth analysis of tracking performance. VOT2018 [10] uniquely applies a reset-based methodology and specially selects 60 sequences of various tracking scenarios for evaluation. GOT-10k [8] and TrackingNet [11] are two recent large-scale high-diversity datasets. They both cover diverse object classes and scenes in the train and test splits. For GOT-10k, there is no overlap in object classes between the train and test splits, promoting the importance of generalization to unseen object classes.

A. Implementation Details

We use SiamFC++ [9] as our base tracker, and the backbone Siamese network adopts GoogLeNet [21]. We do not perform any changes except for the model adaptation component.

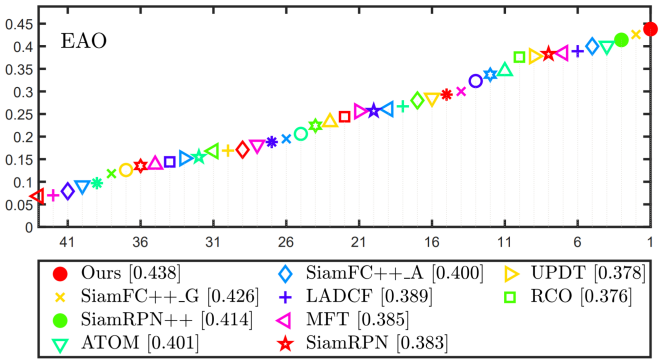

Fig. 3. Expected average overlap graph with trackers ranked from right to left. The right-most tracker is the top-performing according to the VOT2018 expected average overlap values.

The parameter α in Equ. (7) is set to 0.05. Our method is implemented in Python with PyTorch. The proposed tracker runs at over 80 FPS on an NVIDIA RTX 2080Ti GPU.

B. State-of-the-art Comparison

OTB2015 We use success rate to evaluate the performance of trackers on the OTB2015 dataset. Success rate relies on the intersection over union (IOU) of the predicted bounding box and the ground truth bounding box. We compare our method with various tracking algorithms, including ECO [22], MDNet [23], SiamRPN++ [24], ATOM [1] and SiamFC++_GoogLeNet [9]. The iteration number for template update is set to 16. The results are shown in Table I. Our tracker outperforms the online tracker ATOM by 2.8% in terms of the success score, which demonstrates the powerful model adaptation capability of our method.

VOT2018 We compare our method with RCO [10], UPDT [25], SiamRPN [26], MFT [10], LADCF [10], ATOM [1], SiamRPN++ [24], SiamFC++_AlexNet [9] and SiamFC++_GoogLeNet [9] on VOT2018. The trackers are compared using the robustness and accuracy measures. Robustness indicates the number of tracking failures, while accuracy denotes the average overlap between tracker prediction and the ground-truth box. Both of the measures are combined into a single expected average overlap (EAO) score. The iteration number for template update is set to 2. As shown in Fig. 3, the performances of all the listed trackers are not as good as our algorithm.

GOT-10k We use the average overlap (AO) score as performance measure following [8]. We compare our method with CF2 [27], ECO [22], CCOT [28], GOTURN [29], SiamFC [2], SiamFCv2 [30], ATOM [1], SiamFC++_AlexNet [9] and SiamFC++_GoogLeNet [9] on this dataset. The iteration number for template update is set to 2. In Fig. 2, we can find that the proposed algorithm achieves better tracking performance compared with the listed state-of-the-art trackers.

TrackingNet We compare our method with SiamFC [2], ECO [22], MDNet [23], SiamRPN++ [24], ATOM [1], SiamFC++_AlexNet [9] and SiamFC++_GoogLeNet [9]. The iteration number for template update is set to 32. Table II shows that our tracker performs best in terms of precision and the normalized precision while maintaining a very competitive success value.

TABLE III
PERFORMANCE ON OTB2015 USING NOISY INITIAL FRAMES.

Coefficients of Noise			Success Score	
γ_1	γ_2	γ_3	SiamFC++_G	Ours
0.57	1.6	0.93	67.3	69.5
0.88	0.22	0.86	65.7	68.0
0.88	0.43	0.77	64.1	64.4
0.92	0.98	0.82	65.6	66.6
0.98	0.83	0.84	66.5	67.9
1.0	0.81	0.79	65.3	66.0
1.1	1.9	0.90	67.6	68.8
1.2	0.70	0.79	65.7	66.6
1.5	0.19	0.79	64.8	66.3

TABLE IV
11 ATTRIBUTES COMPARISON ON OTB2015 IN TERM OF SUCCESS SCORE.

	BC	DEF	FM	IV	IPR	LR	MB	OCC	OPR	OV	SV
SiamFC++_G	62.5	63.9	68.0	69.2	69.5	**72.5**	66.7	60.9	66.9	55.8	68.4
Ours	**66.5**	**66.8**	**68.7**	**71.3**	**69.6**	69.8	**67.3**	**62.5**	**68.4**	**57.1**	**69.8**

C. Ablation Study

Robustness to Noisy Initial Frames To study how robust the approach is if the first frame is noisy compared to the next frames in the sequence, we add three kinds of noise to the first frame by: changing the image brightness, applying Gaussian blur, and using non-accurate ground truth bounding box annotation. We denote $\gamma_1 \in [0.5, 1.5]$ as the coefficient of brightness variation, $\gamma_2 \in [0, 2]$ the blur radius, and $\gamma_3 \in [0.75, 1]$ the IoU between the non-accurate first frame bounding box annotation and the real bounding box annotation, respectively. We run both the baseline tracker and our tracker 10 times on OTB2015 with randomly sampled γ_1, γ_2 and γ_3 (see Table III). Compared with SiamFC++_GoogLeNet [9], our tracker always performs better under different noise levels, which shows the robustness of our tracker to noisy initial frames.

Attribute-based Analysis For further analyses on the tracking performance, we also demonstrate the advantages of our algorithm through the attribute-based comparison on sequences of the OTB-2015 dataset (see Table IV). In OTB2015, each sequence is annotated with 11 different attributes, namely: background clutters (BC), deformation (DEF), fast motion (FM), illumination variation (IV), in-plane rotation (IPR), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). Compared with SiamFC++_GoogLeNet [9], our tracker achieves better performance on 10 out of 11 attributes, which demonstrates its robustness in challenging tracking scenarios such as illumination variation and motion blur.

IV. CONCLUSION

In this letter, we propose a novel model adaptation method for Siamese trackers, achieving accurate tracking with high speed. We show that the challenging model adaptation task in visual object tracking can be handled by simply manipulating pixels of the template image using the target ground-truth in the first frame. Our model adaptation method is pluggable, in the sense that it does not alter the overall architecture of the base tracker. Numerous experimental results on four object tracking benchmarks demonstrate the effectiveness of the proposed model adaptation method.

REFERENCES

[1] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proc. Comput. Vis. Pattern Recognit.*, 2019, pp. 4660–4669.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 850–865.

[3] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6182–6191.

[4] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, "Target-aware deep tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2019, pp. 1369–1378.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[6] P. Li, B. Chen, W. Ouyang, D. Wang, X. Yang, and H. Lu, "Gradnet: Gradient-guided network for visual object tracking," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6162–6171.

[7] L. Zhang, A. Gonzalez-Garcia, J. v. d. Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for siamese trackers," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6162–6171.

[8] L. Huang, X. Zhao, and K. Huang, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," *arXiv preprint arXiv:1810.11981*, 2018.

[9] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines," in *Proc. Assoc. Adv. Artif. Intell.*, 2020, pp. 12 549–12 556.

[10] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey *et al.*, "The sixth visual object tracking vot2018 challenge results," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2018, pp. 3–53.

[11] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 300–317.

[12] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. Comput. Vis. Pattern Recognit.*, 2013, pp. 2411–2418.

[13] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic regression for visual tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2020, pp. 7183–7192.

[14] T. Yang, P. Xu, R. Hu, H. Chai, and A. B. Chan, "ROAM: recurrently optimizing tracking model," in *Proc. Comput. Vis. Pattern Recognit.*, 2020, pp. 6718–6727.

[15] I. Jung, K. You, H. Noh, M. Cho, and B. Han, "Real-time object tracking via meta-learning: Efficient model adaptation and one-shot channel pruning," in *Proc. Assoc. Adv. Artif. Intell.*, 2020, pp. 11 205–11 212.

[16] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 101–117.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Repres. Workshop*, 2017.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Repres.*, 2015.

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[20] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, "Unitbox: An advanced object detection network," in *Proc. ACM Multimedia*, 2016, pp. 516–520.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[22] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: efficient convolution operators for tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2017, pp. 6638–6646.

[23] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2016, pp. 4293–4302.

[24] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proc. Comput. Vis. Pattern Recognit.*, 2019, pp. 4282–4291.

[25] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg, "Unveiling the power of deep tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 483–498.

[26] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. Comput. Vis. Pattern Recognit.*, 2018, pp. 8971–8980.

[27] C. Ma, J. Huang, X. Yang, and M. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 3074–3082.

[28] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 472–488.

[29] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 749–765.

[30] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2017, pp. 2805–2813.