GENERATING RELIABLE ONLINE ADAPTIVE TEMPLATES FOR VISUAL TRACKING

Jie Guo, Tingfa Xu, Shenwang Jiang, Ziyi Shen

School of Optics and Photonics, Image Engineering & Video Technology Lab, Beijing Institute of Technology, Beijing 100081, China

ABSTRACT

Online adaption of visual tracking is a significant strategy to achieve good tracking performance since the appearance of the object target varies all along with the sequence. However, directly using the tracking results of previous frames to update the model will cause drifting, resulting in tracking failure. We propose a task-guided generative adversarial network (GAN), named TGGAN, to learn the general appearance distribution that a target may undergo through a sequence. Then the online adaption is simply to select templates from the images that are generated from the ground truth template in the first frame and a set of random vectors by the generator. This strategy helps the model alleviate drifting while still obtaining adaptivity. Tracking is treated as a template matching problem under a proposed Siamese matching network structure. Experiments show the effectiveness of the proposed online adaption strategy and the Siamese matching network.

Index Terms— Visual tracking, online adaption, Siamese network, template matching, deep learning

1. INTRODUCTION

Online adaption plays an important role in visual tracking since only the ground truth in the first frame is given. The tracker is supposed to locate the target position in the subsequent frame of the sequence accurately, while the target may undergoes occlusion, illumination change, deformation, background clutter, scale change, and so on.

To deal with various appearance changes, online adaption is adopted to visual tracking. Grabner *et al.* [1] use online boosting to select features and update the weights of weak classifiers using the previous tracking results. Babenko *et al*. [2] adopt Multiple Instance Learning (MIL) to update the model while alleviating the drifting problem. Zhou *et al.* [3] incorporates a deep neural network with an on-line AdaBoost framework to dynamically update the classifiers. Guo *et al*. [4] update the weights of each base matcher and the network parameters after each tracking result. The sparse representation based trackers [5,6] use the current tracking patch to replace the older template patch. The correlation filter based trackers [7,8] use linear interpolation to update their filters after each frame. However, all these online adapting strategies suffer from the problem of self-learning, which means when updated by wrongly labelled samples, the trackers may degrade permanently and drift away.

Siamese convolutional neural networks [9-12] are applied to visual tracking in recent years. Siamese networks have advantageous abilities to learn general matching functions. Methods in [9-11] show that even with a fixed template (the ground truth patch in the first frame), the trackers can achieve state-of-the-art tracking performance.

To avoid drift problem and still benefit from online adaption, we use conditional GAN architecture [13] to learn the general appearance distribution that a target may undergo through a sequence. Then in the tracking phase, we use the learnt generator of the GAN to generate the target templates. The generator takes the ground truth patch in the first frame and random vectors as inputs, and outputs the generated target templates. The generated templates are pulled by the ground truth patch, so they will not drift away. Meanwhile, the generated templates can simulate the possible appearance changes that a target may undergo through a sequence. When tracking, we select the generated templates that best suit the current target appearance. By these strategies, we can update the target templates online while avoiding drifting.

In this work, we first propose a Siamese matching network as illustrated in Fig. 1 to match the candidates with target templates. The Siamese matching network takes advantages of both low-level local information and high-level semantic information. Then we propose a task-guided GAN to learn a generator modeling the general appearance distribution that a target may undergo through a sequence. We use the learnt generator of TGGAN to generate templates and select those best suit the current target appearance, thus obtaining online adaption and boosting tracking performance. To our knowledge, this is the first work trying to learn the general video target appearance distribution using GAN. Experiments on the well-known OTB benchmark [14] show that our tracker outperforms many state-of-the-art trackers

2. TASK-GUIDED GAN (TGGAN)

In this section, we describe our TGGAN in details. The TGGAN includes a generator network and a discriminator network which consists of an adversarial branch and a task-guided branch. The task-guided branch (Siamese matching network) must be well trained in advance to guide the

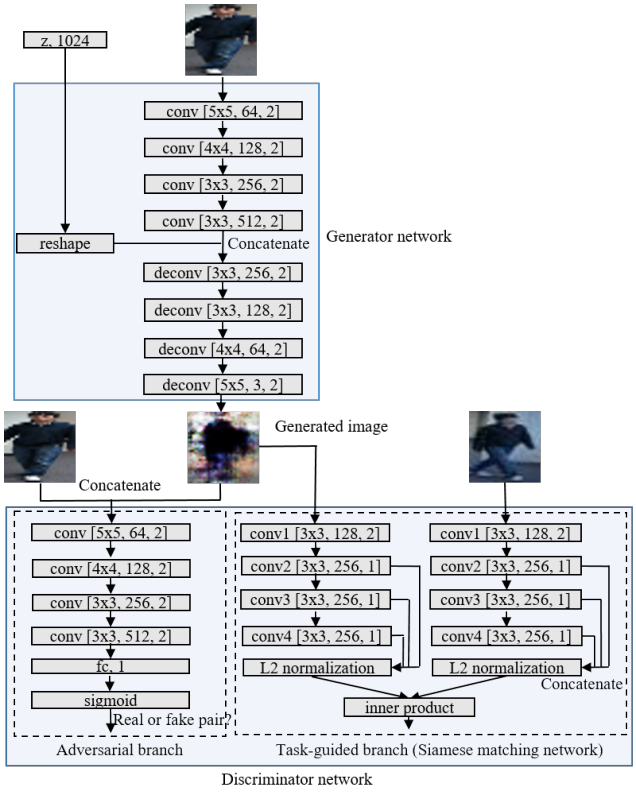

Fig. 1. The proposed task-guided GAN. 'conv', 'deconv' and 'fc' stand for convolution, deconvolution (transposed convolution), and fully-connected layers, respectively. 'z' represents the random noise vector. Numbers in square brackets are kernel size, number of outputs, and stride, respectively. Note that the two branches of Siamese network share weights.

learning of the generator network. We alternatively train the generator network and the adversarial branch of the discriminator network until the task-guided branch cannot tell the generated image is target or not.

2.1. Siamese Matching Network

The Siamese matching network is designed to learn a general similarity function between two images. As shown in Fig. 1, we concatenate the output feature of conv2, conv3, and conv4 to get the final feature which contains both low-level features to capture intraclass variations and high-level features to obtain semantic information. Note that we do not apply any global pooling or flattening operations to get single global feature vectors. Instead, we do the matching with every $1 \times 1 \times 768$ feature vector extracted from each 2-D spatial location. This indicates that the model performs at the scale of local patches. We first extract the feature along the feature channel. Then we normalize the extracted features by L2 normalization. Let $f_{i,j}^1$ and $f_{i,j}^2$ denote the normalized features of the two Siamese branches, respectively, where (i,j) denotes the feature vector position in the concatenated feature. Then the matching score of the two features is simply given as,

$$m_{i,j} = {f_{i,j}^1}^{\mathrm{T}} f_{i,j}^2, \quad (1)$$

where the superscript T represents transposition. Since the use of L2 normalization, the matching score $m_{i,j}$ ranges from -1 to 1.

Let p be the label of the input pair. If the two inputs are of the same object, we consider them as positive pair, then $p = 1$. Otherwise, we consider them as negative pair and $p = 0$. We adopt the margin contrastive loss [15] as the loss function of our Siamese matching network,

$$\mathcal{L}_{i,j} = p(1 - m_{i,j})^2 + (1 - p)\max(0, m_{i,j} - \epsilon)^2, \quad (2)$$

where ϵ is the maximum margin that the scores of negative samples should satisfy. In our method, we set ϵ as 0 to make all the matching scores of negative pairs no more than 0. Because randomly picked two patches are uncorrelated in most cases, resulting in a zero matching value. The final loss of the Siamese matching network is the mean of the losses of all feature vectors, $\mathcal{L}_s(D_s(x_1, x_2)) = \sum_{i,j} \mathcal{L}_{i,j}/N$, where N is the number of feature vectors, D_s means the Siamese matching network, and (x_1, x_2) are the input sample pair of D_s.

To obtain training samples, we randomly pick two images from a sequence. Then we extract the ground truth patch in one image as the template and extract patches in the other frame around the ground truth patch as candidates. We calculate IoU (intersection-over-union) rate between the candidate's bounding box and the ground truth patch's bounding box. Those candidate patches having IoU overlap rates with the ground truth larger than 0.75, we consider them as positive. While those patches with IoU overlap rates less than 0.4, we consider them as negative. The network is trained using the Adam [15] algorithm with each mini-batch consisting of 32 positive sample pairs and 96 negative sample pairs. We train the network 15 epochs and each epoch consists of $6.25K$ iterations, where K is the number of training sequences. The initial learning rate is 0.001 and multiplied by 0.6 after each epoch.

Note that once Siamese matching network has finished the training, it should not be updated anymore when training TGGAN.

2.2. TGGAN

The goal of our TGGAN is to learn a generator that simulates the distribution of the target appearances that may undergo through a sequence. The inputs of TGGAN are a target template patch and a random noise vector. Thus we adopt the conditional GANs [13] framework. The objective of a conditional GAN (cGAN) is given as,

$$\mathcal{L}_{cGAN}(G, D_a) = \mathbb{E}_{x,y\sim p_{data}(x,y)}[\log D_a(x,y)] + \mathbb{E}_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1 - D_a(x, G(x,z))], \quad (3)$$

where G represents the generator that learns to map data x from distribution $p_{data}(x)$ and z from noise distribution $p_z(z)$ to the distribution over y, and D_a means the adversarial branch. Eq. (3) only includes the adversarial branch of the discriminator. However, it is well-known that forcing a model to perform additional tasks will improve the performance on the original task [16,17] and also we want the generated images to be qualified for the matching task. Thus, we guide the generator by adding an additional matching task of minimizing the Siamese loss,

$$\mathcal{L}_{siam}(G, D_s) = \mathcal{L}_s(D_s(x, G(x, z))) \quad (4)$$

Then the final objective is

$$G^* = \arg\min_G \max_{D_a} \mathcal{L}_{cGAN}(G, D_a) + \mathcal{L}_{siam}(G, D_s) \quad (5)$$

We alternatively train the generator network and the adversarial branch of the discriminator network. The adversarial branch is to identify the input pair is real pair or fake pair. We randomly select two frames from one sequence and crop the ground truth patches in them, forming the real pair. For the fake pair, we crop the ground truth patch of a frame and pass it with a 1024-D random vector through the generator to obtain a generated patch. The generated patch together with the input patch form the fake pair. We use binary cross-entropy loss function for the adversarial branch. Each mini-batch has 64 sample pairs. The numbers of real pairs and fake pairs are equal. For the training of generator, we use losses of both adversarial branch and Siamese matching branch. Each mini-batch has 64 sample pairs for each branch. For the Siamese branch, the ratio of positive pairs and negative pairs is 1:1. We use Adam [18] algorithm with learning rate 0.0002 as described in appendix A of ACGAN paper [17]. The network is trained for 10000 iterations when we find that the task-guided branch cannot distinguish the generated patches from the real ones.

2.3. Implementation Details

Fig. 1 illustrates the network architecture of the proposed TGGAN. We add batch normalization [19] after all conv layers except for the first conv layer in the generator and the first conv layer in the adversarial branch [20]. We use leaky ReLUs [21] with slope 0.2 as the activation function of each conv layer except for the output layer of the generator where tanh function is used. After the last conv layer of the generator, a full-connected layer is applied to map to a 1 dimensional output, followed by sigmoid function.

3. TRACKING

In every frame, we draw $D\ (=450)$ candidate samples from the image frame according to the state vector $\mathrm{Q}_t = \left[l_x, l_y, s\right]^{\mathrm{T}}$, where subscript t represents the frame number, and l_x, l_y, s denote x, y translations and scale variation, respectively. The state vector is modeled by the Gaussian distribution, i.e., $N(\mathrm{Q}_t; \mathrm{Q}_{t-1}, \varphi)$, where Q_{t-1} is the estimated target state vector in the last frame, and φ is a diagonal covariance matrix. And the diagonal elements are $(0.25r^2, 0.25r^2, 0.09)$, where r is the mean of current target height and width. Extracting the patch according to the state vectors, we can get a candidate set $\mathrm{C} = \{c_1, c_2, \dots, c_D\}$.

We remain a generated template set $\mathrm{T}^g = \{t_1^{\mathrm{g}}, t_2^{\mathrm{g}}, \dots, t_K^{\mathrm{g}}\}$ which has $K(=10)$ templates. A coarse-to-fine strategy is applied in finding the best candidate. In each frame, we first pass the candidates and the ground truth template in the first frame through the Siamese network. The matching score is given as,

$$M(t, c_d) = \textstyle\sum_{i,j} m_{i,j}/N, \quad (6)$$

where t is the ground truth patch in the first frame and $d \in \{1,2,\dots,D\}$. Then we pick out $B(=50)$ candidates which have the highest matching scores, forming the coarsely selected set $\mathrm{C}^s = \{c_1^{\mathrm{s}}, c_2^{\mathrm{s}}, \dots, c_B^{\mathrm{s}}\}$. Every candidate in C^s is matching to every generated template in T^g. The final matching score of c_b^{s} is given by,

$$M_{fine}(t, \mathrm{T}^g, c_b^{\mathrm{s}}) = \sigma M(t, c_b^{\mathrm{s}}) + \rho \textstyle\sum_k M\left(t_k^{\mathrm{g}}, c_b^{\mathrm{s}}\right)/K, \quad (7)$$

where $b \in \{1,2,\dots,B\}$. Parameter σ and ρ balance the contribution of the ground truth template in the first frame and the generated templates. In our case, we set both values to be 0.5. The best candidate is picked out by,

$$\hat{c} = \underset{c_b^{\mathrm{s}}}{\operatorname{argmax}}\, M_{fine}(t, \mathrm{T}^g, c_b^{\mathrm{s}}). \quad (8)$$

After estimation in each frame, we use the first ground truth patch and 50 random vectors as inputs of the generator to generate 50 patches. Then we choose 2 of the patches which have the highest matching scores with current estimated target to replace the two in T^g which have the lowest matching scores with current estimated target.

4. EXPERIMENTS

This work is implemented using Keras toolbox [22] and performed on a single NVIDIA GeForce GTX Titan X GPU. The tracker runs at 3.1 fps. We use sequences in ALOV [23] and VOT2015 [24] to train the model and evaluate it on OTB50 [14] and OTB100 [14]. The OTB50 is a more challenging subset of OTB100 and is different from the OTB2013 [25]. Note that we pick out the sequences that both exist in the training data sets and test data sets. For convictive evaluation, we do not use them to train the model. Our tracker is compared with eight state-of-the-art trackers, including CFNet_conv3 [12], SINT+ [11], ACFN [26], staple [27], CNN-SVM [28], SiamFC3s [10], MEEM [29], and DSST [30]. The CFNet_conv3 tracker gets the best overall performance in all the variants of CFNet [12]. The SINT+ tracker is an improved version of SINT [11]. It uses sampling strategy in [31] and optical flow to filter out bad candidates. For ablation study, we compare the proposed TGGAN tracker with its variant which only uses the template from the first frame and does not use the generated online adaptive templates. We name this variant FixedT. The templates and

Table 1. AUC score of different tracking methods in terms of different attributes on the OTB100 dataset. The best three results are shown in red, blue, and green fonts.

Attribute	DSST	MEEM	CNN-SVM	ACFN	staple	SiamF-C3s	CFNet_conv3	FixedT	SINT+	TGGAN
OCC	0.449	0.508	0.514	0.537	0.545	**0.547**	0.526	**0.582**	**0.582**	**0.589**
DEF	0.415	0.492	0.547	0.533	0.552	0.510	0.524	**0.587**	**0.558**	**0.597**
FM	0.447	0.542	0.546	0.562	0.537	0.568	0.555	**0.581**	**0.570**	**0.609**
IV	0.556	0.522	0.537	0.567	0.596	0.574	0.542	**0.605**	**0.636**	**0.614**
SV	0.466	0.472	0.489	0.547	0.522	0.556	0.546	**0.599**	**0.565**	**0.622**
MB	0.469	0.556	**0.578**	0.562	0.546	0.550	0.540	**0.591**	**0.591**	**0.630**
BC	0.523	0.519	0.548	0.573	**0.574**	0.523	0.561	0.555	**0.590**	**0.576**
LR	0.383	0.364	0.403	0.425	0.418	**0.592**	**0.552**	0.536	0.521	**0.594**
IPR	0.502	0.529	0.548	0.543	0.552	0.557	**0.571**	0.560	**0.599**	**0.576**
OPR	0.470	0.525	0.548	0.543	0.534	0.558	0.556	**0.583**	**0.598**	**0.603**
OV	0.386	0.488	0.488	0.496	0.481	0.506	0.456	**0.544**	**0.553**	**0.582**
Overall	0.513	0.530	0.554	0.573	0.581	0.582	0.589	**0.606**	**0.592**	**0.618**

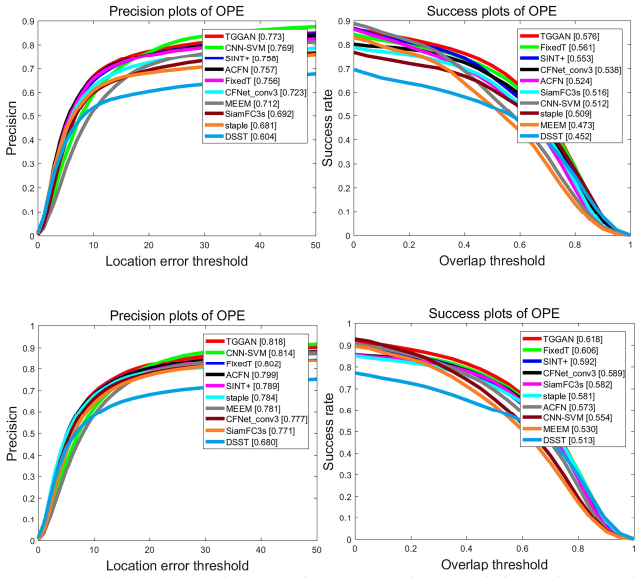

Fig. 2. First row and second row are the precision plots and success plots of OTB50 and OTB100, respectively. The legend in the success plots reports the AUC scores. The legend in the precision plots reports the percentage of frames within the threshold of 20 pixels.

candidates are all resize to a fixed size 32×32. We use success plots and precision plots of OPE [14] to evaluate our tracker as illustrated in Fig. 2. The legend in the precision plots reports the percentage of frames whose distance precision is below 20 pixels. The legend in the success plots shows the area-under-curve (AUC) scores of the success plots.

We can see that our proposed TGGAN achieves the best performance in terms of both precision score and AUC score on both OTB50 dataset and OTB100 dataset. It is worth noting that the FixedT tracker performs the second best in terms of the AUC score. This indicates the effectiveness of our Siamese matching network architecture. The output features obtain both low-level local information and high-level semantic information by concatenating the conv features of conv2, conv3, and conv4 layers. Meanwhile, we separately match the features from different location. This local patch based operation helps the model achieve robustness. In the precision plots, the FixedT performs worse than TGGAN and some other trackers. This verifies the effectiveness of the generated templates improving the tracking accuracy.

Table 1 illustrates the AUC score of different trackers in terms of different tracking attributes on the OTB100 dataset. The attributes include occlusion (OCC), deformation (DEF), fast motion (FM), illumination variation (IV), scale variation (SV), motion blur (MB), background clutters (BC), low resolution (LR), in-plane rotation (IPR), out-plane rotation (OPR), and out-of-view (OV). In eight of the eleven attributes our tracker achieves the best AUC scores. In IV, BC, and IPR, our tracker obtains the second best AUC scores. The overall performance indicates that the proposed TGGAN tracker can deal with various challenging tracking scenarios.

5. CONCLUSION

In this paper, we propose a robust Siamese matching network and a task-guided GAN for visual tracking. The proposed Siamese network takes advantages of both low-level local information and high-level semantic information. Meanwhile, the matching is performed at the scale of local patches for robustness. The task-guided GAN network tries to model the appearance distribution that a target may undergo through a sequence. After the training of TGGAN, we use the generator to generate templates that best suit the current target appearance. Since the templates are generated from the ground truth template in the first frame, they are robust and can alleviate the drift problem, while still catching adaptivity. Numerous experiments show the effectiveness of the proposed Siamese matching network and the task-guided GAN.

Acknowledgment: This work was supported by the Major Science Instrument Program of the National Natural Science Foundation of China (61527802), and the General Program of National Nature Science Foundation of China (61371132, 61471043).

6. REFERENCES

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

[2] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *CVPR*, 2009, pp. 983-990.

[3] X. Zhou, L. Xie, P. Zhang, and Y. Zhang, "An ensemble of deep neural networks for object tracking," in *ICIP*, 2015, pp. 843-847.

[4] J. Guo, and T. Xu, "Deep Ensemble Tracking," *IEEE Signal Process. Lett.,* vol. 24, no. 10, pp. 1562-1566, 2017.

[5] X. Mei, and H. Ling, "Robust visual tracking and vehicle classification via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 33, no. 11, pp. 2259-2272, 2011.

[6] J. Guo, T. Xu, Z. Shen, and G. Shi, "Visual Tracking Via Sparse Representation With Reliable Structure Constraint," *IEEE Signal Process. Lett.,* vol. 24, no. 2, pp. 146-150, 2017.

[7] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *CVPR*, 2015, pp. 2544-2550.

[8] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 37, no. 3, pp. 583-596, 2015.

[9] K. Chen, and W. Tao, "Once for All: a Two-flow Convolutional Neural Network for Visual Tracking," *arXiv preprint arXiv:1604.07507*, 2016.

[10] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," in *ECCV*, 2016, pp. 850-865.

[11] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese Instance Search for Tracking," in *CVPR*, 2016, pp. 1420-1429, 2016.

[12] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for Correlation Filter based tracking," in *CVPR*, 2017, pp.2805-2813.

[13] M. Mirza, and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[14] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 37, no. 9, pp. 1834-1848, 2015.

[15] S. Chopra, R. Hadsell, and Y. Lecun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," in *CVPR*, 2005, pp. 539-546.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *arXiv preprint arXiv:1703.06870*, 2017.

[17] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," *arXiv preprint arXiv:1610.09585*, 2016.

[18] D. Kingma, and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] S. Ioffe, and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML*, 2015, pp. 448-456.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

[22] F. Chollet, "Keras,", https://github.com/ fchollet/keras, 2015.

[23] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 36, no. 7, pp. 1442-1468, 2014.

[24] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2015 challenge results," in *ICCVW*, pp. 1-23.

[25] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013, pp. 2411-2418.

[26] J. Choi, H. Chang, S. Yun, T. Fischer, Y. Demiris, and J. Choi, "Attentional correlation filter network for adaptive visual tracking," in *CVPR*, 2017, pp. 4807-4816.

[27] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *CVPR*, 2016, pp. 1401-1409.

[28] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *ICML*, 2015, pp. 597-606.

[29] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *ECCV*, 2014, pp. 188-203.

[30] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *BMVC*, 2014.

[31] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *ICCV*, 2015, pp. 3101-3109.