Appears in IEEE Intl. Conf. on Multimedia and Expo (ICME) 2021

META-GRAPH ADAPTATION FOR VISUAL OBJECT TRACKING

Qiangqiang Wu *Antoni B. Chan*

Department of Computer Science, City University of Hong Kong
qiangqwu2-c@my.cityu.edu.hk abchan@cityu.edu.hk

ABSTRACT

Existing deep trackers typically use offline-learned backbone networks for feature extraction across various online tracking tasks. However, for unseen objects, offline-learned representations are still limited due to the lack of adaptation. In this paper, we propose a Meta-Graph Adaptation Network (MGA-Net) to adapt backbones of deep trackers to specific online tracking tasks in a meta-learning fashion. Our MGA-Net is composed of a gradient embedding module (GEM) and a filter adaptation module (FAM). GEM takes gradients as an adaptation signal, and applies graph-message propagation to learn smoothed low-dimensional gradient embeddings. FAM utilizes both the learned gradient embeddings and the target exemplar to adapt the filter weights for the specific tracking task. MGA-Net can be end-to-end trained in an offline meta-learning way, and runs completely feed-forward for testing, thus enabling highly-efficient online tracking. We show that MGA-Net is generic and demonstrate its effectiveness in both template matching and correlation filter tracking frameworks.

Index Terms— Meta-graph adaptation, meta learning, single object tracking, real-time tracking

1. INTRODUCTION

Visual object tracking is a fundamental task that has gained much attention due to its wide usage in many other tasks. Recent progress on deep learning-based trackers [1, 2, 3] shows that deep trackers can better handle the general challenges (e.g., rotation and background clutter) in online tracking, due to their learned good feature representations. However, besides these explicit challenges, another main difficulty in visual tracking is the uncertainty of the object type to be tracked, This definition distinguishes the tracking task from other computer vision tasks like object detection and segmentation, where the algorithms only focus on finding objects belonging to pre-defined categories, making the tracking task hard to solve by applying existing deep learning techniques.

One common strategy is to treat visual tracking as a general similarity learning problem. Existing such end-to-end trainable deep trackers (e.g., template matching [1, 4] and correlation filter [2, 5] methods) typically maintain a fixed offline learned backbone network θ, which is trained on a large-scale annotated training, aiming to learn a generally good feature representation for online template matching. However, for some unseen instances or objects with unseen categories, their feature representations of the backbone network are still not discriminative enough, since important features may be missed. As illustrated in Fig. 1, we show the limited generalization ability of SiamFC [1] to some unseen objects.

To adapt the backbone network θ to a specific online tracking task, one naïve approach is to use a gradient-based optimization method like stochastic gradient descent (SGD) to fine-tune the backbone network in an online manner,

$$\hat{\theta} = \theta + \theta_{update} = \theta - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla L_i(\theta, \eta), \qquad (1)$$

where $L_i(\theta, \eta)$ indicates the loss of the i-th training sample with the correlation filter layer η and α is a learning rate. Due to the limited training samples in online tracking, the SGD method suffers from overfitting, thus resulting in tracking performance degradation. Previous works (e.g., using multi-domain learning [3] and meta learning [6]) alleviate this issue by finding a good initial backbone model θ in an offline manner, and hopefully this model can generalize well across various test videos. However, due to the uncertainty of online tracked objects, it is difficult to find such an initial model (or even adapt an initial model) that can generalize well to all categories, instances, and appearance variations of target objects during tracking. Therefore, in this work, instead of finding an optimal offline initial model θ, we focus on learning a specific residual updating term θ_{update}.

In this paper, we improve state-of-the-art deep trackers by dynamically adapting their backbone networks to a specific online tracking task. Given a new target as input, the gradient of the loss function w.r.t. a layer's weights indicates the fit of its offline-learned features to the current target. Thus, we use the loss gradient as an input into a meta-learning network that learns to directly update the weights of the layer. To achieve this, we propose a novel meta-graph adaptation network (called MGA-Net) consisting of two stages. The first stage is a gradient embedding module (GEM), which learns a low-dimensional embedding of the loss gradient that can represent high-level relationships among loss gradients for a variety of objects. Here a graph neural network (GNN) is used in order to aggregate information between similar filters. The

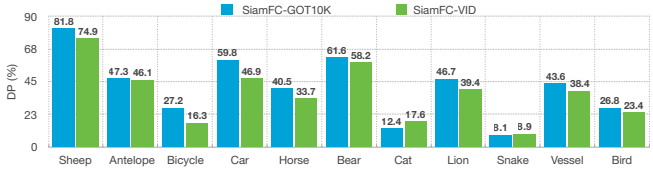

Fig. 1. Object and video domain gap: performance comparison between SiamFC-GOT10K and SiamFC-VID trackers, which are trained on 11 common categories from GOT-10K [7] or VID-2015 [8] (using the same number of videos from each dataset). The two trackers are tested on a held-out set of videos (10% of each category) from GOT-10K. Since the test videos in GOT-10K contain many objects belonging to subcategories that never appear in VID-2015, for these unseen objects, SiamFC-VID cannot generalize well and thus achieves inferior results on the 11 main categories, especially for *Bicycle*, *Car* and *Lion*, as compared to SiamFC-GOT10K.

second stage is the filter adaptation module (FAM), which uses the gradient embedding and target to adapt the feature extraction layer. In the offline training stage, we show that our MGA-Net can be easily incorporated into existing deep tracking frameworks and be effectively trained in an end-to-end meta learning fashion.

In summary, this paper makes the following contributions:

- We propose a novel meta-graph adaptation network (MGA-Net) to effectively adapt backbone feature extractors in existing deep trackers to a specific online tracking task.
- To demonstrate its effectiveness and broad applicability, we apply MGA-Net to three deep trackers, including two template matching trackers (SiamFC [1] and SiamRPN++ [9]) and one end-to-end trainable correlation filter tracker (DCFNet [10]).
- Through extensive evaluations on four popular benchmarks, OTB-2013/15 [11], VOT-2016 [12], and VOT-2018 [13], we show that the MGA-adapted trackers outperform their baselines while running at above real-time speeds. In addition, our MGA-Net can be effectively trained using the same training data as the original trackers, which does not introduce additional training data.

2. RELATED WORK

Deep Visual Tracking. The pioneering work of template-matching tracker is SiamFC [1], which learns a general template-matching function in an offline manner. Based on SiamFC, some improved variants include [4, 9, 14, 15]. Compared with template matching-based deep trackers, deep discriminant correlation filters (DCFs) naturally models target appearance variations via an online learned CF model. Commonly, state-of-the-art DCFs adopt deep features extracted from either a pre-trained classification backbone network [16] or a backbone network that is specifically offline end-to-end

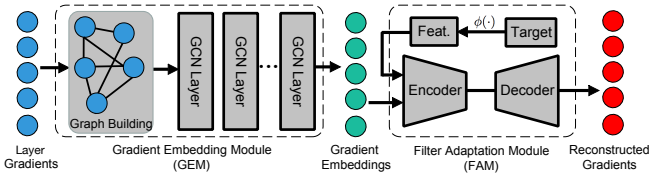

Fig. 2. Pipeline of the proposed meta-graph adaptation network (MGA-Net).

trained [17, 10] for the tracking task. In recent years, much efforts [18] have been made on how to effectively learn a CF model in online tracking, such that the learned CF model can handle well with online target appearance variations. Similar to Siamese network trackers, DCFs usually maintain a fixed backbone network for target feature extraction in online tracking, thus lead to suboptimal results. We show that our MGA-Net can improve both deep Siamese and DCF trackers by changing the underlying features extracted from their offline-learned backbone networks.

Meta Learning for Visual Tracking. The representative work is MetaTracker [6], which follows the idea in MAML, i.e., learning a general initial tracking model and performs first-frame adaptation for online tracking. However, due to the uncertainty of tracked objects, finding such an optimal initial model becomes very challenging and the first-layer adapted model can also be easily corroded during the online tracking. GradNet [19] and UpdateNet [20]) mainly focus on learning to update online target template for Siamese trackers. Different from these methods, our method aims to perform online backbone network adaptation, which can also be applied to these methods. MLT [21] is a closely related method to ours, which uses additional convolutional filters to encode online target information. But this way cannot change the underlying features of the backbone network, thus still leading to limited online adaptation. For our method, instead of adding additional filters, we directly modify the underlying feature extractors of the backbone network for more effective online adaptation.

3. METHODOLOGY

In this section, we introduce a novel meta-graph adaptation network (MGA-Net) to adapt backbone networks of existing deep trackers to specific online tracking tasks. The whole framework is shown in Fig. 2.

3.1. Gradient Embedding Learning

We propose a gradient embedding module (GEM) to effectively learn gradient embeddings that summarize the gradient information in a low-dimensional space. The input into GEM are the raw loss gradients of filters in a layer of the backbone. The gradients can usually serve as a signal to judge whether the backbone needs to be adapted or how to adapt the backbone, i.e., large loss gradients typically indicate a large domain gap between offline training and online testing. Instead of encoding the gradient embedding of each filter separately, it is better to capture co-dependencies between similar filters

in the layer. In this way, global updating information can be well gained for each filter, thus facilitating the learning.

To achieve this, we implement GEM with a graph neural network (GNN) architecture [22]. Let $\delta_i \in \mathbb{R}^{k\times k\times c_{in}\times c_{out}+c_{out}}$ (i.e., k is kernel size, c_{in} and c_{out} are the input and output channel numbers) be the gradients of the i-th filter at a specific layer. We firstly construct an undirected graph $G(V, E)$, where $V = \{v_1, v_2, ..., v_N\}$ are nodes that represent the N filters in the layer, whose initial representations are the raw gradients, $v_i = \delta_i$. E represents the set of graph edges, whose edge weights $E_{i,j}$ are computed according to the similarity between the corresponding two filters,

$$E_{i,j} = \max(0, \cos(W_i, W_j)) \tag{2}$$

where W_i denotes the filter weight of the i-th filter and cos is the cosine similarity function. Note that we remove the connection (i.e., $E_{i,j} = 0$) between two nodes if their filter weights are not positively correlated. For these two filters, their features are not typically activated at the same time due to the ReLU non-linearity zeroing out any negative responses, and thus they capture different features of the object. The initial node representations in the graph $G(V, E)$ are processed iteratively with a neighborhood aggregation schema, for $l \in \{1, \cdots, k\}$,

$$h_i^l = F^l\Big(\big(1+\mu^l\big)\cdot h_i^{l-1} + \sum_{j=1}^{N} E_{i,j}\cdot h_j^{l-1}\Big), \tag{3}$$

where $i \in \{1, \cdots, N\}$, l is the layer number of GEM and h_i^l is the aggregated value for node i in layer l, with initial value $h_i^0 = v_i$. F^l denotes a multi-layer perceptron (MLP) in the l-th layer, and μ^l is a learnable weight parameter. Our GEM iteratively aggregates the gradient information in neighboring nodes to obtain the global gradient embedding h_i^k in the last layer. In the next subsection, we show how to leverage the learned h_i^k to perform filter adaptation in the backbone.

3.2. Filter Adaptation

To adapt the filters in the specific layer of the backbone network, a novel filter adaptation module (FAM) is presented. FAM is composed of an encoder E_n and a decoder D_e. E_n improves the previous learned gradient embeddings $\{h_i^k\}_{i=1}^N$ by explicitly incorporating target-specific information, to obtain a gradient-feature embedding. D_e takes these gradient-feature embeddings as input to reconstruct the one-step gradient updates of each filter.

In more detail, we first use a shallow feature extractor $\phi(z)$ (i.e., containing three convolutional layers with kernel sizes of 3×3 and two fully-connected layers) to extract features of target z, which encodes the target exemplar into a low-dimensional vector $\phi(z)$. We input both $\{h_i^k\}_{i=1}^N$ and $\phi(z)$ into E_n to obtain the gradient-feature embeddings $\{\hat{h}_i\}$,

$$[\hat{h}_1, \cdots, \hat{h}_N] = E_n\Big(h_1^k, \cdots, h_N^k, \phi(z)\Big). \tag{4}$$

Since the gradient-feature embeddings have already encoded both the global gradient information and the target information, we then separately input them to D_e to generate the final update gradients:

$$\hat{\delta}_i = D_e\big(\hat{h}_i\big), \quad \forall i = 1, 2, ..., N, \tag{5}$$

and the generated gradients are used to update the filter weights:

$$\hat{W}_i = W_i + \hat{\delta}_i, \quad \forall i = 1, 2, ..., N. \tag{6}$$

The adapted filters $\{\hat{W}_i\}_{i=1}$ in the backbone network are then directly used in the online test video. Note that the original gradient δ_i could also be used to update the weights. However, the generated gradients $\hat{\delta}_i$ should be better since: 1) they are end-to-end learned to perform an optimal one-step update (see next section); 2) they explicitly encode the target information; 3) the GEM regularizes the raw gradient by aggregating gradients of similar filters and embedding into a low-dimensional space.

3.3. End-to-end Offline Meta Learning

We next show how to effectively offline train MGA-Net to learn to update backbone networks of deep trackers during online tracking. Let $\mathcal{T}$ be a deep tracker with an offline-learned backbone network θ and $\mathcal{L}$ be the loss function. Given a target exemplar z, the loss of the tracker $\mathcal{T}$ estimated on the searching patch x is $\mathcal{L}(z, x, \theta)$. By back-propagating the loss, the loss gradients δ of filters in θ can be obtained. By inputing the gradients δ and z into MGA-Net, we can get the adapted backbone network $\hat{\theta}$.

Additionally, we sample another test patch q sampled in a future frame to construct triplet training samples (z, x, q). The tracker $\mathcal{T}$ with the adapted backbone network $\hat{\theta}$ is tested on both the searching patches x and q, and the overall meta loss is

$$\mathcal{L}_{all} = \mathcal{L}(z, x, \hat{\theta}) + \mathcal{L}(z, q, \hat{\theta}). \tag{7}$$

Using $\mathcal{L}_{all}$, we can train the proposed MGA-Net in an end-to-end offline manner. Note that the second loss term $\mathcal{L}(z, q, \hat{\theta})$ in (7) is a meta test loss, which makes our MGA-Net generalize well to future frames, and avoids overfitting on a frame.

4. PROPOSED META-GRAPH ADAPTATION TRACKERS

We apply our MGA-Net to 3 representative trackers, i.e., SiamFC [1], DCFNet [10] and SiamRPN++ [9]. The overall adaptation pipeline of the proposed trackers is shown in Fig. 3.

4.1. Meta-Graph Adaptation for SiamFC

SiamFC [1] compares a target exemplar z and a searching image x using a cross-correlation operation. The tracking output is $S_\theta(x, z) = f_\theta(z) * f_\theta(x)$, where f_θ denotes the backbone with parameters θ, $*$ is cross-correlation and $S_\theta(x, z)$

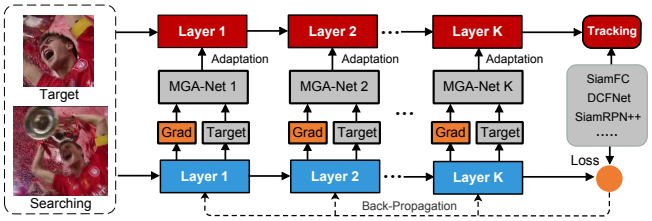

Fig. 3. The overall adaptation pipeline of the proposed meta-graph adaptation (MGA) trackers. The original backbone network is in blue, and the adapted backbone is in red.

indicates the response map. The backbone θ of SiamFC is offline trained using the logistic loss $\mathcal{L}_{sim}$ between $S_\theta(x,z)$ and ground truth $y(x)$.

Offline MGA-Net learning. For the training of MGA-Net in SiamFC, the loss $\mathcal{L}$ described in Section 3.3 is the SiamFC loss, $\mathcal{L}_{sim}(y(x), S_\theta(z,x))$. Thus, the training loss for MGA-Net in (7) is

$$\mathcal{L}_{all} = \mathcal{L}_{sim}(y(x), S_{\hat{\theta}}(z,x)) + \mathcal{L}_{sim}(y(q), S_{\hat{\theta}}(z,q)). \quad (8)$$

Online MGA-Net tracking. We first sample the target exemplar z as described in SiamFC [1] and a searching image x_1 centered at the target center position in the first frame, which are used by MGA-Net to obtain the updated backbone $\hat{\theta}$. In the following frames, we use the updated backbone $\hat{\theta}$ for template matching. During tracking, we store intermediate reliable target samples whose response values are larger than the target response in the first frame. After every T frames, the latest stored target sample x_t is used for backbone adaptation, yielding a new $\hat{\theta}$ for tracking.

4.2. Meta-Graph Adaptation for SiamRPN++

Different from SiamFC, SiamRPN++ [9] uses two training losses: a smooth L_1 loss [23] $\mathcal{L}_{reg}$ for scale regression, and a logistic loss [1] $\mathcal{L}_{cls}$ for classification.

Offline MGA-Net learning. We set the loss function $\mathcal{L}$ for meta-learning to the loss of SiamRPN++, $\mathcal{L}_{reg}(z,x,\theta) + \lambda\mathcal{L}_{cls}(z,x,\theta)$, where λ is a parameter set according to [9]. Thus the meta-loss of MGA-Net in (7) for SiamRPN++ is

$$\mathcal{L}_{all} = \mathcal{L}_{reg}(z,x,\hat{\theta}) + \mathcal{L}_{reg}(z,q,\hat{\theta}) \quad (9)$$
$$+\lambda(\mathcal{L}_{cls}(z,x,\hat{\theta}) + \mathcal{L}_{cls}(z,q,\hat{\theta})). \quad (10)$$

Online MGA-Net tracking. We use the same tracking procedure as described above for MGA-Net with SiamFC.

4.3. Meta-Graph Adaptation for DCFNet

DCFNet [10] improves the basic correlation filter (CF) framework by learning the backbone network feature extractor f_θ in an end-to-end offline manner. Given a target exemplar z and a searching image x, the learning loss for DCFNet is

$$\mathcal{L}_{cf}(z,x,\theta) = ||\mathbf{r}_z * f_\theta(x) - y(x)||^2 + \gamma||\theta||^2, \quad (11)$$

where $\mathbf{r}_z$ is the CF learned from the target exemplar z (via the ridge regression problem in [24]), γ is a regularization parameter, and $y(x)$ is the desired Gaussian label map corresponding to the target image x.

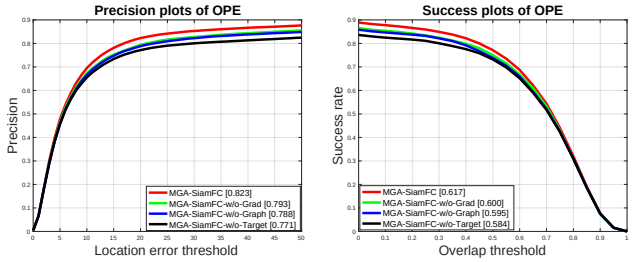

Fig. 4. Distance precision and overlap success plots obtained by the variants of our MGA-SiamFC on OTB-2015 [11].

Offline MGA-Net learning. For DCFNet, the overall meta loss of MGA-Net in (7) is

$$\mathcal{L}_{all} = \mathcal{L}_{cf}(z,x,\hat{\theta}) + \mathcal{L}_{cf}(z,q,\hat{\theta}). \quad (12)$$

Online MGA-Net tracking. We first collect tracking samples in the first 10 frames. In the 10-th frame, the sample with the highest response value is selected for adaptation, resulting in the updated backbone $\hat{\theta}$. We use $\hat{\theta}$ for further correlation tracking, which follows the same step as described in the original DCFNet paper.

5. EXPERIMENTS

5.1. Implementation Details

Offline Learning. GEM is implemented as a 5-layer GNN, where each layer consists of a two-layer MLP and the number of neurons in each layer is 128. For FAM, the encoder E_n and the decoder D_e are three-layer MLPs with a single hidden layer of 256 units. The outputs of E_n and D_e are 128-D feature vectors and the adapted gradients respectively. Except for the output layers in the above modules, each layer is followed by a batch normalization and ReLU activation functions. We use VID-2015 [8] to train all MGA-trackers. We sample various triplet training samples abcn(z,x,q) in each video. The cropping strategy is the same as original trackers. The MGA-Net is trained for 20 epochs with a mini-batch of 8 triplets, using the Adam optimizer with a learning rate of 1×10^{-4} and decay rate of 0.866 every 5 epochs.

Online Tracking. MGA-Net performs adaptation on the last two layers of SiamFC, and adapts the tracker in every $T = 25$ frames. For DCFNet, since its backbone feature extractor only contains two convolutional layers, we adapt both layers for online tracking. For SiamRPN++, we focus on the variant using ResNet-50 as the backbone. We use MGA-Net to adapt the filters in the second layer of ResNet-50, which contains 3 bottlenecks and 10 convolutional layers in total. The other tracking parameters in MGA-trackers are set as the same as their original trackers. Note that for SiamRPN++, we use the offline-learned ResNet-50 backbone[1] for evaluation on all datasets.

5.2. Ablation Study

We choose the basic MGA-SiamFC for ablation study on the OTB-2015 [11] dataset.

[1]The model is *siamrpn_r50_l234_dwxcorr*, which can be found in PySOT zoo (`https://github.com/STVIR/pysot/`).

Table 1. Performance comparison between the proposed MGA-trackers and their baseline trackers on 4 tracking benchmarks. For the OTB datasets, we report DPR/AUC scores, and for VOT datasets, the expected accuracy overlap (EAO) score is reported. Our MGA improves perfomance over the baselines (highlighted in bold), while maintaining similar tracking speed.

Trackers	MGA-SiamFC	MGA-SiamRPN++	MGA-DCFNet	SiamFC	SiamRPN++	DCFNet
OTB-2013	**85.5/63.7**	**91.2/69.2**	**88.5/66.5**	80.9/60.7	89.5/67.1	85.4/64.7
OTB-2015	**82.3/61.7**	87.6/**67.1**	**82.6/62.4**	77.1/58.2	87.9/66.6	81.2/62.1
VOT-2016	**0.263**	**0.475**	**0.244**	0.235	0.461	0.233
VOT-2018	**0.237**	**0.425**	**0.181**	0.188	0.410	0.170
Avg. FPS	77.0	65.0	255.9	86.0	71.4	266.3

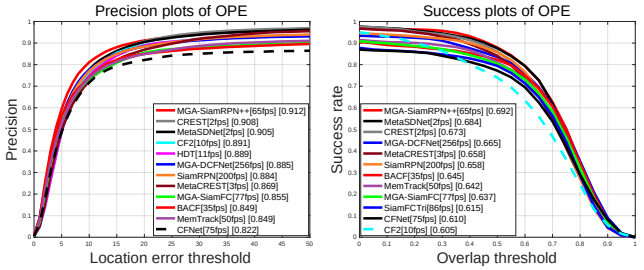

Fig. 5. Precision and success plots on the OTB-2013 dataset using one-pass evaluation.

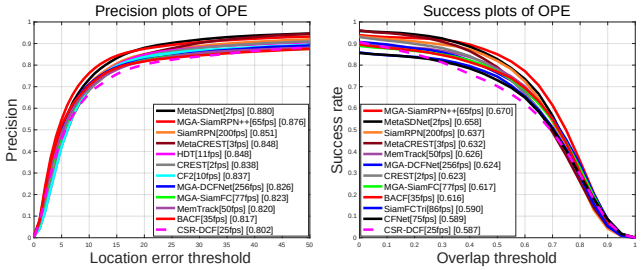

Fig. 6. Precision and success plots on the OTB-2015 dataset using one-pass evaluation.

Impact of graph learning. To show the impact of graph learning in GEM, we replace GNN with a fully-connected network with the same architecture. As shown in Fig. 4, MGA-SiamFC-w/o-Graph achieves inferior results (78.8%/59.5%) to our MGA-SiamFC (82.3%/61.7%), The main reason is that graph message propagation can effectively regularize the raw gradient by aggregating gradients of similar filters, which is better than learning separately.

Impact of target input. In Fig. 4, without using the target input, we can find that MGA-SiamFC-w/o-Target significantly degrades the performance. This shows the importance of using the target input, which facilitates the gradient embedding to better reconstruct the weight update.

Impact of gradient input. We remove GEM and only use FAM inputing with the target exemplar for filter adaptation. In Fig. 4, the corresponding tracker MGA-SiamFC-w/o-Grad can still achieve better results than its baseline tracker SiamFC, which demonstrates that the image-level target information can provide rich information for adaptation.

5.3. Comparison with Baseline Trackers

We compare our MGA-SiamFC, MGA-DCFNet and MGA-SiamRPN++ trackers with their baseline trackers (i.e., SiamFC, DCFNet and SiamRPN++) on OTB-13/15 [25, 11] and VOT-16/18 [12, 13]. For OTB, we report both the distance precision rates (DPR) and the area under the curve (AUC). For VOT, the expected average overlap (EAO) is used.

Table 2. Results on VOT-2016 based on EAO, accuracy (Acc.) and robustness (Rob.). The best results are highlighted by bold.

Trackers	MGA-SiamFC	MGA-SiamRPN++	MGA-DCFNet	MemTrack	MDNet	MetaSDNet	SiamMask	SiamRPN++
EAO↑	0.263	**0.475**	0.244	0.273	0.257	0.310	0.442	0.461
Acc.↑	0.55	0.64	0.54	0.53	0.54	0.54	**0.67**	0.64
Rob.↓	0.440	**0.196**	0.41	0.38	0.34	0.26	0.23	0.20
Avg. FPS	77	65	**256**	50	1	1	11	71

The results are presented in Table 1. By applying our MGA method, the performances of the three baseline trackers are improved, which is because the adapted backbones in our MGA-trackers provide more rich target-specific information, which enables the trackers to gain more discriminative features of target objects, thus enabling robust online tracking.

5.4. Comparison with State-of-the-Art Trackers

In this subsection, we extensively compare the proposed MGA-trackers with state-of-the-art trackers.

OTB: We select two groups of trackers to compare with our MGA-trackers on OTB-2013 [25] and OTB-2015 [11]: 1) 8 *real-time* trackers including SiamRPN [14], MemTrack [4], SiamFC-tri [15], BACF [18], CSR-DCF [26], CFNet [17], Staple [27] and KCF [24], and 2) 5 state-of-the-art deep trackers that are *not real-time*, including MetaCREST [6], CF2 [16], HDT [28], MetaSDNet [6] and CREST [29].

Fig. 5 shows the overall performance achieved by our MGA-trackers and the 11 top-performing tested trackers on OTB-2013. MGA-SiamRPN++ achieves the best DPR (91.2%) and AUC (69.2%) scores, while running at the high speed of 65 FPS. Compared with CF-based meta tracker (i.e., MetaCREST) that aims to learn an initial model, our MGA-DCFNet learns to adapt the model in an online manner, and achieves better performance than MetaCREST.

Figs. 6 shows the overall comparison on OTB-2015. Our MGA-SiamRPN++ achieves the best AUC scores (67.0%). Without time-consuming online optimization steps, MGA-SiamRPN++ runs significantly faster than MetaSDNet (i.e., about 32 times faster). Compared with the CFNet that uses a CF layer for adaptation, MGA-SiamFC achieves the better results, showing the effectiveness of our method.

VOT-2016: We evaluate the proposed three MGA-trackers on the VOT-2016 [12] dataset with the comparison to five recent trackers, including SiamRPN++ [9], SiamMask [30], MetaSDNet [6], MemTrack [4] and MDNet [3]. The overall performance of each tracker is shown in Table 2. Our MGA-

SiamRPN++ outperforms SiamRPN++ and SiamMask, as well as other adaptive-based (MemTrack and MDNet) or meta learning-based trackers (MetaSDNet).

6. CONCLUSIONS

This paper addresses the online adaptation problem of how to effectively and efficiently adapt backbone feature extractors used in existing deep trackers to a specific online tracking task. To achieve this, we propose a general meta-graph adaptation (MGA) method and demonstrate its effectiveness on two deep tracking frameworks, i.e., the template-matching framework based on Siamese network and the end-to-end trainable correlation filter framework. By applying the proposed method to these two frameworks, we propose three MGA trackers, i.e., MGA-SiamFC, MGA-DCFNet and MGA-SiamRPN++. We show that the proposed MGA-trackers can both keep very close running speed with their baseline trackers, while also achieving better performance.

Acknowledgements. This work was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11212518).

7. REFERENCES

[1] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, and P.H.S. Vedaldi, "Fully-convolutional siamese networks for object tracking," in *ECCVW*, 2016.

[2] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *CVPR*, 2017, pp. 21–26.

[3] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *CVPR*, 2016, pp. 4293–4302.

[4] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *ECCV*, 2018.

[5] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *CVPR*, 2017, pp. 4335–4343.

[6] E. Park and A. C. Berg, "Meta-tracker: Fast and robust online adaptation for visual object trackers," in *ECCV*, 2018, pp. 569–585.

[7] L. Huang, X. Zhao, and K. Huang, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," *PAMI*, 2019.

[8] O. Russakovsky, J. Deng, and et al, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[9] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *CVPR*, 2019.

[10] Q. Wang, J. Gao, and J. Xing, "Dcfnet: Discriminant correlation filters network for visual tracking," in *arXiv:1704.04057*, 2017.

[11] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *PAMI*, vol. 37, no. 9, pp. 1834–1848, 2015.

[12] M. Kristan, A. Leonardis, and J. Metas, "The visual object tracking vot2016 challenge results," in *ICCV Workshop*, 2016, pp. 777–823.

[13] M. Kristan, A. Leonardis, and M. Felsberg, "The sixth visual object tracking vot2018 challenge results," in *ECCVW*, 2018.

[14] B. Li, W. Wu, Z. Zhu, and J. Yan, "High performance visual tracking with siamese region proposal network," in *CVPR*, 2018.

[15] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in *ECCV*, 2018.

[16] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *ICCV*, 2015, pp. 3074–3082.

[17] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *CVPR*, 2017.

[18] H. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *ICCV*, 2017.

[19] P. Li, B. Chen, and W. Ouyang, "Gradnet: Gradient-guided network for visual object tracking," in *ICCV*, 2019.

[20] L. Zhang, A. G.-Garcia, and F. Khan, "Learning the model update for siamese trackers," in *ICCV*, 2019.

[21] J. Choi, J. Kwon, and K. Lee, "Deep meta learning for real-time target-aware visual tracking," in *ICCV*, 2019.

[22] K. Xu, W. Hu, J. Leskovec, and S. jegelka, "How powerful are graph neural networks," in *arXiv:1810.00826*, 2018.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time onject detection with region proposal network," in *NIPS*, 2015.

[24] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *PAMI*, vol. 37, no. 3, pp. 583–596, 2015.

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

[26] A. Lukezic and T. Vojir, "Discriminative correlation filter with channel and spatial reliability," in *CVPR*, 2017.

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

[28] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged deep tracking," in *CVPR*, 2016.

[29] Y. Song, C. Ma, L. Gong, J. Zhang, R. W.H. Lau, and M.-H. Yang, "Crest: Convolutional residual learning for visual tracking," in *ICCV*, 2017.

[30] Q. Wang, L. Zhang, and L. Bertinetto, "Fast online object tracking and segmentation: A unifying approach," in *CVPR*, 2019.