GradNet: Gradient-Guided Network for Visual Object Tracking

Peixia Li†, Boyu Chen†, Wanli Ouyang§, Dong Wang†*, Xiaoyun Yang‡, Huchuan Lu†

† Dalian University of Technology, China, § The University of Sydney, Australia,
‡ China Science IntelliCloud Technology Co., Ltd

{pxli, bychen}@mail.dlut.edu.cn, wanli.ouyang@sydney.edu.au, {wdice,lhchuan}@dlut.edu.cn, xiaoyun.yang@intellicloud.ai

Abstract

The fully-convolutional siamese network based on template matching has shown great potentials in visual tracking. During testing, the template is fixed with the initial target feature and the performance totally relies on the general matching ability of the siamese network. However, this manner cannot capture the temporal variations of targets or background clutter. In this work, we propose a novel gradient-guided network to exploit the discriminative information in gradients and update the template in the siamese network through feed-forward and backward operations. To be specific, the algorithm can utilize the information from the gradient to update the template in the current frame. In addition, a template generalization training method is proposed to better use gradient information and avoid overfitting. To our knowledge, this work is the first attempt to exploit the information in the gradient for template update in siamese-based trackers. Extensive experiments on recent benchmarks demonstrate that our method achieves better performance than other state-of-the-art trackers. The source codes are available at https://github.com/LPXTT/GradNet-Tensorflow.

1. Introduction

Visual object tracking is an important topic in computer vision, where the target object is identified in the initial video frame and successively tracked in subsequent frames. In recent years, deep networks [37, 3, 19, 44, 23, 39] have significantly improved the tracking performance due to their representation prowess.

There are two groups of deep-learning-based trackers. The first group [36, 28, 32, 4] improves the discriminative ability of deep networks by frequent online update. They utilize the first frame to initialize the model and update it every few frames. Timely online update enables trackers to capture target variations but also requires more computational time. Therefore, the speed of these trackers generally cannot meet the real-time requirements.

Figure 1. The motivation of our algorithm. Images in the first and third column are target patches in SiameseFC. The other images show absolute values of their gradients, where the red regions have large gradient. As we can see, the gradient values can reflect the target variations and background clutter.

Siamese-based trackers are representative in the second group [3, 44, 22] which is totally based on offline training. They learn the similarity between objects in different frames through massive offline training. During online testing, the initial target feature is regarded as template and used to search the target in the following frames. These methods need no online updating, thus, they usually run at real-time speeds. However, these methods cannot adapt to appearance variations of target without important online adaptability, thereby increasing the risk of tracking drift. To solve this problem, many researches [16, 45, 40] present different mechanisms to update template features. However, these methods only focus on combining the previous target features, ignoring the discriminative information in background clutter. This results in a big accuracy gap between the siamese-based trackers and those with online update.

Generally, gradients are calculated through the final loss which considers both positive and negative candidates.

*Corresponding Author: Dr. Dong Wang

Table 1. The number of backward iterations to update the template of SiameseFC. 'LR' means learning rate; '$n\times$' means n times the basic learning rate; 'ITERs' means the needed iterations to converge. There is no proper step to converge by one iteration.

LR	1×	3×	5×	7×	9×	10×	30×	50×	70×	90×	100×	500×	1000×	3000×	5000×
ITERs	449	136	77	64	60	58	59	51	54	56	55	54	61	67	∞

Thus, gradients contain the discriminative information to reflect the target variations and distinguish the target from background clutter. As shown in Figure 1, when objects are occluded with noise or similar objects coexist at the neighborhood of the target, the absolute value of gradients at these locations are prone to be higher. The high value in gradients can force the template to focus on these regions and capture the core discriminative information. Most gradient-based trackers [36, 32] concentrate on hand-designed optimization algorithms, such as momentum [34], Adagrad [11], ADAM [20] and so on. These algorithms need hundreds of iterations to converge, which lead to more computation and a lower speed. How to take a trade-off between the speed and accuracy of update is still a problem.

If we expect to reduce the number of training iterations but still keep online update through gradients, the extreme case is to adapt the template through one backward propagation. However, training by one backward propagation is a difficult task. As shown in Table 1, there is no proper learning rate to make the template of SiameseFC converge through one iteration. Generally, even with the optimal step length, moving according to the gradient at only one iteration cannot update the template properly, because the normal gradient-based optimization is a nonlinear process. On the other hand, we can learn a nonlinear function by CNNs, which simulates the non-linear gradient-based optimization by exploring the rich information in gradients. Therefore, we propose a gradient-guided network (GradNet) to perform gradient-guided adaptation in visual tracking. The GradNet integrates the adaptation process that consists of two feed-forward and one backward calculation, simplifying the process of gradient-based optimization.

It is a very tough task to train a robust GradNet due to two main reasons. The first reason is that the network is prone to use the appearance of the template instead of using the gradient for tracking (details can be found in Section 3.3), because learning to use the gradients is more difficult than learning to use appearance. The second reason is that the network is prone to overfit. As shown in Figure 2, the model with normal training (Ours-T) can quickly get a low distance error but its test accuracy is not promising, compared with our model. To handle these issues, we propose a template generalization method to effectively explore gradient information and avoid overfitting.

The major contributions can be summarized as follows:

- A GradNet is proposed to conduct gradient-guided template updating for visual tracking.
- A template generalization method is proposed to ensure strong adaptation ability and avoid overfitting.
- Extensive experiments conducted on four popular benchmarks show that the proposed tracker achieves promising results at a real-time speed of ***80fps***.

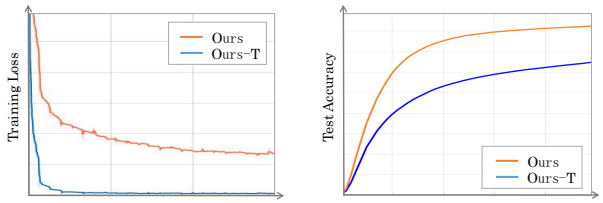

Figure 2. The training and testing plots of models through normal training (Ours-T) and our training method (Ours). The left map shows the error between the predicted map and the real map during training and the right map shows the accuracy during testing.

2. Related Work

2.1. Siamese Network based Tracking

SiameseFC [3] is the most representative trackers based on template matching. Bertinetto *et al.* [3] present a siamese network with two shared branches to extract features of both the target and the search region. During online tracking, the template is fixed as the initial target feature and the tracking performance mainly relies on the discriminative ability of the offline-trained network. Without online updating, the tracker achieves beyond real-time speed. Similarly, SINT [33] also designs a network to match the initial target with candidates in a new frame. Its speed is much lower because hundreds of candidate patches are sent into the network instead of one search image. Another siamese-based tracker is GOTURN [17] which proposes a siamese network to regress the target bounding box with a speed of 100fps. All these methods are lack of important online updating. The fixed model cannot adapt to appearance variations, which makes the tracker easily disturbed by similar instances or background noise. In this paper, we choose SiameseFC as our basic model and propose a gradient-guided method to update the template.

2.2. Model Updating in Tracking

Timely updating is essential to keep trackers robust. There are three main dominant strategies of model updating, including template combination, gradient-descent based and correlation-based strategies.

Template Combination. Algorithms [16, 45] based on template combination aim to effectively combine the target features from previous frames. Guo *et al.* [16] propose a fast transformation learning model to enable effective online learning from previous frames. Zhu *et al.* [45] utilize the optical flow information to convert templates and integrate them according to their weights. All these methods focus on using the information of templates, which ignore the background clutter. Different from these methods, we take full use of the discriminative information in backward gradients instead of just integrating previous templates.

Gradient-descent based approaches. Deep trackers [36, 32] based on gradient descent explore the discriminative information in backward gradients to update the model through hundreds of iterations. Wang *et al.* [36] train two separate convolutional layers to regress Gaussian maps with the initial frame and update these layers every few frames. Similarly, Song *et al.* [32] also utilize a number of gradient descent iterations in initialization and online update procedures. These trackers need many training iterations to capture the appearance variations of the target, which makes the tracker less effective and far from real-time requirements. We propose a GradNet that needs only one backward propagation and two forward propagations to update the template effectively. Besides, our template generalization method for handling overfitting is not investigated in existing works.

Correlation based Tracking. Correlation based trackers [18, 26, 10, 43, 42, 6] train classifier through circular convolution, which can be quickly calculated in Fourier domain. The final classifier is trained and updated by solving the closed-form solution of the optimization function. The classifier training cannot be simulated totally by deep networks, so most correlation based trackers just utilize deep networks to extract robust features. Differently, our method aims to update the template in an end-to-end network.

2.3. Gradient Exploiting

Currently, most deep neural networks adopt gradients in offline training based on hand-designed optimization strategies, such as momentum [34], Adagrad [11], ADAM [20] and so on. These methods usually need expensive computation and large-scale data sets. How to accelerate the training of deep networks is a hot topic in computer vision.

Meta Learning. Meta learning approaches can be broadly divided into different categories, including optimization-based methods [1], memory-based methods [31], variable-based methods [14, 30, 24] and so on. Our algorithm can be seen as an improved version of the optimization-based method [1] to adapt to the update task in visual tracking. Our approach has three main differences compared with [1]. First, ours only learns to update template, but not the network branch of search region. This is specifically designed for the tracking task. Second, our update process only contains one iteration instead of multiple iterations. Finally, our training of the optimizer includes second-order gradient which is not used in [1].

Meta Learning for Tracking. Despite the popularity of meta learning in many fields, there are few works [40, 29] applying it to visual tracking. Yang *et al.* [40] design a memory structure to dynamically write and read previous templates for model updating. Differently, we focus on exploring the discriminative information of gradients. Eunbyung *et al.* [29] train the initialization parameters of filters with pixel-wise learning rate offline and utilize a matrix multiplication to update the filters. The update is a linear process. While, our template update is a non-linear process with convolutional layers and Relu. Besides, we use the target feature as the prior information to speed up the update process by providing a good initial value.

3. Proposed Algorithm

The whole pipeline of GradNet is shown in Figure 3, which consists of two branches. One branch extracts features of the search region $\mathbf{X}$ and the other branch generates the template according to the target information and gradients, detailed in Section 3.2. The template generation process consists of initial embedding, gradient calculation and template updating. First, the shallow target feature $f_2(\mathbf{Z})$ is sent to one sub-net $\mathbf{U}_1$ (shown in purple in Figure 3) to obtain an initial template β which is used to calculate the initial loss L. Second, the gradient of the shallow target feature is calculated through backward propagation, and sent to the other sub-net $\mathbf{U}_2$ (shown in orange in Figure 3) for being non-linearly converted to better gradient representation. Finally, the converted gradient is added to the shallow target feature to get an updated target feature which is sent to the sub-net $\mathbf{U}_1$ again to output the optimal template. It should be noted that the two sub-nets in the initial embedding and template update process share parameters. The optimal template is used to search targets on search regions through cross correlation convolution.

3.1. Basic Tracker

We adopt SiameseFC [3] as the basic tracker. $f_x(.)$ is used to model the feature extraction branch for search region, $f_z(.)$ is used to model the feature extraction branch for target region. We assume that the movement of the target is smooth between two consecutive frames. Thus, we can crop a search region $\mathbf{X}$ which is larger than the target patch $\mathbf{Z}$ in the current frame, centered at the target's position in the last frame. The final score map is calculated by:

$$\mathbf{S} = \beta * f_x(\mathbf{X}), \tag{1}$$

where β is the template to perform an exhaustive search over the search region $\mathbf{X}$, $*$ means cross correlation convo-

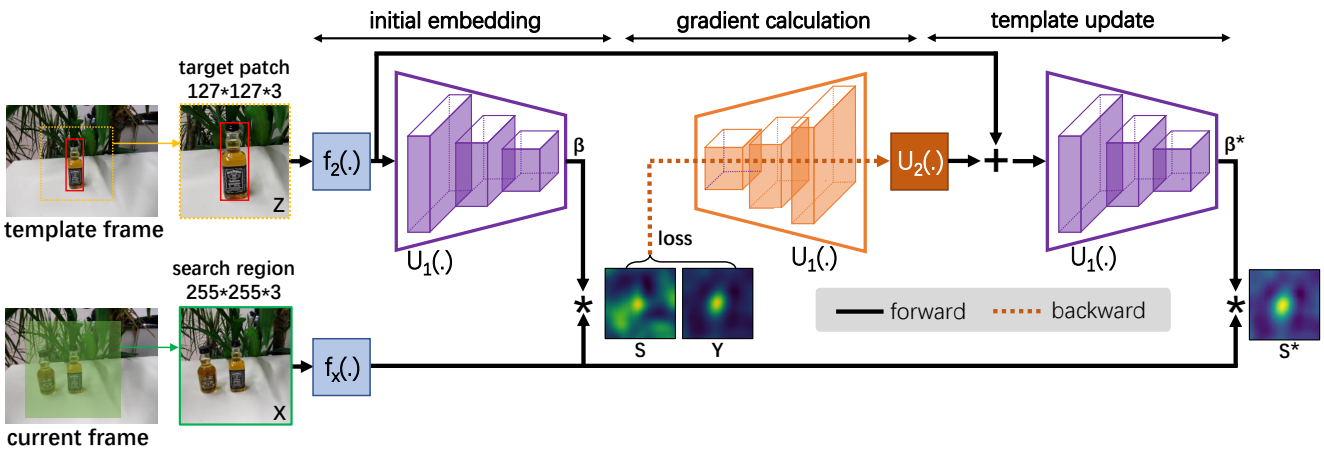

Figure 3. The pipeline of the proposed algorithm, which consists of two branches. The bottom branch extracts the feature of search region $\mathbf{X}$ and the top branch (named update branch) is responsible for template generation. The two purple trapezoids in the figure represent sub-nets with shared parameters; the solid and dotted line represents forward and backward propagation respectively.

lution, $\mathbf{S}$ denotes the score map to find the target. In SiameseFC, the template β is defined as the deep target feature:

$$\beta_{sia} = f_z(\mathbf{Z}), \tag{2}$$

where $\mathbf{Z}$ is the target patch in the first frame. In order to improve the discriminative ability of the template β during online tracking, we design the update branch $U(\alpha)$ to explore the rich information in gradients:

$$\beta_{our} = U(\mathbf{Z}, \mathbf{X}, \alpha), \tag{3}$$

where α is the parameter of the update branch which can not only capture the template information in $\mathbf{Z}$ but also the background information in $\mathbf{X}$ through gradients.

3.2. Template Generation

Initial Embedding. Given the image pair $(\mathbf{X}, \mathbf{Z})$, we want to get the optimal template $\beta^\star$ which is suitable to distinguish the target from the background in search region $\mathbf{X}$. First, we get the target feature $f_2(\mathbf{Z})$ (using two convolutional layers) and sent $f_2(\mathbf{Z})$ to the sub-net U_1 to get the initial template β:

$$\beta = U_1(f_2(\mathbf{Z}), \alpha_1), \tag{4}$$

where α_1 is the parameter of U_1. The initial template only contains template information without background information. Thus, we need to explore the discriminative information in gradient to make it more robust. After getting β, the initial score map $\mathbf{S}$ is calculated through equation (1).

Gradient Calculation. Based on the initial score map $\mathbf{S}$ and the training label $\mathbf{Y}$, we can get the initial loss L by:

$$L = l(\mathbf{S}, \mathbf{Y}), \tag{5}$$

where $l(.)$ is logistic loss function. We utilize this loss to calculate the gradient of $f_2(\mathbf{Z})$ and added it to $f_2(\mathbf{Z})$. Then, the updated target feature is obtained by:

$$h_2(\mathbf{Z}) = f_2(\mathbf{Z}) + U_2(\frac{\partial L}{\partial f_2(\mathbf{Z})}, \alpha_2), \tag{6}$$

where α_2 is the parameter of U_2. *Here, the gradient is related to U_1 and used as the input of the sub-net U_2 to calculate the final loss, so the second-order guidance is introduced in the parameter training of the sub-net U_1.*

Template Update. Finally, we send the updated target feature $h_2(\mathbf{Z})$ to the sub-net U_1 again to obtain the optimal template $\beta^\star$ and the final score map $S^\star$ by:

$$\begin{aligned} \beta^\star &= U_1(h_2(\mathbf{Z}), \alpha_1), \\ \mathbf{S}^\star &= \beta^\star * f_x(\mathbf{X}). \end{aligned} \tag{7}$$

The optimal score map $\mathbf{S}^\star$ is utilized to estimate the target position. Our goal is to let $\mathbf{S}^\star$ have the highest value at the target position and lower values at other positions. Thus, we utilize the loss which is calculated by $\mathbf{S}^\star$ to train the update branch:

$$\arg\min_\alpha \sum l(\mathbf{S}^\star, \mathbf{Y}). \tag{8}$$

To our knowledge, this work is the first attempt to exploit the discriminative information of gradients to update the template in SiameseFC. To simplify the introduction of template generation process, we just utilize one image pair here. In the next subsection, we will discuss the training method more generally and detailedly.

3.3. Template Generalization

Problem of Basic Optimization. Image pairs from different videos and their training labels form the training set

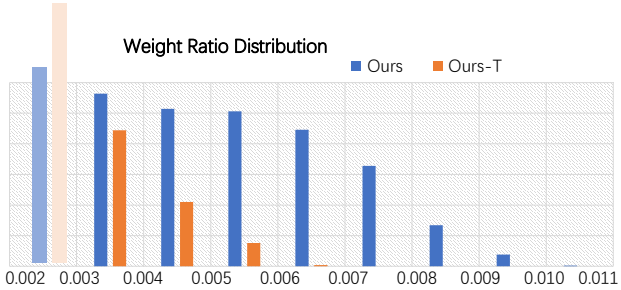

Figure 4. The distribution of weight ratio between gradients and features. The weight ration is calculated by the absolute value of α_2, which reflects the proportion of the gradient during the template update. The rectangles at different positions represent the number of points in those ranges.

$T = \{(\mathbf{X}_1, \mathbf{Z}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Z}_2, \mathbf{Y}_2), \ldots, (\mathbf{X}_n, \mathbf{Z}_n, \mathbf{Y}_n)\}$, $\mathbf{X}_i$ is search region which is larger than target patch $\mathbf{Z}_i$, $\mathbf{Y}_i$ is training label and n is the number of training samples. It should be noted that $\mathbf{X}_i$ and $\mathbf{Z}_i$ are from different frames of the same video, while $\mathbf{X}_i$ and $\mathbf{X}_j$ $(i \neq j)$ are from different videos. One simple idea to train our network is to utilize image pairs $(\mathbf{X}_i, \mathbf{Z}_i, \mathbf{Y}_i)$ in the training set T to get optimal template $\beta_i^\star$ and final score maps $\mathbf{S}_i^\star$ by equations (4−7). The update branch is trained through:

$$\arg\min_\alpha \sum_{i=1}^{n} l\left(\mathbf{S}_i^\star, \mathbf{Y}_i\right). \tag{9}$$

This method has two main problems according to our experiment. The first one is that the update branch of the network is prone to focus on the template appearance instead of the gradient, because learning to use the gradient is harder than modeling the similarity metric. As shown in Figure 4, the network trained without template generalization has lower weight ratio of gradients. This means that the network focuses less on gradients. The second one is that the network cannot avoid overfitting under this training process as shown in Figure 2.

Template Generalization. Our goal is forcing the update branch to focus on gradients and avoiding overfitting. Based on these requirements, we propose a template generalization method which adopts search regions from different videos to obtain a versatile template and make it perform well on all search regions in each training batch. We show the training process of our model without template generalization (a) and our model with template generalization (b) in Figure 5 based on four image pairs. The main difference is that we utilize one template (instead of four templates) to search targets on four images from different videos.

We choose k ($k = 4$ in Figure 5) training image pairs from the training set T to form a training batch and utilize the target patch $\mathbf{Z}_1$ in the first image pair to calculate the target feature $f_2(\mathbf{Z}_1)$. The initial template β_1 can be obtained

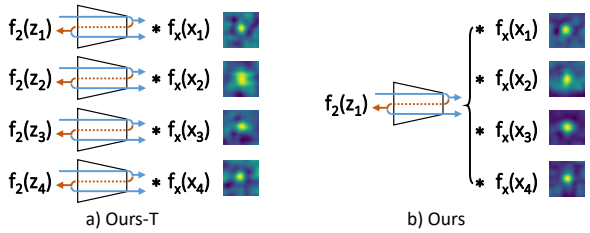

Figure 5. Illustration of 'Ours-T' and 'Ours' on exploiting templates. 'Ours-T' denotes training without template generalization; 'Ours' represents training through template generalization.

by equation (4). Here, β_1 means the template which is calculated through $\mathbf{Z}_1$. Then, we utilize β_1 to find the target on all search regions:

$$\mathbf{S}_i = \beta_1 * f_x(\mathbf{X}_i), i = 1, 2, ...k. \tag{10}$$

Then, we can obtain the initial loss by equation (5) and update the template β_1 through equations (6, 7). After obtaining the updated template $\beta_1^\star$, we utilize it to search the target in all search regions $(\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_k)$ and train the update branch through equation (9). In this way, the $\beta_1^\star$ is required to track the targets in $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_k$ simultaneously. To clarify, we show the details in Algorithm 1.

The template generalization offers the target feature with multiple search regions and aims to obtain a general template feature which performs well on all search regions. This strategy can force the network to focus on the gradients during offline training, because the initial target features are misaligned and the gradients are aligned. The sub-nets U_1 and U_2 need to correct the initial misaligned template according to the gradients and thereby obtaining a great power to update templates according to gradients. As shown in Figures 2 and 4, the template generalization algorithm can effectively avoid overfitting and pay attention on gradients.

3.4. Online Tracking

After offline training, the update branch is totally fixed and used for initialization and update during online testing .

Initialization. Given the ground truth in the first frame, we crop a target patch $\mathbf{Z}_1$ and a search region $\mathbf{X}_1$ as inputs of the network. Then, we can obtain the optimal template $\beta^\star$ according to equations (4−7). Besides, the updated target features $h_2(\mathbf{Z_1})$ is calculated through equation (6) and used to update the template in the following frames.

Online Update. We update the template $\beta^\star$ with one reliable training sample through one iteration. We save the reliable sample $(\mathbf{X}_i, \mathbf{Y}_i)$ according to tracking results and use it to update the current template $\beta^\star$ based on equations (4−7) (replacing $f_2(\mathbf{Z})$, $\mathbf{X}$, $\mathbf{Y}$ with $h_2(\mathbf{Z})$, $\mathbf{X}_i$, $\mathbf{Y}_i$). Namely, we obtain updated feature $h_2(\mathbf{Z}_1)$ through the initial frame. Then, the update branch of network is used

Algorithm 1 Offline training the update branch

Input: Training samples $(\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_n)$ from different videos and gaussian maps $(\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n)$
Output: Trained weights α for the update branch.

Initialize the update branch with weights α^0.
Initialize the feature extraction part of the tracker with parameters from SiameseFC [3].
Crop template images $\mathbf{Z}$ and search regions $\mathbf{X}$ from the training samples to construct the training set $T = \{(\mathbf{X}_1, \mathbf{Z}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Z}_2, \mathbf{Y}_2), \ldots, (\mathbf{X}_n, \mathbf{Z}_n, \mathbf{Y}_n)\}$.
while not converged **do**
1. Randomly select k training samples from T.
2. Utilize the update branch to get β_1 and $\beta_1^\star$.
for $i \in 0, \ldots, k$ **do**
(a). $\beta_1 = U_1(f_2(\mathbf{Z}_1), \alpha_1)$
(b). $\mathbf{S}_i = \beta_1 * f_x(\mathbf{X}_i)$
(c). $L = \sum_{i=1}^{k} l(\mathbf{S}_i, \mathbf{Y}_i)$
(d). Get $h_2(\mathbf{Z}_1)$ according to equation (6).
(e). $\beta_1^\star = U_1(h_2(\mathbf{Z}_1), \alpha_1)$
end for
3. Train the update branch by minimizing the loss.
for $i \in 0, \ldots, k$ **do**
(a). $\mathbf{S}_i{}^\star = \beta_1^\star * f_x(\mathbf{X}_i)$
(b). $L^\star = \sum_{i=1}^{k} l(\mathbf{S}_i{}^\star, \mathbf{Y}_i)$
(c). Minimize $L^\star$ to update α^0 by SGD.
end for
end while

to update $h_2(\mathbf{Z}_1)$ according to the reliable sample $(\mathbf{X}_i, \mathbf{Y}_i)$ and produce optimal templates $\beta^\star$ for the regression part.

3.5. Implementation Details

The feature extraction $f_x(.)$ for the search region consists of five convolutional layers with the same structure and parameters as SiameseFC [3]. The shallow target features $f_2(.)$ are from the second convolutional layers of SiameseFC. There are three convolutional layers in U_1 which have the same structure with the last three layers of SiameseFC. The kernel size of the convolutional layer in U_2 is 3×3. The size of template β and $\beta^\star$ is 6×6 and the size of score map is 17×17. During tracking, we update the template $\beta^\star$ every 5 frames. The reliable training sample is chosen according to the max value of the score map. We set the max value of the score map in the first frame as a threshold $thre$. If the max value of the current score map is larger than $thre * 0.5$, we think that the result is accurate and crop the training sample $\mathbf{X}_t$ as the reliable training sample. The scale evaluate, learning rate and training epoch in the proposed method are the same as those in SiameseFC [3]. To take the trade-off between the fast adaptation and error accumulation, the final template is obtained by combining the initial template and $\beta^\star$. We only train the network on ILSVRC2014 VID dataset and the whole network is fixed during inference.

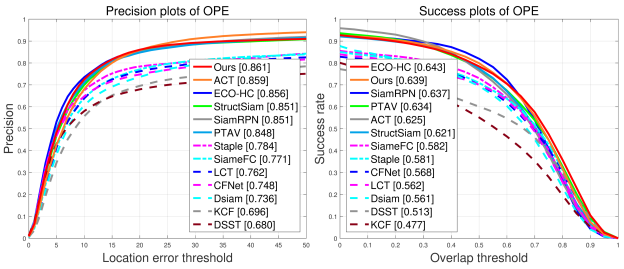

Figure 6. Precision and success plots on the OTB-2015 dataset.

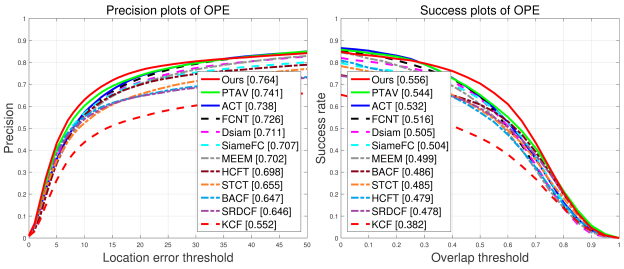

Figure 7. Precision and success plots on the TC128 dataset.

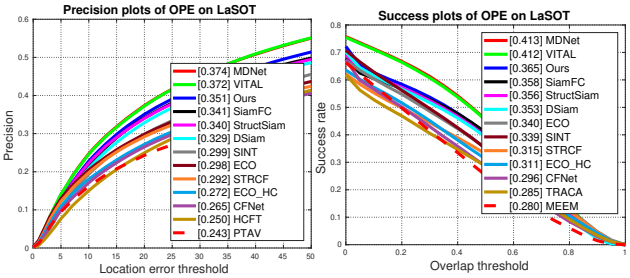

Figure 8. Precision and success plots on the LaSOT dataset.

4. Experiments

Our tracker is implemented in Python with the Tensorflow framework, which runs at **80***fps* with an intel i7 3.2GHz CPU with 32G memory and a Nvidia 1080ti GPU with 11G memory. We compare our tracker with many state-of-the-art trackers with real-time performance (i.e., their speeds are faster than 25*fps*) on recent benchmarks, including OTB-2015 [38], TC-128 [25], VOT-2017 [21] and LaSOT [12].

4.1. Evaluation on the OTB-2015 dataset

The OTB-2015 [38] dataset is one of the most popular benchmarks, which consists of 100 challenging video clips annotated with 11 different attributes. We refer the reader to [38] for more detailed information. Here, we adopt both success and precision plots to evaluate different trackers on OTB-2015. The precision plot reports the percentages that the center location errors are smaller than certain thresh-

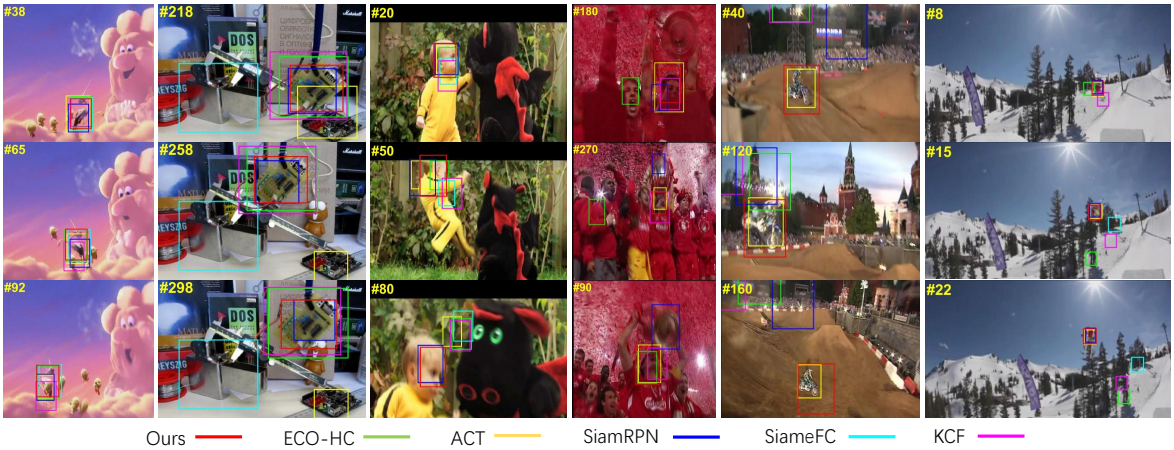

Figure 9. Representative visual results of different tracking algorithms on the OTB-2015 dataset.

Table 2. The accuracy (A), robustness (R) and expected average overlap (EAO) scores of different trackers on VOT2017.

Trackers	A	R	EAO
Ours	0.507	0.375	0.247
SiamRPN	0.490	0.460	0.244
CSRDCF++	0.459	0.398	0.212
SiamFC	0.502	0.604	0.182
ECO_HC	0.494	0.571	0.177
Staple	0.530	0.688	0.170
KFebT	0.451	0.684	0.169
SSKCF	0.530	0.656	0.164
CSRDCFf	0.475	0.646	0.158
UCT	0.490	0.777	0.145
MOSSEca	0.400	0.810	0.139
SiamDCF	0.503	0.988	0.135

olds. Whereas the success plot reports the percentages of frames where the overlap between the predicted and the ground truth bounding boxes is higher than a series of given ratios. We compare our algorithm with twelve state-of-the-art trackers including nine real-time deep trackers (ACT [5], StructSiam [44], SiamRPN [22], ECO-HC [7], PTAV [13], CFNet [35], Dsiam [16], LCT [27], SiameFC [3]) and three traditional trackers (Staple [2], DSST [8], KCF [18]).

Figure 6 illustrates the precision and success plots of all compared trackers over OTB-2015, which shows the proposed tracker achieves very good performance (merely a slightly lower than ECO-HC in success). Especially, our tracker performs significantly better than the baseline model (SiameseFC) by almost 8% in precision and 6% in success. To facilitate more detailed analysis, we demonstrate the visual results of some representative methods in Figure 9. From these figures, we can see that our method can well handle various challenging factors and consistently achieve good performance.

4.2. Evaluation on the TC-128 dataset

The TC128 [25] dataset consists of 128 fully-annotated image sequences with 11 various challenging factors, which is larger than OTB-2015 and focuses more on color information. We also adopt both success and precision plots to evaluate different trackers (the same evaluation protocol as OTB-2015). We compare our algorithm with eleven trackers, including ACT [5], PTAV [13], Dsiam [16], SiameFC [3], HCFT [26], FCNT [36], STCT[37], BACF [15], SRDCF [9], KCF [18] and MEEM [41]. Figure 7 shows that our tracker achieves the best results in terms of both precision and success criterion.

4.3. Evaluation on the VOT2017 dataset

The VOT2017 [21] dataset contains 60 short sequences annotated with 6 different attributes. According to its evaluation protocol, the tested tracker is re-initialized whenever a tracking failure is detected. In this benchmark, the accuracy (A) and robustness (R) as well as expected average overlap (EAO) are three important criterion. Different trackers are ranked based on the EAO criterion. We refer the reader to [21] for more detailed information. In this subsection, we compare our algorithm with top ten trackers reported in the VOT2017 real-time Challenge [21] and another state-of-the-art tracker SiamRPN [22]. Table 2 shows that our tracker achieves the best performance in terms of EAO while maintaining a very competitive accuracy and robustness. The EAO of our tracker is higher than the winner (CSRDCF++) of the VOT2017 real-time Challenge by 3.5%. Our tracker can also perform better than SiamRPN whose training data (over 100,000 videos) is much larger than ours (about 4,000 videos).

4.4. Evaluation on the LaSOT dataset

The LaSOT [12] dataset is a very large-scale dataset consisting of $1,400$ sequences with 70 categories and more than 3.5M frames in total. The average frame length of this dataset is more than $2,500$ frames. Up to now, this dataset is the largest for visual tracking. Following one-pass evaluation, different trackers are compared based on three criteria including precision, normalized precision and success. We also adopt precision and success plots to compare 35 trackers and show the performance of the top 12 trackers in Figure 8 (more compared results are presented in the supplementary material). From Figure 8, we can see that our tracker performs the third-best in this dataset. Although MDNet and VITAL achieve better accuracies than our tracking algorithm, their speeds are far from the real-time requirement (MDNet, *1fps* and VITAL, *1.5fps*).

4.5. Ablation Analysis

Self-comparison. To verify the contribution of each component in our algorithm, we implement and evaluate several variations of our approach (Ours) on OTB-2015. These versions include: (1) 'Ours w/o M': GradNet without template generalization training process; (2) 'Ours w/o MG': GradNet removed template generalization training process and gradient application. It can be seen as SiameseFC with two unshared branches; (3) 'Ours w/o U': the proposed method without template update; (4) 'Ours w 2U': the two sub-nets (in purple) in Figure 3 do not share parameters; (5) 'Ours-baseline': SiameseFC.

Table 3. Precision and success scores on OTB-2015 for different variations of our algorithm.

Variations	**PRE**	**IOU**	**FPS**
Ours	0.861	0.639	80
Ours w/o M	0.823	0.615	80
Ours w/o MG	0.717	0.524	94
Ours w/o U	0.775	0.552	85
Ours w 2U	0.833	0.628	80
Ours-baseline	0.771	0.582	94

The performance of all variations and our final method is reported in Table 3, from which we can see that all components facilitate improving the tracking accuracy. For examples, the comparison of the 'Ours w/o M' and final methods demonstrates the template generalization training method could effectively learn an expected GradNet. With the same amount of training data, 'Ours' improves the precision and IOU score of 'Ours-baseline' about 9% and 5% respectively, which demonstrates the effectiveness of the GradNet.

Training Analysis. To further analyze the template generalization, we show the initial score map $\mathbf{S}$ and the optimal score map $\mathbf{S}^\star$ of two different training methods in Figure 10.

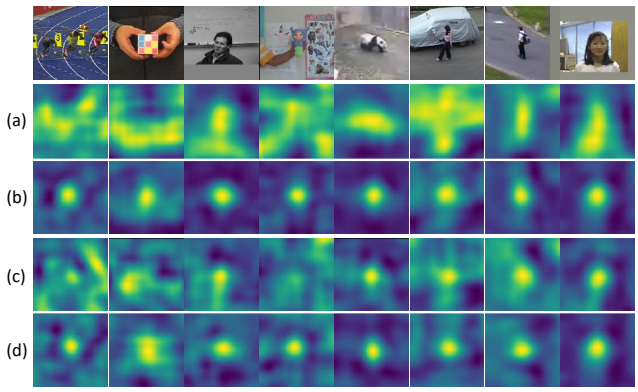

Figure 10. The first row displays the search regions from different videos. (a) and (b) shows $\mathbf{S}$ and $\mathbf{S}^\star$ of our model; (c) and (d) shows $\mathbf{S}$ and $\mathbf{S}^\star$ of the model without template generalization. Model through template generalization can get general initial score maps $\mathbf{S}$ and optimal final score maps $\mathbf{S}^\star$.

The initial score maps of the model with template generalization (a) are noisy score maps where the approximate area of all objects has high response values. After the template updating based on gradients, the promising score maps (b) only have a high response at the target position. Differently, the model without template generalization is likely to output initial score maps (c) with a high response at the target position directly. Thus, we think the model trained by template generalization learns different tasks in the initial embedding and template update processes. During initial embedding, it learns a general template to detect the target and background clutter. This manner provides the model more discriminative gradients. Then, the model learns to update the template based on these gradients in the template update process. The discriminative gradients enable the fast adaptation of the network.

5. Conclusions

In this work, we propose a GradNet for template update, achieving accurate tracking with a high speed. The two sub-nets in GradNet exploits the discriminative information in gradients through feed-forward and backward operations and speeds up the hand-designed optimization process. To take full use of gradients and obtain versatile templates, a template generalization method is applied during offline training, which can force the update branch to concentrate on the gradient and avoid overfitting. Experiments on four benchmarks show that our method significantly improves the tracking performance compared with other real-time trackers.

Acknowledgements The paper is supported in part by Natioal Natural Science Foundation of China No.61725202, 61829102, 61751212 and the Fundamental Research Funds for the Central Universities under Grant Nos. DUT19GJ201, DUT18JC30.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

[2] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016.

[3] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

[4] Boyu Chen, Peixia Li, Chong Sun, Dong Wang, Gang Yang, and Huchuan Lu. Multi attention module for visual tracking. *Pattern Recognition*, 87:80–93, 2019.

[5] Boyu Chen, Dong Wang, Peixia Li, and Huchuan Lu. Real-time 'actor-critic' tracking. In *ECCV*, 2018.

[6] Kenan Dai, Dong Wang, Huchuan Lu, Chong Sun, and Jianhua Li. Visual tracking via adaptive spatially-regularized correlation filters. In *ICCV*, 2019.

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *CVPR*, 2017.

[8] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[9] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015.

[10] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[11] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[12] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. *CoRR*, abs/1809.07845, 2018.

[13] Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *ICCV*, 2017.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[15] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017.

[16] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, 2017.

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

[18] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[19] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, 2017.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[21] Matej Kristan, Ales Leonardis, and et al. Jiri Matas. The visual object tracking VOT2017 challenge results. In *ICCVW*, 2017.

[22] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[23] Peixia Li, Dong Wang, Lijun Wang, and Huchuan Lu. Deep visual tracking: Review and experimental comparison. *Pattern Recognition*, 76:323–338, 2018.

[24] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835, 2017.

[25] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: Algorithms and benchmark. *IEEE Transactions on Image Processing*, 24(12):5630–5644, 2015.

[26] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[27] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *CVPR*, 2015.

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

[29] Eunbyung Park and Alexander C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *ECCV*, 2018.

[30] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *CoRR*, abs/1807.05960, 2018.

[31] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[32] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson W. H. Lau, and Ming-Hsuan Yang. CREST: convolutional residual learning for visual tracking. In *ICCV*, 2017.

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

[34] Paul Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[35] Jack Valmadre, Luca Bertinetto, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[36] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[37] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Sequentially training convolutional networks for visual tracking. In *CVPR*, 2016.

[38] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[39] Bin Yan, Haojie Zhao, Dong Wang, Huchuan Lu, and Xiaoyun Yang. 'skimming-perusal' tracking: A framework for real-time and robust long-term tracking. In *ICCV*, 2019.

[40] Tianyu Yang and Antoni B. Chan. Learning dynamic memory networks for object tracking. In *ECCV*, 2018.

[41] Jianming Zhang, Shugao Ma, and Stan Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014.

[42] Tianzhu Zhang, Si Liu, Changsheng Xu, Bin Liu, and Ming-Hsuan Yang. Correlation particle filter for visual tracking. *IEEE Transactions on Image Processing*, 27(6):2676–2687, 2018.

[43] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Learning multi-task correlation particle filters for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):365–378, 2019.

[44] Yunhua Zhang, Lijun Wang, Dong Wang, Mengyang Feng, Huchuan Lu, and Jinqing Qi. Structured siamese network for real-time visual tracking. In *ECCV*, 2018.

[45] Zheng Zhu, Wei Wu, Wei Zou, and Junjie Yan. End-to-end flow correlation tracking with spatial-temporal attention. In *ECCV*, 2018.