Deformable Object Tracking with Gated Fusion

Wenxi Liu, Yibing Song, Dengsheng Chen, Yuanlong Yu, Shengfeng He, Rynson W.H. Lau

Abstract—The tracking-by-detection framework receives growing attentions through the integration with the Convolutional Neural Network (CNN). Existing methods, however, fail to track objects with severe appearance variations. This is because the traditional convolutional operation is performed on fixed grids, and thus may not be able to find the correct response while the object is changing pose or under varying environmental conditions. In this paper, we propose a deformable convolution layer to enrich the target appearance representations in the tracking-by-detection framework. We aim to capture the target appearance variations via deformable convolution and supplement its original appearance through residual learning. Meanwhile, we propose a gated fusion scheme to control how the variations captured by the deformable convolution affect the original appearance. The enriched feature representation through deformable convolution facilitates the discrimination of the CNN classifier on the target object and background. Extensive experiments on the standard benchmarks show that the proposed tracker performs favorably against state-of-the-art methods.

Index Terms—visual tracking, deformable convolution, gating.

I. INTRODUCTION

Visual object tracking is one of the fundamental problems in computer vision and has many applications, e.g., surveillance security, autonomous driving, and human-computer interactions. In recent years, with the advancement of deep convolutional neural networks (CNNs), which can extract features that are more discriminative than the empirical ones, visual tracking has achieved favorable performance on multiple standard benchmarks.

Despite the demonstrated success, existing state-of-the-art tracking methods suffer from large object appearance variations. The setting of visual tracking uses only the first frame as input, which contains limited representations of the target appearance. Tracking accuracy deteriorates when the target object undergoes severe appearance variations (e.g., pose variation, deformation, and rotation) as shown in Fig. 1). Existing methods are designed without sufficient modeling of such severe appearance variations, degrading the classifier’s ability to discriminate the target object from the background.

As we observe, the deformable objects do not always reside in the regular grids of the image space and the relative locations of the object parts often vary in video frames. However, the existing CNN-based tracking methods lack internal mechanisms to handle deformations, since the standard CNNs perform the convolution operation over a fixed geometric structure. In prior vision tasks, a common solution to this problem is by collecting an extensive amount of training samples. Training data, however, is difficult to collect during online tracking. Therefore, given an input sample in the first frame, the normal convolutional features often fail to model the object with significant pose variation, deformation, or occlusion in tracking.

To tackle this problem, we present a deformable convolutional layer in the CNN-based tracking-by-detection framework in order to model the appearance variations. The deformable convolutional layer enables free-form deformation of the sampling grid. Thus, it can extract features adaptively according to varied object appearances. In particular, when the target object undergoes severe appearance variations, the deformable convolution layer aims to generate a normal response similar to those in the ordinary scenarios by estimating the free-form deformation. Besides, although the training samples collected online are limited and similar, deformable convolution is capable of modeling unseen deformations via online learning.

On the other hand, relying solely on the deformable convolution may have some limitations, e.g., the degradation of scale estimation and localization. This is because it may treat scaling and shifting as some forms of deformations, and thus try to recover those negative samples and categorize them as positive samples. We note that when the target object is in an ordinary scenario with minor appearance variations, the well-trained normal convolutional features are effective. When the target object has significant appearance variations, the deformable convolution will be more effective. Hence, we introduce a soft gate mechanism to balance between the normal convolutional features and the deformable convolutional features. The soft gate approves the fusion between the deformable features and the original ones when the appearance variations of the target object are accurately modeled. Specifically, the soft
gate adaptively blends these two types of features. As a result, the gated fusion of the normal convolutional features and the deformable ones will recover the rapid appearance variations of the target object into ordinary conditions. The gated fusion produces accurate deformable feature maps for the target object, which in turn enrich target appearances and facilitate the classifier prediction. Moreover, the deformable convolutional layer with gated fusion is integrated into the tracking-by-detection framework for end-to-end training and prediction. Extensive experiments on the standard benchmarks indicate that the proposed tracker performs favorably against the state-of-the-art methods.

We summarize the contributions of this work as follows:

- We present a deformable convolutional layer into the CNN-based tracking-by-detection framework to model target appearance variations.
- We propose a gated fusion mechanism to control the effect of the deformable convolutional layer on the output feature maps. It facilitates the classifier’s discrimination on the target object and background.
- Our tracker outperforms state-of-the-art approaches on the standard tracking benchmarks, especially on those challenging scenarios.

In the following sections, we first survey related literatures in Section II. We then present our deformable object tracker with gated fusion in Section III. Finally, we evaluate the performance of our proposed tracker in public benchmarks in Section IV.

II. RELATED WORKS

In visual tracking, state-of-the-art trackers can roughly be categorized into four approaches: regression based methods, Siamese network based methods, reinforcement learning based methods, and tracking-by-detection based methods. In this section, we survey these approaches and prior deformable object trackers.

Regression based methods. The discriminative correlation filter (DCF) based trackers \cite{1, 2} are a kind of regression based approach. They gain much attention in recent years due to their real-time performance. In particular, they regress all the circular-shifted samples into soft labels and transfer the correlation as an element-wise product in the Fourier domain. However, most of DCF based methods hardly achieve the state-of-the-art performance as the tracking-by-detection framework though it reaches the real-time performance. This is because of the boundary effect in DCF and the model overfitting, which caused by the online updating. However, they require a large amount of training data in the offline stage.

Reinforcement learning based methods. Recently, researchers introduce deep reinforcement learning (DRL) into visual tracking. Deep reinforcement learning utilizes the deep neural network to model the active-value function to play games (e.g., Atari \cite{23}), which reaches human-level performance. As a potential research direction in tracking, prior works begin to use DRL to learn robust policies \cite{24, 25, 26}.

Tracking-by-detection based methods. The tracking-by-detection framework is one of the traditional tracking frameworks. It poses the tracking task as a target/background classification problem. Numerous learning schemes have been proposed including P-N learning \cite{27}, online boosting \cite{28}, multiple instance learning \cite{29}, structured SVMs \cite{30}, CNN-SVMs \cite{31}, random forests \cite{32}, domain adaptation \cite{33}, LSTM-based \cite{34}, adversarial learning \cite{35}, and ensemble learning \cite{36}. Our proposed method is based on a CNN-based tracking-by-detection framework. Here we focus on handling the challenging tracking task: tracking deformable objects while maintaining the high-quality tracking performance in ordinary situations. To accomplish this, we introduce the gated fusion module which adaptively approves the fusion of the normal convolutional features and the deformable convolutional features.

Deformable object trackers. Tracking deformable objects is an important problem in visual tracking. There are prior works focusing on tracking non-rigid deformable objects while segmenting their contours \cite{37, 38, 39, 40, 41}. The general approach is to divide the object into parts. Local connectivity is typically applied among different parts of the object and a trade-off of the visual and geometric agreement is then optimized. However, the overall tracking performance of these methods is not comparable to the holistic methods. In this paper, we introduce deformable convolution that can both model the appearance of the deformable object as well as be incorporated into the pipeline of the CNN-based framework and trained in an end-to-end manner. Therefore, it can achieve state-of-the-art performance on public benchmarks.

III. DEFORMABLE TRACKER WITH GATED FUSION

Our approach is designed based on the tracking-by-detection framework, which includes the deformable convolution module and the gating module, as shown in Fig. 2. In particular, one image patch cropped from the current video frame is first fed to the network as input and its feature maps are calculated accordingly via several pretrained convolution layers (i.e., Conv layers). Three branches are then incorporated. In one branch, the deformable convolution module processes the incoming features to produce deformable features. Another
branch keeps the normal convolutional features. The third branch is used in the gating module, which infers the weights that balance the effect of the normal convolutional features and the deformable features. In the final stage, the fused features are sent to the fully connected layers (i.e., FC layers), which serve as a foreground/background classifier to detect the object in an online manner. In the following subsections, we introduce these modules and our tracking framework.

### A. Deformable Convolution

In order to model the variation of the object in tracking, we introduce a deformable convolution module. As shown in Fig. 2, the deformable convolution module is inserted in one of the three branches of the CNN-based framework for extracting the deformable features. We illustrate the principle of how the deformable convolution module works in Fig. 3.

To learn how to deform feature maps, we are inspired by [42]. As shown in Fig. 3, the feature maps (the pink rectangle) are passed through an extra branch containing multiple convolutional layers (the blue rectangles) to regress the 2D offsets of the regular grid sampling locations. Then, the offsets are applied back to the feature maps and produce the new feature maps by resampling. In formal notations, the feature map, $X$ ($X \in \mathbb{R}^{H \times W \times C}$), is passed to convolutional layers which can be simply noted as a non-linear function $F_{\text{deform}}$. Thus, its output is reshaped and regresses the deformation offsets of the sampling locations, $\Theta (\Theta \in \mathbb{R}^{H \times W \times 2})$, as follows:

$$\Theta = F_{\text{deform}}(X),$$

where the convolution layers consist of a convolution layer with $3 \times 3$ kernel size followed by a fully connected layer whose output has the same size as $\Theta$, i.e., $H \times W \times 2$. In particular, $\theta_{i,j}$ ($\theta_{i,j} \in \Theta$) refers to the 2D offset vector how the element $X_{i,j}$ ($X_{i,j} \in X$) in the normal feature maps deform. Hence, the deformable feature maps are calculated as follows:

$$X'_{i,j} = F_{\text{sample}}(X_{[i,j]} + \theta_{i,j}),$$

s.t. $\theta_{i,j} \in \mathbb{R}^{1 \times 2}$, $\theta_{i,j} \in \Theta$,
$$1 \leq i \leq H, 1 \leq j \leq W,$$

where the deformable features at the location $[i,j]$, $X'_{i,j}$, are sampled from the location $[i,j] + \theta_{i,j}$ instead of the location $[i,j]$, with the shape of the offset vector $\theta_{i,j}$ transformed to $\mathbb{R}^{1 \times 2}$ in advance. Note that $\theta_{i,j}$ is often fractional, so a sampling kernel is applied to the feature maps as shown in Fig. 3. Here, $F_{\text{sample}}(X_{[d_x,d_y]})$ is a bilinear interpolation kernel which samples $X$ at location $[d_x,d_y]$, as follows:

$$F_{\text{sample}}(X_{d}) = \sum_{x=1}^{W} \sum_{y=1}^{H} G([x, y], [d_x, d_y])X_{x,y},$$

$$G([x, y], [d_x, d_y]) = \max(0, 1 - |x - d_x|) \max(0, 1 - |y - d_y|),$$

where $F_{\text{sample}}(X_{[d_x,d_y]})$ samples the feature maps by calculating the weighted sum of the features at neighboring locations. $G(a, b)$ computes the weights of features at the location $a$ for sampling the features at the location $b$.

As we know, the standard CNNs perform the convolution operation over the fixed geometric structure, which is not reasonable for handling the object with significant appearance variations. For instance (as shown in Fig. 4), when the athlete runs, his body parts rapidly change the appearance and locations. The normal convolutional feature cannot adapt to such variation well. On the contrary, the deformation convolution roughly estimates how the visual parts of the object will move.
by $F_{deform}$ and then resamples the feature maps by bilinear interpolation $F_{sample}$. It estimates the feature maps after object deformation and thus the approximated feature maps serve as the deformable features, which recovers the deformed object features into normal ones. Since the deformation is simply modeled by offsets of sampling grids, it supports free-form deformation instead of the affine transformation in STN \cite{Jaderberg2015}. Therefore, it is useful for tracking objects with not only the in-plane rotation but also the out-of-plane rotation. Besides, the performance of the standard CNNs is degraded because the training samples collected online are limited and similar. Due to the online estimated offsets, the deformable convolution can adaptively model the unseen deformation in an online manner.

In Fig. 4, we illustrate an example of the effect of the deformable convolutional features in tracking. We densely draw samples from the region of the interest from the frame #15, #21, #26, #35 in the video Bolt2 and compute their confidence scores. This figure demonstrates the locations of the drawn samples and their corresponding confidence scores. As observed, without deformable features, the classifier is uncertain around the target region when the athlete runs with the appearance variations. In frames #15 and #21, the high confidence samples disperse, which indicates the target pose affects the performance of the tracker. In the frame #26, the high confidence area concentrates on the athlete, because the pose has been ‘seen’ before and the standard CNN can well classify it. In the frame #35, the confident samples are gathered in two clusters, which means the tracker is confused by the target object and the neighboring object. In contrast, the high confidence area of the classifier with the deformable convolutional features is more concentrated on the target object, even though the target object exhibits some unseen poses. In frames #15 and #26, the high confidence area is accurately localized on the target. In #21, we observe that one of the legs is occluded, which degrades the classification, but it recovers in the frame #26 and #35. In summary, the deformable convolutional features are robust for modeling the appearance variations of the deformable object.

**B. Gated Fusion**

In practice, the deformable convolution layer may not perform well and its output feature maps will be erroneous, so it requires to compensate with the normal convolutional features. Here we introduce a gating module which can control the fusion of both features, as shown in Fig. 2. The gating mechanism is first proposed in Long-Short Term Memory (LSTM) cells \cite{Goodfellow2016} for regulating the information flow through the network.

In our framework, we introduce a soft-gate which adaptively fuses the deformable convolutional features and normal convolutional features. The motivation of applying gating module is to enhance the features fusion between the deformable convolutional features and the normal convolutional features. As we mention in Sec. III-A when tracking objects with severe appearance variations, the normal convolutional features may fail due to the unseen appearance of the object, while the deformable convolutional features can to some extent adapt to the deformation. However, the deformable convolution also has limitations. As we observe, in ordinary scenarios, the normal convolutional features demonstrate more robust performance in tracking objects with few appearance variations. In addition, in some extreme situations such as significant illumination changes or severe occlusion, the deformation cannot accurately estimate the deformation offsets, so it may result in larger errors than the normal convolutional features. Thus, we not only switch between these features accordingly, but combine them adaptively as well.

The output of gating module is learned by another branch of convolution layers, $F_{gate}$, followed by a sigmoid activation function, $F_σ$. The sigmoid layer constrains the gating output within $[0, 1]$, which serves as a weight that balances which type of features should be dominant. Thus, the output of gating module is computed as:

$$\sigma = F_σ(F_{gate}(X)).$$  

(5)

To better control the fusion, we set the dimension of the output as $\sigma \in \mathbb{R}^{H \times W}$, which is the same as the spatial dimension of the normal convolutional features and the deformable convolutional features. Given the deformable convolutional features $X'$ and the normal convolutional features $X$, the fusion process is computed as:

$$Y = X' \odot \sigma + X \odot (1 - \sigma),$$  

(6)

where $\odot$ indicates the element-wise multiplication and $Y$ refers to the fused features. When object shows appearance variations, the gating module should compute a large $\sigma$, thus the deformable convolutional features will be dominant in the fused features. When the deformable convolution cannot achieve good performance, the gating module will output a small $\sigma$ value.

In Fig. 5, we illustrate two examples of how the gating module works in online tracking. The gating module is offline trained with diversified samples and then performs online
As depicted in Fig. 5, our approach leverages the pretrained VGG-M as the front end to extract features. In the second stage, the extracted feature passes through three parallel branches for the gated feature fusion. In the final stage, the fused feature is transferred to fully connected layers which serve as a non-linear classifier.

Model training. We train our model using positive and negative samples from the training data offline. We prepare the training data following [33]. For offline training, 50 positive and 200 negative samples are collected from every frame, where positive and negative examples have \( \geq 0.7 \) and \( \leq 0.5 \) IoU overlap ratios with ground-truth bounding boxes, respectively. To train the deformable convolution layers and the gating module, we follow three steps: 1) training the network without deformable convolution and gating; 2) training the network with the deformable convolution only; 3) training the network with both modules. Hence, in the step 1) and 2), we actually fine-tune the normal convolutional feature and the deformable convolutional feature, respectively. In the step 3), we train the gating module to make sure the tracker can adaptively switch between both modules. All of them are trained on an end-to-end manner.

Model initialization. With the trained model, we fine-tune the network using the samples as the candidate proposals from the first frame of the input sequence. The front end of the network is frozen and only the gating feature fusion and the fully connected layers will completely rely on the normal features. With the \( \sigma \) value 1, the full flow of the deformable convolution will pass through and none of the normal features will be used.

In summary, our proposed framework mainly consists of two modules: the deformable convolution module and the gating module. The deformable convolution module generates the deformable features to handle the object appearance variations, while the gating module adaptively fuses the deformable convolutional features and normal ones in an online manner.

C. Tracking

As depicted in Fig. 5, our approach leverages the pretrained VGG-M as the front end to extract features. In the second stage, the extracted feature passes through three parallel branches for the gated feature fusion. In the final stage, the fused feature is transferred to fully connected layers which serve as a non-linear classifier.

Model training. We train our model using positive and negative samples from the training data offline. We prepare the training data following [33]. For offline training, 50 positive and 200 negative samples are collected from every frame, where positive and negative examples have \( \geq 0.7 \) and \( \leq 0.5 \) IoU overlap ratios with ground-truth bounding boxes, respectively. To train the deformable convolution layers and the gating module, we follow three steps: 1) training the network without deformable convolution and gating; 2) training the network with the deformable convolution only; 3) training the network with both modules. Hence, in the step 1) and 2), we actually fine-tune the normal convolutional feature and the deformable convolutional feature, respectively. In the step 3), we train the gating module to make sure the tracker can adaptively switch between both modules. All of them are trained on an end-to-end manner.

Model initialization. With the trained model, we fine-tune the network using the samples as the candidate proposals from the first frame of the input sequence. The front end of the network is frozen and only the gating feature fusion and the fully connected layers are updated. To estimate the scale of the object, we also train a bounding box regression using a simple linear regression model to predict the precise target location using \( \text{conv3} \) features of the samples near the target location at the first frame.

Model update. We incrementally update the tracker online. Around the estimated position, we generate multiple samples and assign them with binary labels according to the intersection-over-union ratios with the estimated bounding box. The optimal target state is the drawn sample with the highest positive score. In addition, we adopt the hard negative mining in mini-batch sampling. The hard negative samples are selected from the classification confidence of samples which are evaluated in our network. In particular, we select the negative samples with top confidence scores as the hard negative samples which are used for updating the model.

IV. Experiments

In this section, we introduce the implementation details of our proposed model and analyze the effects of the modules in the network. In specific, we denote our tracker as \( \text{GDT} \) (Gated-fusion Deformable object Tracker) and we compare
GDT with state-of-the-art trackers on the benchmark datasets OTB-2013 [46], OTB-2015 [47] and VOT-2016 [48] for performance evaluation.

A. Implementation Details and Experimental Setup

**Implementation details** In the deformable convolution, the resolution of the deformable convolutional feature is set as $3 \times 3$ the same as the feature maps computed from the front end, while the gating module’s output $\sigma$ is also set as $3 \times 3$. For offline training the network, we apply SGD solver to run for 200K iterations. The learning rates of the front-end convolution layers and feature fusion modules are set as $10^{-4}$ and the fully connected layers as $10^{-3}$. For the evaluation in OTB, the training data is collected from labeled sequences of VOT challenges excluding the sequences from OTB-100. For the evaluation in VOT-2016, similarly we use OTB data for training. At the initialization stage, we train the feature fusion and the fully connected layers for 30 iterations with learning rate 0.0003 except the last layer as 0.001. For online update, we assign the learning rate 0.0005 for the feature fusion module. The momentum and weight decay are always set to 0.9 and 0.0005, respectively. Each mini-batch consists of 32 positives and 96 hard negatives selected out of 1024 negative examples. Our proposed tracker GDT runs on a PC with an i7 3.6GHz CPU and a NVIDIA Geforce 1080Ti GPU with the MatConvNet toolbox [59]. In tracking, the initial training costs around 30 seconds in average and the running time of tracking is 1.33 FPS. The tracker is updated every 10 frames online and the updating costs around 3 seconds.

**Evaluation metrics.** We follow the standard evaluation approaches. In the OTB-2013 and OTB-2015 datasets, we use the one-pass evaluation (OPE) with precision and success plots metrics. The precision metric measures the frame locations rate within a certain threshold distance from groundtruth locations. The threshold distance is set to 20 pixels. The success plot metric is set to measure the overlap ratio between the predicted bounding boxes and the groundtruth. In the VOT-2016 dataset [48], we measure the performance in terms of Expected Average Overlap (EAO), Accuracy Ranks (Ar) and Robustness Ranks (Rr).

B. Ablation Studies

To validate the effectiveness of each module, we first train a complete network with the deformable convolutional features and the gating module and then we remove both modules. The following three models are evaluated: network without both models (Baseline), the network bound with the deformable convolution module only (Baseline+Deform), and the complete network (Baseline+Deform+Gate). We compare the performance of these models on the OTB-2013 dataset. In Fig. 6(a), we evaluate the overall tracking performance by the success rate metric in the entire dataset. In the rest figures in Fig. 6(b-f), we measure the tracking performance for the videos containing deformation, in-plane rotation, out-of-plane rotation, background clutter, and scale variation attributes.

As shown in Fig. 6(a) and (b), the deformable convolutional features slightly boosts the overall success rate from 0.699 to 0.701, but it significantly improves the capability of tracking deformable objects (the success rate from 0.699 to 0.712). It indicates that the deformable convolutional features indeed strengthens the capability of tracking deformable object. However, the overall tracking capability is not obviously improved. The reason is that, as we mentioned in Sec. III-B the deformable convolutional features are not that robust as the normal convolutional features. After incorporating the gating module, the overall performance significantly gets better (from 0.701 to 0.711), while not degrading the capability of tracking deformable objects (staying at 0.712). The result indicates that the gating module supplements the deformable convolutional features and improve the general tracking performance. More importantly, the capability of tracking deformable object is not degraded, which means the gating module can adaptively switch the fusion to the deformable convolutional features when tracking deformable objects. As illustrated in Fig. 6(c-f), the deformable convolutional features do not significantly improve or even degrade the the tracking performance. However, as we observe, the involvement of the gating module plays an important role to boost the performance.
C. Quantitative Evaluation

OTB-2013 Dataset. We compare GDT with 29 trackers from the OTB-2013 benchmark [46] and other 27 state-of-the-art trackers including TCNN [50], EBT [51], DSST [3], KCF [2], MEEM [52], LCT [4], MUSTer [53], HCFT [9], FCNT [14], SRDCF [7], CNN-SVM [31], DeepSRDCF [54], Staple [55], SRDCFdecon [56], CCOT [6], SiamFC [17], MDNet [33], ADNet [57], ECO [5], MCPF [11], and CREST [15], etc. We evaluate all the trackers on 50 video sequences using the one-pass evaluation with distance precision and overlap success metrics.

Fig. 7 shows the quantitative results from compared trackers. For the presentation clarity, we only show the top 10 trackers. The numbers listed in the legend indicate the AUC overlap success and 20 pixel distance precision scores. Overall, our GDT performs favorably against state-of-art trackers in both distance precision and overlap success. GDT achieves the top success rate (0.711) than other methods including ECO and MDNet. For the distance precision, GDT performs better than other trackers except for MDNet that is similar to our baseline. It means that the deformable convolution slightly decreases the robustness of the model.

In addition, Fig. 10 compares the performance of selected six video attributes including deformation, in-plane rotation, out-of-plane rotation, scale variation, low resolution, and illumination variation, using one-pass evaluation. Observed from the results, our tracker GDT generally outperform most of the state-of-the-art trackers. In particular, our tracker handles large appearance variations caused by deformation, in-plane and out-of-plane rotations better than all of the other methods. It indicates that the deformable convolution enhances the capability of modeling a variety of appearance variations. And the gating module blends the normal convolutional features and the deformable convolutional features, which to some extent can model the unseen appearance variations by approximately recovering them to the normal conditions. For videos with scale variation, low resolution and illumination variations, the object’s appearance cannot be well-observed, so it is difficult for the deformable convolution to estimate the deformation offsets. Some scenarios with low resolution and illumination variations are very challenging that the gated feature fusion may not well balance the deformable feature maps and the normal ones.

OTB-2015 Dataset. We compare GDT on the OTB-2015 benchmark [47] with the state-of-the-art trackers. Fig. 8 shows that our proposed tracker GDT performs generally well. Although the ECO tracker achieves the best result in overlap success, our GDT performs the same as ECO in distance precision. As we know, ECO is the state-of-the-art tracker in several public benchmarks, which improves continuous convolutional operators on a multi-level feature maps. Since the OTB-2015 dataset contains more challenging videos with fast motion and low resolution, our tracker fails to match up with ECO in success rate. Besides, our method outperforms all of other methods in both success rate and precision.

VOT-2016 Dataset. We compare our tracker with state-of-the-art trackers on the VOT-2016 benchmark, including ECO [5], Staple [55], MDNet [33], and CCOT [3]. Table I shows the EAO metric of GDT is comparable to state-of-the-art trackers. Our method is only worse than ECO in EAO and ranks the top in the accuracy rank (Ar). Considering the slight loss of robustness brought by the deformable convolution, our GDT’s robustness rank (Rr) is behind ECO and CCOT. Herein, the

| Tracker       | EAO  | Ar  | Rr   |
|---------------|------|-----|------|
| ECO           | 0.374| 1.60| 1.63 |
| CCOT          | 0.331| 1.72| 1.70 |
| Staple        | 0.295| 1.68| 2.65 |
| MDNet         | 0.257| 1.67| 2.08 |
| GDT (our)     | 0.353| 1.48| 1.93 |

Fig. 9 shows the comparison with more state-of-the-art trackers on the VOT2016 dataset. The results are presented specifically in terms of expected average overlap (EAO).
EAO measures the expected no-reset overlap of a tracker run on a short-term sequence. Since it is computed from the VOT reset-based experiment, it does not suffer from the large variance and has a clear relation to the definition of short-term tracking. We focus on analyzing EAO by comparing with 41 other trackers including CCOT, TCNN, Staple, EBT, and so forth. The comparison is illustrated in Fig. 9. Our GDT has the highest expected average overlap among them. Since VOT-2016 consists of many short-term challenging tracking scenarios with severe appearance deformation (e.g. motocross1, gymnastics1), our approach outperforms other methods in these scenarios.

D. Qualitative Evaluation

Fig. 11 visually compare the results of the top-performing trackers: CCOT [3], MDNet [33], ECO [5] and our GDT on 12 challenging sequences. In the scenarios of Diving, Jump, Skater2, and Trans, which are close-up sequences from sports and movies, the objects exhibit severe pose variations. They present the superiority of our method qualitatively compared to the state-of-the-art trackers. For instance, in Diving, CCOT loses track in the early stage and ECO fails to cover the body of the object. MDNet has a slight deviation at the frame #91, while our tracker performs well in the entire sequence. Similarly in Jump, MDNet, COT, and ECO lose track and they cannot well estimate the object size. Since this sequence is very challenging, our tracker cannot estimate the object scale accurately either, but it still manages to track the person. In the sequence Skater2 where two skaters frequently interact, the trackers tend to be distracted by the partner of the tracking target, while GDT always clings to the target. Both CCOT and ECO extract CNN features and learn correlation filters separately, which is not based on an end-to-end architecture, so their features are not robust against deformation. MDNet also cannot track the object with severe deformation due to the lack of the mechanism to handle deformation. In the scenario ClifBar and MotorRolling where the object rotates, ECO and MDNet perform poorly. CCOT also drifts away in MotorRolling. With the aid of the deformable convolution, our tracker is better in estimating the scales of the object and tracking the object with rotations. There are also some examples (Basketball, Girl2, Football, Freeman1) that GDT tracks objects with partial occlusions.

V. Conclusion

We propose a deformable convolution layer to model target appearance variations in the CNN-based tracking-by-detection framework. We aim to capture target appearance variations via deformable convolution and supplement its normal convolution features through the online learned gating module. The gating module controls how the deformable convolutional features and the normal features are fused. Experimental results show that the proposed tracker performs favorably against the state-of-the-art methods.

There are still limitations in our proposed model. Our deformable convolution slightly degrades the robustness of the model, since its deformation estimation may fail in some extreme situations. As the future work, to enhance the robustness of the deformable convolution, we are going to collect more data and adopt data augmentation technique (e.g. image warping) to specifically train the deformable convolution module. Besides, the online learned gating module may not be adequately adaptive in difficult videos. To alleviate this problem, we aim to improve the offline training process by...
Fig. 11. Qualitative evaluation of our GDT, CCOT [14], MDNet [41] and ECO [9] on 12 challenging sequences (from left to right and top to down: Basketball, ClifBar, Girl2, Freeman1, Bird1, Football, MotorRolling, Bolt2, Diving, Jump, Skater2 and Trans) from OTB-2015. Our proposed tracker outperforms the state-of-the-art methods.

leverage the confidence scores of the normal convolution and the deformable convolution.

REFERENCES

[1] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” in CVPR, 2010.
[2] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” TPAMI, 2015.
[3] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” in BMVC, 2014.
[4] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, “Long-term correlation tracking,” in CVPR, 2015.
[5] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, “Eco: efficient convolution operators for tracking,” in CVPR, 2017.
[6] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in ECCV, 2016.
[7] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” in ICCV, 2015.
[8] H. K. Galoogahi, T. Sim, and S. Lucey, “Correlation filters with limited boundaries,” in CVPR, 2015.
[9] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Robust visual tracking via hierarchical convolutional features,” in TPAMI, 2018.
[10] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, “Hedged deep tracking,” in CVPR, 2016.
[11] T. Zhang, C. Xu, and M.-H. Yang, “Multi-task correlation particle filter for robust object tracking,” in CVPR, 2017.
[12] H. Kiani Galoogahi, A. Fagg, and S. Lucey, “Learning background-aware correlation filters for visual tracking,” in CVPR, 2017.
[13] C. Sun, D. Wang, H. Lu, and M.-H. Yang, “Learning spatial-aware regressions for visual tracking,” in CVPR, 2018.
[14] L. Wang, W. Ouyang, X. Wang, and H. Lu, “Visual tracking with fully convolutional networks,” in ICCV, 2015.
[15] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, and M.-H. Yang, “Crest: Convolutional residual learning for visual tracking,” in ICCV, 2017.
[16] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, “End-to-end representation learning for correlation filter based tracking,” in CVPR, 2017.
[17] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, “Fully-convolutional siamese networks for object tracking,” in ECCV. Springer, 2016, pp. 850–865.
[18] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in ECCV, 2016.
[19] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “End-to-end representation learning for correlation filter based tracking,” in CVPR, 2017.
[20] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, “Learning dynamic siamese network for visual object tracking,” in ICCV, 2017.
[21] B. Li, W. Wu, Z. Zhu, and J. Yan, “High performance visual tracking with siamese region proposal network,” in CVPR, 2018.
[22] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, “Learning attentions: Residual attentional siamese network for high performance online visual tracking,” in CVPR, 2018.
[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, “Human-level control through deep reinforcement learning,” Nature, 2015.
[24] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, “Action-decision networks for visual tracking with deep reinforcement learning,” in CVPR, 2017.
[25] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, “Hyperparameter optimization for tracking with continuous deep q-learning,” in CVPR, 2018.

[26] C. Huang, S. Lucey, and D. Ramanan, “Learning policies for adaptive tracking with deep feature cascades,” in ICCV, 2017.

[27] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking learning detection,” TPAMI.

[28] H. Grabner, M. Grabner, and H. Bischof, “Real-time tracking via on-line boosting,” in BMVC, 2006.

[29] B. Babenko, M.-H. Yang, and S. Belongie, “Visual tracking with online multiple instance learning,” in CVPR, 2009.

[30] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[31] S. Hong, T. You, S. Kwak, and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network,” in ICML, 2015.

[32] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[33] S. Hong, T. You, S. Kwak, and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network,” in ICML, 2015.

[34] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[35] S. Hong, T. You, S. Kwak, and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network,” in ICML, 2015.

[36] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[37] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[38] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[39] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[40] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[41] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[42] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[43] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[44] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[45] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[46] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[47] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[48] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[49] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[50] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[51] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[52] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[53] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[54] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[55] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[56] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.

[57] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” in TPAMI.