CREST: Convolutional Residual Learning for Visual Tracking

Yibing Song[1], Chao Ma[2], Lijun Gong[3], Jiawei Zhang[1], Rynson W.H. Lau[1], and Ming-Hsuan Yang[4]

[1]City University of Hong Kong, Hong Kong   [2]The University of Adelaide, Australia
[3]SenseNets Technology Ltd, China   [4]University of California, Merced, U.S.A.

Abstract

Discriminative correlation filters (DCFs) have been shown to perform superiorly in visual tracking. They only need a small set of training samples from the initial frame to generate an appearance model. However, existing DCFs learn the filters separately from feature extraction, and update these filters using a moving average operation with an empirical weight. These DCF trackers hardly benefit from the end-to-end training. In this paper, we propose the CREST algorithm to reformulate DCFs as a one-layer convolutional neural network. Our method integrates feature extraction, response map generation as well as model update into the neural networks for an end-to-end training. To reduce model degradation during online update, we apply residual learning to take appearance changes into account. Extensive experiments on the benchmark datasets demonstrate that our CREST tracker performs favorably against state-of-the-art trackers. [1]

1. Introduction

Visual tracking has various applications ranging from video surveillance, human computer interaction to autonomous driving. The main difficulty is how to utilize the extremely limited training data (usually a bounding box in the first frame) to develop an appearance model robust to a variety of challenges including background clutter, scale variation, motion blur and partial occlusions. Discriminative correlation filters (DCFs) have attracted an increasing attention in the tracking community [4, 8, 30], due to the following two important properties. First, since spatial correlation is often computed in the Fourier domain as an element-wise product, DCFs are suitable for fast tracking. Second, DCFs regress the circularly shifted versions of input features to soft labels, i.e., generated by a Gaussian function ranging from zero to one. In contrast to most existing tracking-by-detection approaches [22, 1, 14, 34]
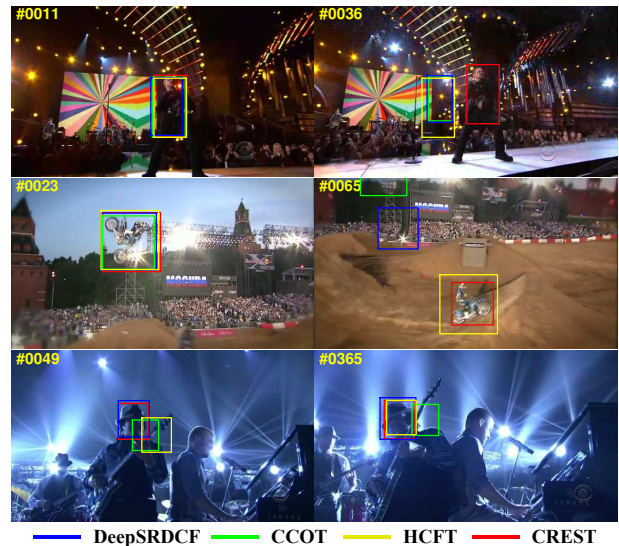
---

[1]More results and code are provided on the authors' webpages.

Figure 1: Convolutional features improve DCFs (Deep-SRDCF [8], CCOT [11], HCFT [30]). We propose the CREST algorithm by formulating DCFs as a shallow convolutional layer with residual learning. It performs favorably against existing DCFs with convolutional features.

that generate sparse response scores over sampled locations, DCFs always generate dense response scores over all searching locations. With the use of deep convolutional features [25], DCFs based tracking algorithms [30, 8, 11] have achieved state-of-the-art performance on recent tracking benchmark datasets [45, 46, 24].

However, existing DCFs based tracking algorithms are limited by two aspects. First, learning DCFs is independent of feature extraction. Although it is straightforward to learn DCFs directly over deep convolutional features as in [30, 8, 11], DCFs trackers benefit little from the end-to-end training. Second, most DCFs trackers use a linear interpolation operation to update the learned filters over time. Such an empirical interpolation weight is unlikely to strike a good balance between model adaptivity and stability. It leads to drifting of the DCFs trackers due to noisy updates. These

limitations raise two questions: (1) whether DCFs with feature representation can be modeled in an end-to-end manner, and (2) whether DCFs can be updated in a more effective way rather than using those empirical operations such as linear interpolation?

To address these two questions, we propose a Convolutional RESidual learning scheme for visual Tracking (CREST). We interpret DCFs as the counterparts of the convolution filters in deep neural networks. In light of this idea, we reformulate DCFs as a one-layer convolutional neural network that directly generates the response map as the spatial correlation between two consecutive frames. With this formulation, feature extraction through pre-trained CNN models (e.g., VGGNet [38]), correlation response map generation, as well as model update are effectively integrated into an end-to-end form. The spatial convolutional operation functions similarly with the dot product between the circulant shifted inputs and the correlation filter. It removes the boundary effect in Fourier transform through directly convolving in the spatial domain. Moreover, the convolutional layer is fully differentiable. It allows updating the convolutional filters using back propagation. Similar to DCFs, the convolutional layer generates dense response scores over all searching locations in a one-pass manner. To properly update our model, we apply residual learning [15] to capture appearance changes by detecting the difference between the output of this convolutional layer and ground truth soft label. This helps alleviate a rapid model degradation caused by noisy updates. Meanwhile, residual learning contributes to the target response robustness for large appearance variations. Ablation studies (Section 5.2) show that the proposed convolutional layer performs well against state-of-the-art DCFs trackers and the residual learning approach further improves the accuracy.

The main contributions of this work are as follows:

- We reformulate the correlation filter as one convolutional layer. It integrates feature extraction, response generation, and model update into the convolutional neural network for end-to-end training.
- We apply residual learning to capture the target appearance changes referring to spatiotemporal frames. This effectively alleviates a rapid model degradation by large appearance changes.
- We extensively validate our method on benchmark datasets with large-scale sequences. We show that our CREST tracker performs favorably against state-of-the-art trackers.

2. Related Work

There are extensive surveys of visual tracking in the literature [47, 37, 39]. In this section, we mainly discuss tracking methods that are based on correlation filters and CNNs.

Tracking by Correlation Filters. Correlation filters for visual tracking have attracted considerable attention due to the computational efficiency in the Fourier domain. Tracking methods based on correlation filters regress all the circular-shifted versions of the input features to a Gaussian function. They do not need multiple samples of target appearance. The MOSSE tracker [4] encodes target appearance through an adaptive correlation filter by optimizing the output sum of squared error. Several extensions have been proposed to considerably improves tracking accuracy. The examples include kernelized correlation filters [17], multiple dimensional features [12, 18], context learning [49], scale estimation [7], re-detection [31], subspace learning [28], short-term and long-term memory [20], reliable collection [27] and spatial regularization [9]. Different from existing correlation filters based frameworks that formulate correlation operation as an element wise multiplication in the Fourier domain, we formulate the correlation filter as a convolution operation in the spatial domain. It is presented by one convolutional layer in CNN. In this sense, we demonstrate that feature extraction, response generation as well as model update can be integrated into one network for end-to-end prediction and optimization.

Tracking by CNNs. Visual representations are important for visual tracking. Existing CNN trackers mainly explore the pre-trained object recognition networks and build upon discriminative or regression models. Discriminative tracking methods propose multiple particles and refine them through online classification. They include stacked denoising autoencoder [44], incremental learning [26], SVM classification [19] and fully connected neural network [33]. These discriminative trackers require auxiliary training data as well as an off-line pre-training. On the other hand, regression based methods typically regress CNN features into soft labels (e.g., a two dimensional Gaussian distribution). They focus on integrating convolutional features with the traditional DCF framework. The examples include hierarchical convolutional features [30], adaptive hedging [36], spatial regularization [8] and continuous convolutional operations [11]. In addition, there are methods based on CNN to select convolutional features [42] and update sequentially [43]. Furthermore, the Siamese networks receive growing attention due to its two stream identical structure. These include tracking by object verification [40], tracking by correlation [3] and tracking by location axis prediction [16]. Besides, there is investigation on the recurrent neural network (RNN) to facilitate tracking as object verification [6]. Different from the existing frameworks, we apply residual learning to capture the difference of the predicted response map between the current frame and the ground-truth (the initial frame). This facilitates to account for appearance changes and effectively reduce model degradation caused by noisy updates.
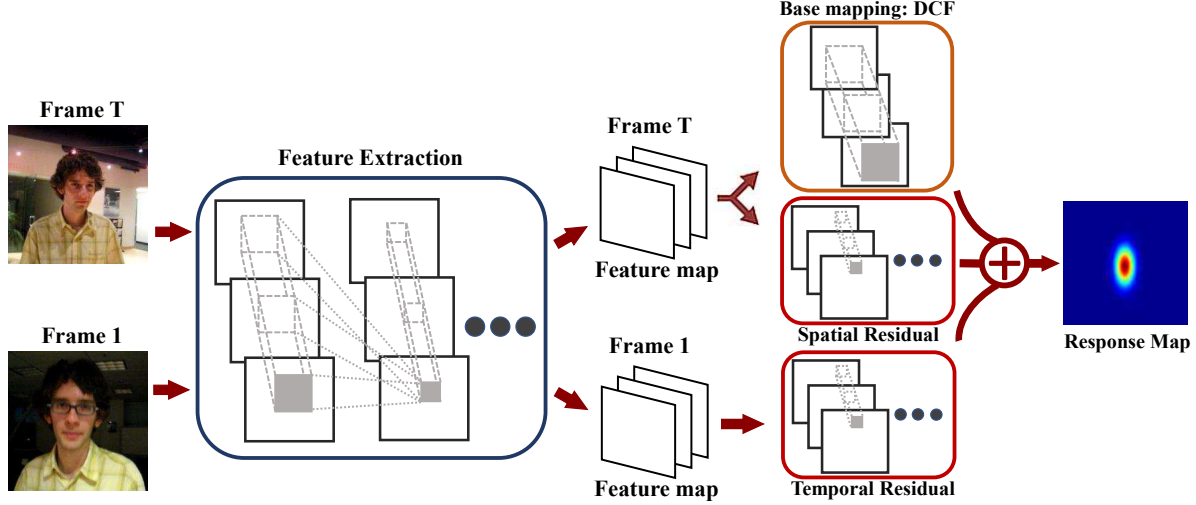
Figure 2: The pipeline of our CREST algorithm. We extract convolutional features from one search patch of the current frame T and the initial frame. These features are transformed into the response map through the base and residual mappings.

3. Convolutional Residual Learning

Our CREST algorithm carries out feature regression through the base and residual layers. The base layer consists of one convolutional layer which is formulated as the traditional DCFs. The difference between the base layer output and ground truth soft label is captured through the residual layers. Figure 2 shows the CREST pipeline. The details are discussed below.

3.1. DCF Reformulation

We revisit the DCF based framework and formulate it as our base layer. The DCFs learn a discriminative classifier and predict the target translation through searching the maximum value in the response map. We denote the input sample by X and denote the corresponding Gaussian function label by Y. A correlation filter W is then learned by solving the following minimization problem:

$$W^{\star} = \arg\min_{W} ||W * X - Y||^2 + \lambda||W||^2, \quad (1)$$

where λ is the regularization parameter. Typically, the convolution operation between the correlation filter W and the input X is formulated into a dot product in the Fourier domain [17, 31, 18].

We reformulate the learning process of DCFs as the loss minimization of the convolutional neural network. The general form of the loss function [21] can be written as:

$$L(W) = \frac{1}{N} \sum_{i}^{|N|} \mathcal{L}_W(X^{(i)}) + \lambda r(W), \quad (2)$$

where N is the number of samples, $\mathcal{L}_W(X^{(i)})$ $(i \in N)$ is the loss of the i-th sample, and $r(W)$ is the weight decay.

We set $N = 1$ and take the L2 norm as $r(W)$. The loss function in Eq. 2 can be written as:

$$L(W) = \mathcal{L}_W(X) + \lambda||W||^2, \quad (3)$$

where $\mathcal{L}_W(X) = ||\mathcal{F}(X) - Y||^2$. It is equivalent to the L2 loss between $\mathcal{F}(X)$ and Y where $\mathcal{F}(X)$ is the network output and Y is the ground truth label. We take $\mathcal{F}(X) = W * X$ as the convolution operation on X, which can be achieved through one convolutional layer. The convolutional filters W is equivalent to the correlation filter and the loss function in Eq. 3 is equivalent to the DCFs objective function. As a result, we formulate the DCFs as one convolutional layer with L2 loss as the objective function. It is named as the base layer in our network. Its filter size is set to cover the target object. The convolutional weights can be effectively calculated using the gradient descent method instead of the closed form solution [18].

3.2. Residual Learning

We formulate DCFs as a base layer represented by one convolutional layer. Ideally, the response map from the base layer output will be identical to the ground truth soft label. In practice, it is unlikely that a single layer network is able to accomplish that. Instead of stacking more layers which may cause the degradation problem [15], we apply the residual learning to effectively capture the difference between the base layer output and the ground truth.

Figure 3 shows the structure of the base and residual layers. We denote $\mathcal{H}(X)$ as the optimal mapping of input X and $\mathcal{F}_B(X)$ as the output from the base layer. Rather than stacking more layers to approximate $\mathcal{H}(X)$, we expect these layers to approximate the residual function: $\mathcal{F}_R(X) = \mathcal{H}(X) - \mathcal{F}_B(X)$. As a result, our expected network out-
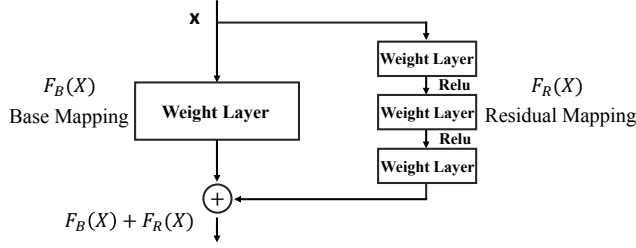
Figure 3: Base and spatial residual layers.

put can be formulated as follows:

$$\begin{aligned} \mathcal{F}(X) &= \mathcal{F}_B(X) + \mathcal{F}_R(X) \\ &= \mathcal{F}_B(X, \{W_B\}) + \mathcal{F}_R(X, \{W_R\}), \quad (4) \end{aligned}$$

where $\mathcal{F}_B(X) = W_B * X$. The mapping $\mathcal{F}_R(X, \{W_R\})$ represents the residual learning and W_R is a general form of convolutional layers with biases and ReLU [32] omitted for simplifying notations. We adopt three layers in the residual learning with small filter size. They are set to capture the residual which is not presented in the base layer output. Finally, input X is regressed through the base and residual mapping to generate the output response map.

In addition, we can also utilize the temporal residual which helps to capture the difference when the spatial residual is not effective. We develop a temporal residual learning network that contains similar structure as the spatial residual learning. The temporal input is extracted from the first frame which contains the initial object appearance. Let X_t denote the input X on frame t. Thus we have

$$\mathcal{F}(X_t) = \mathcal{F}_R(X_t) + \mathcal{F}_{SR}(X_t) + \mathcal{F}_{TR}(X_1), \quad (5)$$

where $\mathcal{F}_{TR}(X_1)$ is the temporal residual from the first frame. The proposed spatiotemporal residual learning process encodes elusive object representation into the response map generation framework and no additional data is needed to train the network.

Figure 4 shows one example of the filter response from the base and residual layers. The feature maps are scaled for visualization purpose. Given an input image, we first crop the search patch centered at the estimated position of the previous frame. The patch is sent into our feature extraction network and then regressed into a response map through the base and residual layers. We observe that the base layer performs similarly to the traditional DCF based trackers to predict response map. When target objects undergo small appearance variations, the difference between the base layer output and the ground truth is minor. The residual layers have little effect on the final response map. However, when target objects undergo large appearance variations, such as background clutter, the response from the base layer is limited and may not differentiate the target and the background.
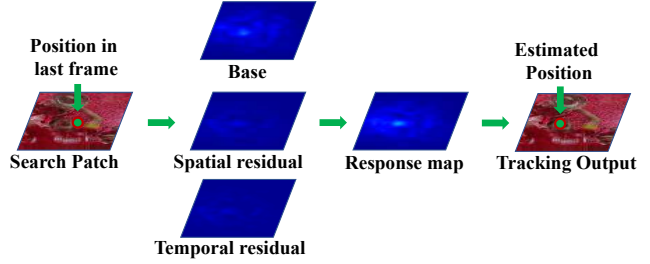

Figure 4: Visualization of the feature maps in the target localization step.

Nevertheless, this limitation is alleviated by the residual layers, which effectively model the difference between the base layer output and the ground truth. It helps to reduce the noisy response values on the final output through the addition of base and residual layers. As a result, target response is more robust to large appearance variations.

4. Tracking via CREST

We illustrate the detailed procedure of CREST for visual tracking. As we do not need offline training we present the tracking process through the aspects of model initialization, online detection, scale estimation and model update.

Model Initialization. Given an input frame with the target location, we extract a training patch which is centered on the target object. This patch is sent into our framework for feature extraction and response mapping. We adopt the VGG network [38] for feature extraction. Meanwhile, all the parameters in the base and residual layers are randomly initialized following zero mean Gaussian distribution. Our base and residual layers are well initialized after a few steps.

Online Detection. The online detection scheme is straightforward. When a new frame comes we extract the search patch from the center location predicted by the previous frame. The search patch has the same size with the training patch and fed into our framework to generate a response map. Once we have the response map, we locate the object by searching for the maximum response value.

Scale Estimation. After we obtain the target center location, we extract search patches in different scales. These patches are then resized into a fixed size of training patches. Thus the candidate object sizes in different scales are all normalized. We send these patches into our network to generate the response map. The width w_t and height h_t of the target object at frame t is updated as:

$$(w_t, h_t) = \beta(w_t^\star, h_t^\star) + (1 - \beta)(w_{t-1}, h_{t-1}), \quad (6)$$

where $w_t^\star$ and $h_t^\star$ are the width and height of the scaled object with maximum response value. The weight factor β enables the smooth update of the target size.

Model Update. We consistently generate training data during online tracking. For each frame, after predicting the target location we can generate the corresponding ground truth response map, and the search patch can be directly adopted as the training patch. The collected training patches and response maps from every T frames are adopted as training pairs which will be fed into our network for online update.

5. Experiments

In this section, we introduce the implementation details and analyze the effect of each component including the base and residual layers. We then compare our CREST tracker with state-of-the-art trackers on the benchmark datasets for performance evaluation.

5.1. Experimental Setups

Implementation Details. We obtain the training patch from the first frame. It is 5 times the maximum value of object width and height. Our feature extraction network is from VGG-16 [38] with only the first two pooling layers retained. We extract the feature maps from the *conv4-3* layer and reduce the feature channels to 64 through PCA dimensionality reduction, which is learned using the first frame image patch. The regression target map is generated using a two-dimensional Gaussian function with a peak value of 1.0. The weight factor β for scale estimation is set to 0.6. Our experiments are performed on a PC with an i7 3.4GHz CPU and a GeForce GTX Titan Black GPU with MatConvNet toolbox [41]. In the training stage, we iteratively apply the adam optimizer [23] with a learning rate of 5e-8 to update the coefficients, until the loss in Eq. 3 is below the given threshold of 0.02. We observe that in practice, our network converges from random initialization in a few hundred iterations. We update the model every 2 frames for only 2 iterations with a learning rate of 2e-9.

Benchmark Datasets. The experiments are conducted on three standard benchmarks: OTB-2013 [45], OTB-2015 [46] and VOT-2016 [24]. The first two datasets contain 50 and 100 sequences, respectively. They are annotated with ground truth bounding boxes and various attributes. In the VOT-2016 dataset, there are 60 challenging videos from a set of more than 300 videos.

We follow the standard evaluation metrics from the benchmarks. For the OTB-2013 and OTB-2015 we use the one-pass evaluation (OPE) with precision and success plots metrics. The precision metric measures the rate of frame locations within a certain threshold distance from those of the ground truth. The threshold distance is set as 20 for all the trackers. The success plot metric measures the overlap ratio between predicted and ground
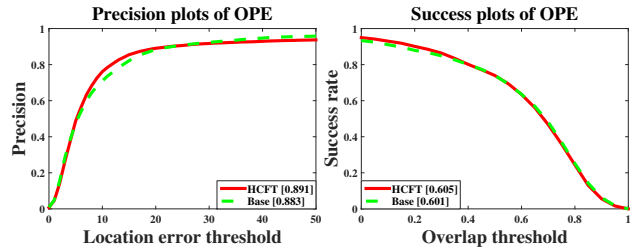

Figure 5: Precision and success plots using one-pass evaluation on the OTB-2013 dataset. The performance of the base layer without scale estimation is similar to that of HCFT [30] on average.
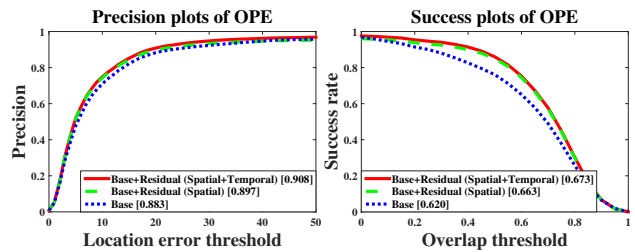

Figure 6: Precision and success plots using one-pass evaluation on the OTB-2013 dataset. The performance of the base layer is improved gradually with the integration of spatiotemporal residual.

truth bounding boxes. For the VOT-2016 dataset, the performance is measured in terms of expected average overlap (EAO), average overlap (AO), accuracy values (Av), accuracy ranks (Ar), robustness values (Rv) and robustness ranks (Rr). The average overlap is similar to the AUC metric in OTB benchmarks.

5.2. Ablation Studies

Our CREST algorithm contains base and residual layers. As analyzed in Section 3.1, the base layer is formulated similar to existing DCF based trackers, and the residual layers refine the response map. In this section, we conduct ablation analysis to compare the performance of the base layer with those of the DCF based trackers. In addition, we also evaluate the base layer and its integration with spatiotemporal residual layers.

We analyze our CREST algorithm in the OTB-2013 dataset. We first compare the base layer performance with that of HCFT [30], which is a traditional DCF based tracker with convolutional features. The objective function of HCFT is shown in Eq. 1, which is the same as ours. As there is no scale estimation in HCFT, we remove this step in our algorithm. Figure 5 shows the quantitative evaluation under AUC and average distance precision scores. We observe that the performance of our base layer is similar to that of HCFT on average. It indicates that our base layer achieves similar performance as the DCF based trackers with con-
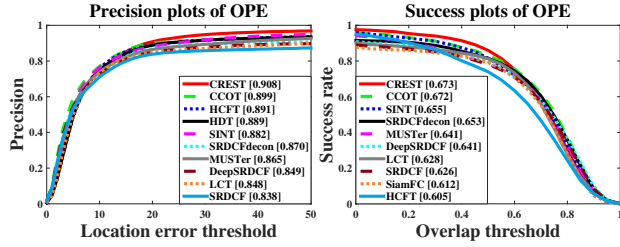
Figure 7: Precision and success plots over all 50 sequences using one-pass evaluation on the OTB-2013 Dataset. The legend contains the area-under-the-curve score and the average distance precision score at 20 pixels for each tracker.
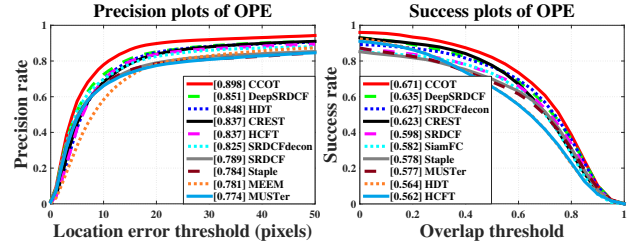

Figure 9: Precision and success plots over all 100 sequences using one-pass evaluation on the OTB-2015 Dataset. The legend contains the area-under-the-curve score and the average distance precision score at 20 pixels for each tracker.

volutional features. The DeepSRDCF [8] and CCOT [11] trackers are different from the traditional DCF based trackers because they add a spatial constraint on the regularization term, which is different from our objective function. We also analyze the effect of residual integration in Figure 6. The AUC and average distance precision scores show that the base layer obtains obvious improvement through the integration of spatial residual learning. Meanwhile, temporal residual contributes little to the overall performance.

5.3. Comparisons with State-of-the-art

We conduct quantitative and qualitative evaluations on the benchmark datasets including OTB-2013 [45], OTB-2015 [46] and VOT-2016 [24]. The details are discussed in the following.

5.3.1 Quantitative Evaluation

OTB-2013 Dataset. We compare our CREST tracker with 29 trackers from the OTB-2013 benchmark [45] and other 21 state-of-the-art trackers including KCF [17], MEEM [48], TGPR [13], DSST [7], RPT [27], MUSTer [20], LCT [31], HCFT [30], FCNT [42], SRDCF [9], CNN-SVM [19], DeepSRDCF [8], DAT [35], Staple [2], SRDCFdecon [10], CCOT [11], GOTURN [16], SINT [40], SiamFC [3], HDT [36] and SCT [5]. The evaluation is conducted on 50 video sequences using one-pass evaluation with distance precision and overlap success metrics.

Figure 7 shows the evaluation results. We only show the top 10 trackers for presentation clarity. The AUC and distance precision scores for each tracker are reported in the figure legend. Among all the trackers, our CREST tracker performs favorably on both the distance precision and overlap success rate. In Figure 7, we exclude the MDNet tracker [33] as it uses tracking videos for training. Overall, the precision and success plots demonstrate that our CREST tracker performs favorably against state-of-the-art trackers.

In Figure 8, we further analyze the tracker performance under different video attributes (e.g., background clutter,

deformation and illumination variation) annotated in the benchmark. We show the one pass evaluation on the AUC score under eight main video attributes. The results indicate that our CREST tracker is effective in handling background clutters and illumination variation. It is mainly because the residual layer can capture the appearance changes for effective model update. When the target appearance undergoes obvious changes or becomes similar to the background, existing DCF based trackers (e.g., HCFT, CCOT) can not accurately locate the target object. They do not perform well compared with SINT that makes sparse prediction through generating candidate bounding boxes and verifying through the Siamese network. This limitation is reduced in our CREST tracker, where the residual from the spatial and temporal domains effectively narrow the gap between the noisy output of the base layer and ground truth label. The dense prediction becomes accurate and the target location can be correctly identified. We have found similar performance in deformation, where LCT achieves a favorable result through integrating the redetection scheme. However, it does not perform well as our CREST tracker with the temporal residual integrated into the framework and optimized as a whole. In motion blur and fast motion sequences, our tracker does not perform well as CCOT. This can be attributed to the convolutional features from multiple layers CCOT has adopted. Their feature representation performs better than ours from single layer output. The feature representation from multiple layers will be considered in our future work.

OTB-2015 Dataset. We also compare our CREST tracker on the OTB-2015 benchmark [46] with the 29 trackers in [45] and the state-of-the-art trackers, including KCF [17], MEEM [48], TGPR [13], DSST [7], MUSTer [20], LCT [31], HCFT [30], SRDCF [9], CNN-SVM [19], Deep-SRDCF [8], Staple [2], SRDCFdecon [10], CCOT [11], HDT [36]. We show the results of one-pass evaluation using the distance precision and overlap success rate in Figure 9. It indicates that our CREST tracker performs better than DCFs trackers HCFT and HDT with convolutional features.
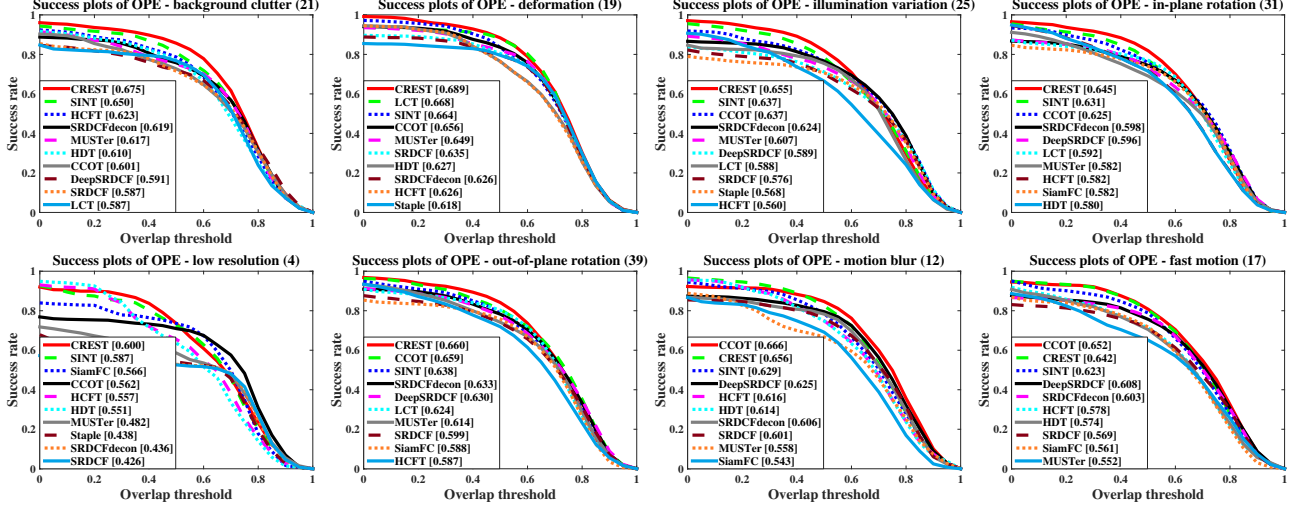
Figure 8: The success plots over eight tracking challenges, including background clutter, deformation, illumination variation, in-plan rotation, low resolution, out-of-plane rotation, motion blur and fast motion.

Table 1: Comparison with the state-of-the-art trackers on the VOT 2016 dataset. The results are presented in terms of expected average overlap (EAO), average overlap (AO), accuracy value (Av), accuracy rank (Ar), robustness value (Rv), and robustness rank (Rr).

	CCOT	Staple	EBT	DSRDCF	MDNet	SiamFC	CREST
EAO	0.331	0.295	0.291	0.276	0.257	0.277	0.283
AO	0.469	0.388	0.370	0.427	0.457	0.421	0.435
Av	0.523	0.538	0.441	0.513	0.533	0.549	0.514
Ar	2.167	2.100	4.383	2.517	1.967	2.465	2.833
Rv	0.850	1.350	0.900	1.167	1.204	1.382	1.083
Rr	2.333	3.933	2.467	3.550	3.250	2.853	2.733

Overall, the CCOT method achieves the best result. Meanwhile, CREST achieves similar performance with Deep-SRDCF in both distance precision and overlap threshold. Since there are more videos involved in this dataset in motion blur and fast motion, our CREST tracker is less effective in handling these sequences as CCOT.

VOT-2016 Dataset. We compare our CREST tracker with state-of-the-art trackers on the VOT 2016 benchmark, including CCOT [11], Staple [2], EBT [50], DeepSRDCF [8], MDNet [33] and SiamFC [3]. As indicated in the VOT 2016 report [24], the strict state-of-the-art bound is 0.251 under EAO metrics. For trackers whose EAO values exceed this bound, they will be considered as state-of-the-art trackers.

Table 1 shows the results from our CREST tracker and the state-of-the-art trackers. Among these methods, CCOT achieves the best results under the EAO metric. Meanwhile, the performance of our CREST tracker is similar to those of Staple and EBT. In addition, these trackers perform better than DeepSRDCF, MDNet and SiamFC trackers. Note that in this dataset, MDNet does not use external tracking videos

for training. According to the analysis of VOT report and the definition of the strict state-of-the-art bound, all these trackers can be regarded as state-of-the-art.

5.3.2 Qualitative Evaluation

Figure 10 shows some results of the top performing trackers: MUSTer [20], SRDCFDecon [10], DeepSRDCF [8], CCOT [11] and our CREST tracker on 12 challenging sequences. The MUSTer tracker does not perform well in all the presented sequences. This is because it adopts empirical operations for feature extraction (e.g., SIFT [29]). Although keypoint matching and long-term processing are involved, handcrafted features with limited performance are not able to differentiate the target and background. In comparison, SRDCFDecon incorporates the tracking-by-detection scheme to jointly optimize the model parameters and sample weights. It performs well on in-plane rotation (*Box*), out-of-view (*Liquor*) and deformation (*Skating2*) because of detection. However, the sparse response leads to the lim-
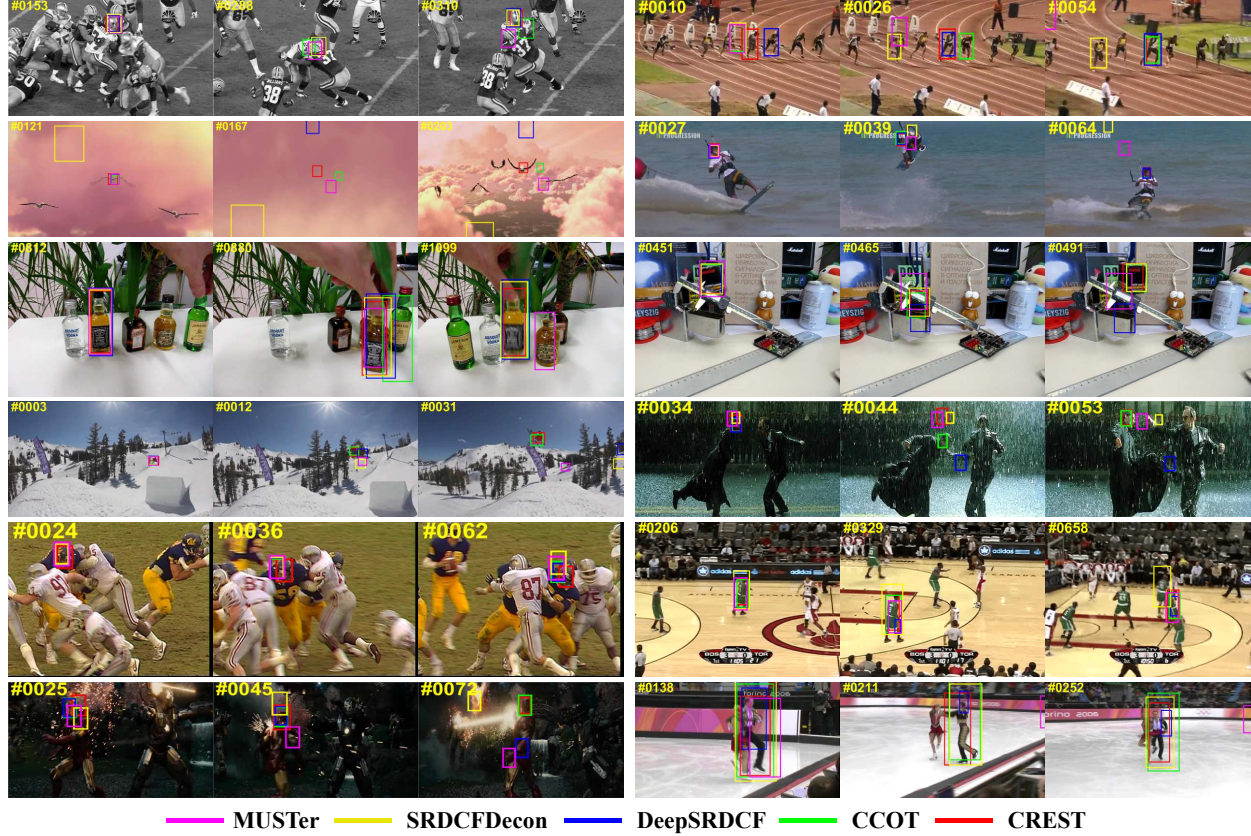
Figure 10: Qualitative evaluation of our CREST tracker, MUSTer [20], SRDCFDecon [10], DeepSRDCF [8], CCOT [11] on twelve challenging sequences (from left to right and top to down: *Football*, *Bolt2*, *Bird1*, *KiteSurf*, *Liquor*, *Box*, *Skiing*, *Matrix*, *Football1*, *Basketball*, *Ironman* and *Skating-2*, respectively). Our CREST tracker performs favorably against the state-of-the-art trackers.

itation on background clutter (*Matrix*), occlusion (*Basketball*, *Bird1*) and fast motion (*Bolt2*). DeepSRDCF improves the performance through combining convolutional features with SRDCF. The dense prediction performs well on the fast motion (*Bolt2*) and occlusion (*Basketball*) scenes. However, the direct integration does not further exploit the model potential and thus limitation occurs on the illumination variation (*Ironman*) and out-of-view (*Bird1*). Instead of involving multiple convolutional features from different layers like CCOT, our CREST algorithm further exploits the model potential through DCF formulation and improves the performance through residual learning. We generate the target prediction in base layer while capturing the elusive residual via residual layers. These residual layers facilitate the learning process since they are jointly optimized with the base layer. As a result, the response map predicted by our network is more accurate for target localization, especially in the presented challenging scenarios. Overall, the visual evaluation indicates that our CREST tracker performs favorably against state-of-the-art trackers.

6. Concluding Remarks

In this paper, we propose CREST to formulate the correlation filter as a one-layer convolutional neural network named base layer. It integrates convolutional feature extraction, correlation response map generation and model update as a whole for end-to-end training and prediction. This convolutional layer is fully differentiable and allows the convolutional filters to be updated via online back propagation. Meanwhile, we exploit the residual learning to take target appearance changes into account. We develop spatiotemporal residual layers to capture the difference between the base layer output and the ground truth Gaussian label. They refine the response map through reducing the noisy values, which alleviate the model degradation limitations. Experiments on the standard benchmarks indicate that our CREST tracker performs favorably against state-of-the-art trackers.

7. Acknowledgements

References

[1] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier. Randomized ensemble tracking. In *ICCV*, 2013.

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

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshop*, 2016.

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[5] J. Choi, H. Jin Chang, J. Jeong, Y. Demiris, and J. Young Choi. Visual tracking using attention-modulated disintegration and integration. In *CVPR*, 2016.

[6] Z. Cui, S. Xiao, J. Feng, and S. Yan. Recurrently target-attending tracking. In *CVPR*, 2016.

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

[8] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshops*, 2015.

[9] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015.

[10] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *CVPR*, 2016.

[11] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[12] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, 2014.

[13] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014.

[14] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. *IEEE TPAMI*, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012.

[18] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE TPAMI*, 2015.

[19] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015.

[20] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *CVPR*, 2015.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[22] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE TPAMI*, 2012.

[23] D. Kingma and J. Ba. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[24] M. Kristan and et al. The visual object tracking vot2016 challenge results. In *ECCV Workshops*, 2016.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[26] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *BMVC*, 2014.

[27] Y. Li, J. Zhu, and S. C. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, 2015.

[28] T. Liu, G. Wang, and Q. Yang. Real-time part-based visual tracking via adaptive correlation filters. In *CVPR*, 2015.

[29] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[30] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[31] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, 2015.

[32] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[33] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[34] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang. Object tracking via dual linear structured svm and explicit feature map. In *CVPR*, 2016.

[35] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *CVPR*, 2015.

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

[37] S. Salti, A. Cavallaro, and L. Di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE TIP*, 2012.

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2014.

[39] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE TPAMI*, 2014.

[40] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[41] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACM Multimedia*, 2015.

[42] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[43] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. In *CVPR*, 2016.

[44] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

[45] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[46] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE PAMI*, 2015.

[47] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 2006.

[48] J. Zhang, S. Ma, and S. Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014.

[49] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang. Fast visual tracking via dense spatio-temporal context learning. In *ECCV*, 2014.

[50] G. Zhu, F. Porikli, and H. Li. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *CVPR*, 2016.