DCFNet: Discriminant Correlation Filters Network for Visual Tracking

Weiming Hu, Qiang Wang, Jin Gao, and Bing Li
(State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190)
{wmhu, qiang.wang, jin.gao, bli}@nlpr.ia.ac.cn

Stephen Maybank
(Department of Computer Science and Information Systems, Birkbeck College, Malet Street, London WC1E 7HX)
sjmaybank@dcs.bbk.ac.uk

Abstract: CNN (convolutional neural network)-based real time trackers usually do not carry out online network update in order to maintain rapid tracking speed. This inevitably influences the adaptability to changes in object appearance. Correlation filter-based trackers can update the model parameters online in real time. In this paper, we present an end-to-end lightweight network architecture, namely the discriminant correlation filter network (DCFNet). A differentiable DCF layer is incorporated into a Siamese network architecture in order to learn the convolutional features and the correlation filter simultaneously. The correlation filter can be efficiently updated online. We introduce a joint scale-position space to the DCFNet, forming a scale DCFNet which carries out the predictions of object scale and position simultaneously. We combine the scale DCFNet with the convolutional-deconvolutional network, learning both the high-level embedding space representations and the low-level fine-grained representations for images. The adaptability of the fine-grained correlation analysis and the generalization capability of the semantic embedding are complementary for visual tracking. The back-propagation is derived in the Fourier frequency domain throughout the entire work, preserving the efficiency of the DCF. Extensive evaluations on the OTB (object tracking benchmark) and VOT (visual object tracking challenge) datasets demonstrate that the proposed trackers have fast speeds, while maintaining tracking accuracy.

Index Terms: Correlation filters, Convolutional neural networks, Visual tracking

1. Introduction

Object tracking is a fundamental problem in computer vision with wide applications, such as human computer interaction and assistant driving systems. Its aim is to estimate the trajectory of an object in consecutive frames [1, 2, 3, 4, 5, 6]. Without knowing the object class a priori, tracking arbitrary objects requires the online learning of their discriminant information. It is a challenging problem [7, 8, 9, 10, 11, 12] because of changes in object appearance, object deformation, variations in scale, variations in pose, severe occlusions, and background clutter, etc.

Maintaining real time processing for visual tracking in complex scenarios is vital in real applications, because any non-real-time trackers cannot be put into actual use. However, real time speed is usually unobtainable for state-of-the-art trackers with online trained classifiers. Correlation filter-based trackers [9, 13, 14] have received a great deal of attention because of their remarkable tracking performance and speed. These trackers model the correlations between the target patch and the full set of approximate surrounding patches in the position space by solving a ridge regression problem efficiently in the Fourier frequency domain. Primitive correlation filter-based trackers were equipped with feature extractors and correlation filters separately. It is proved that good features greatly enhance the tracking performance [15]. Later work concentrated on the integration of multi-layer deep features for correlation filter-based tracking [16, 17, 18, 19, 20, 21]. The object representation has evolved from hand-crafted features (e.g., raw grey level features [13], HOG [9, 22] and color names [23]) to pre-trained multi-layer deep features [16, 17, 18, 21], producing convolutional neural networks (CNN) and correlation filter combined trackers. In the following, we review CNN-based trackers and correlation

filter-based trackers.

1.1. Related work

1.1.1. CNN-based trackers

The good performance of deep convolutional networks on several challenging vision tasks [24, 25, 26, 27, 28] encourages recent work to either utilize existing CNN features [16, 21, 29], or design deep architectures [10, 30, 31, 32, 33] for discriminative visual tracking. CNN-based trackers [32, 33, 34] show great powers for robust tracking. Some work [10, 30, 31] follows the offline training and online fine-tuning paradigm. Although CNN features are highly discriminative, it is computationally expensive to extract the features from each video frame and train or update the tracker using CNN features which are high-dimensional. Online fine-tuning CNNs to account for changes in object appearance severely hampers speed of trackers [10, 30, 31]. Siamese networks are exploited in [32, 33, 34, 35, 36, 37, 38, 39] to formulate visual tracking as a verification problem to build template matching-based trackers without online updating. This achieves high tracking speed. In the offline network pre-training, an embedding space for classification [32, 40] or regression [34] is learnt on external video datasets [41] using a CNN backbone architecture, such as AlexNet [42] and VGGNet [43]. The representations projected in the learnt embedding space have high-level information. They are useful for distinguishing objects with different classes. Such representations have generalization capabilities across different datasets. This makes tracking more robust. During online tracking, these trackers estimate the object position just through a feed forward network pass and do not carry out fine tuning of the network parameters. These CNN-based representations have the advantage of discriminative learning of the classes in the training samples. They are less sensitive to the details for comparing two objects with the same attributes or semantics. These CNN-based trackers have the following limitations:

- The resolution of the representation in the embedding space is often low. Object-specific details useful for fine-grained localization may be lost and the domain shift problem [10] more easily occurs.
- Online network update is usually not carried out in order to maintain rapid tracking speed. This inevitably influences the adaptability to changes in object appearance.

To make the learnt semantic embedding more robust to avoid domain shifts, it is necessary for a Siamese network-based tracker to online update the model and learn the fine-grained image features.

1.1.2. Correlation filter-based trackers

Many developments of correlation filter-based tracking have taken place over an extended period. Bolme et al. [13] introduced correlation filters to visual tracking. The tracker runs at high speed simply using the single channel grey level features. Henriques et al. [9, 22] used circulant matrices to interpret correlation filters and generalize to multi-channel feature cases. Danelljan et al. [23] incorporated color names' features to boost the performance of correlation filter-based tracking. Recently more and more work [16, 17] concentrates on the integration of pre-trained multi-layer deep features into correlation filter-based tracking. Ma et al. [16] learned correlation filters on each hierarchical convolutional layer for tracking. Danelljan et al. [17] built the correlation filter only on the first layer of single-resolution deep feature maps. Valmadre et al. [40] learned tracking-specific deep learning features end to end, and improved tracking accuracy without losing high speed. However, deep learning features with low resolution and wide feature channels were used. Some tracking methods maintain a tracker ensemble [31, 44, 45]. Failures in a single tracker are able to be compensated from other trackers. Hong et al. [46] and Ma et al. [47] added redetection mechanisms to achieve long-term correlation filter-based tracking. Fan and Ling [48] equipped short term tracking based on correlation filtering with long term conservative verifications or redetections. Bertinetto et al. [19] incorporated a color statistics-based model to achieve complementary traits for correlation filter-based tracking. Recent advances of correlation filter-based tracking have achieved great success by using multi-feature channels [16, 23], scale estimation [11, 14, 49], and boundary effect alleviation [20, 50, 51, 52]. Danelljan et al. [14] added one

more scale regression to achieve accurate scale estimation. Danelljan et al. [20] added a spatial regularization term to penalize filter coefficients near template boundaries. Danelljan et al. [14] separately exploited the position filter and one-dimensional scale filter [14], where the simultaneous variations in object scale and position are not well handled. The correlation filter-based trackers have the following limitations:

- The gap between the feature extractors and the correlation filters is usually not effectively and efficiently bridged. The object scale factor is usually not well considered in the appearance modeling process, so as not to take advantage of the correlations between object positions and scales.
- For Siamese network-based correlation filter trackers [32, 33], there is usually no online learning process. Video-specific cues are not exploited and tracking adaptability is lost. The global context constraint in which image patches around the object are used as negative samples is not incorporated in the correlation filter learning process.

It is interesting to construct an online adaptive CNN-combined correlation filter complemented by the high level generic semantic embedding which contains the geometric and structural information about images.

1.2. Our work

To address the above issues in CNN-based trackers and correlation filter-based trackers, we propose three end-to-end network architectures to automatically learn the features suitable for correlation filter-based object tracking in real time.

Different from the correlation filter methods which employ existing features, we develop a discriminative correlation filter network (DCFNet) to automatically learn the features for effectively fitting correlation filter- based tracking in an end-to-end way [53]. This is achieved by incorporating a differentiable correlation filter layer into a Siamese network. The network is trained through error back-propagation. The architecture of the proposed network contains a few convolutional layers which encode the prior tracking knowledge in the offline training process and constitute a feature extractor. Behind these convolutional layers is the correlation filter layer which efficiently completes the online learning and tracking by defining the network output as the probability heatmap of object location. To reduce the computational cost, we make the convolutional layers lightweight. In contrast with [40], which introduces a differentiable correlation filter layer into a Siamese architecture, our work carries out the derivation of the correlation filter layer in the Fourier frequency domain, making tracking more efficient. Our Gaussian expected response-based regression loss is more suitable for tracking than the element-wise one-hot logistic loss in [40].

We extend the DCFNet to a scale DCFNet [53] which learns the convolutional feature representations and carries out adaptive correlation tracking, simultaneously, in the joint scale-position space. The correlation filter layer works on convolutional features extracted from the target template and the search area in the joint scale- position space. A joint correlation analysis is carried out to estimate the object position and size simultaneously. The weights of the correlation filter layer are learnt and updated online in the same joint space to make the tracker adaptive to continuous changes in object appearance. In contrast with [40], our work considers a correlation filter layer which explicitly operates in the scale space in addition to the position space. The differentiable correlation filter layer propagates the gradient from both the position and scale estimation errors.

We add multi-resolution representations to the scale DCFNet, benefitting from both the high-level semantics and fine-grained details, and develop a convolutional-deconvolutional DCFNet for fast, robust, and adaptive visual tracking. The high-level representations of the image from semantic embedding space ensure that the correlation filter has generalization capabilities for visual tracking, because the reconstruction constraint from the deconvolutional network additionally regularizes the semantic embedding. This reconstruction constraint effectively relieves the tracking shifts, and ensures that the learnt semantic embedding keeps the structural and geometric information in the images. The low-level representations with high-resolution in the correlation filter make fine-grained localization of the object feasible. A

global context constraint from negative samples is incorporated for appearance modeling to boost the discriminative power of the tracker. The filter, which serves as a differentiable correlation filter layer, is efficiently updated online for adaptive tracking. A multiple task learning method is used to carry out the image reconstruction and the correlation analysis simultaneously. This causes the tracking robustness and the model adaptability.

Extensive experimental evaluations on four benchmarks, OTB2013 [1], OTB2015 [2], VOT2015 [3], and VOT2017 [54], demonstrate the state-of-the-art performance of the proposed trackers. In particular, it is shown that our trackers run at very high speed, while still achieving competitive tracking accuracy with several state-of-the-art slow trackers.

The rest of the paper is organized as follows: Section 2 introduces the preliminaries of discriminant correlation filters. Section 3 presents our DCFnet for visual tracking. Section 4 describes our scale DCFNet for visual tracking. Section 5 proposes our convolutional-deconvolutional DCFNet for visual tracking. Section 6 reports the experimental results. Section 7 concludes the paper.

2. Discriminant Correlation Filters

In the standard discriminant correlation filters (DCF), a discriminative regression is trained on the features of the target patch and the ideal response which is a unimodal 2D Gaussian function. Let $\varphi^l(\mathbf{x})$ be the feature vector of a target image patch $\mathbf{x}$ on channel l. Let D be the number of the channels. The full set of features for $\mathbf{x}$ is represented by $\{\varphi^l(\mathbf{x})\}_{l=1}^D$. Learning a correlation filter using samples extracted densely around the object is carried out by modeling the circular shifts of the vectorized target image patch $\mathbf{x}$. This modeling is achieved by circular right shifts of elements of feature vector $\varphi^l(\mathbf{x})$, forming a set of feature vectors which are merged into a feature matrix $\mathbf{\Phi}^l(\mathbf{x})$. Each row in the matrix $\mathbf{\Phi}^l(\mathbf{x})$ contains the channel l's features extracted from certain circulant shifts of $\mathbf{x}$. The ideal response $\mathbf{y}$ is defined for pixel (u,v) as follows:

$$y_{u,v} = e^{-\frac{\left(u-\left\lfloor \frac{\Gamma}{2} \right\rfloor\right)^2+\left(v-\left\lfloor \frac{\Gamma}{2} \right\rfloor\right)^2}{\sigma^2}}, \tag{1}$$

where Γ is the edge size of $\mathbf{y}$, $(\lfloor \Gamma/2 \rfloor, \lfloor \Gamma/2 \rfloor)$ is the center of the response map, and σ denotes the response bandwidth. Let $\mathbf{w}^l$ represent the channel l of filter $\mathcal{W}$. The learnt correlation filter $\mathcal{W}$ is obtained by minimizing the output ridge regression loss:

$$\begin{aligned} &\min_{\mathcal{W}} \left\| \sum_{l=1}^{D} \mathbf{\Phi}^l(\mathbf{x})\mathbf{w}^l - \mathbf{y} \right\|_2^2 + \lambda \sum_{l=1}^{D} \left\| \mathbf{w}^l \right\|_2^2 \\ &= \min_{\mathcal{W}} \left\| \sum_{l=1}^{D} \mathrm{CirCorre}(\mathbf{w}^l, \varphi^l(\mathbf{x})) - \mathbf{y} \right\|_2^2 + \lambda \sum_{l=1}^{D} \left\| \mathbf{w}^l \right\|_2^2, \end{aligned} \tag{2}$$

where CirCorre(,) is the circular correlation operation and $\lambda \geq 0$ is a regularization coefficient. The circulant structure of the matrix $\mathbf{\Phi}^l(\mathbf{x})$ ensures that $\mathbf{\Phi}^l(\mathbf{x})\mathbf{w}^l$ is equal to the result of the circular correlation operation on vectors $\varphi^l(\mathbf{x})$ and $\mathbf{w}^l$. In the Fourier domain the circular correlation operation is converted to the Hadamard product, which can be computed very quickly. The ridge regression problem thus has a very efficient solution in the Fourier domain. Let the hat "^" denote the discrete Fourier transform which converts a vector of real numbers to a vector of complex numbers. Let "*" denote the complex conjugate of a complex number**.** Let "$\odot$" denote the Hadamard product. The solution of (2) is obtained by [14]:

$$\hat{\mathbf{w}}^l = \frac{\hat{\varphi}^l(\mathbf{x}) \odot \hat{\mathbf{y}}^*}{\sum_{k=1}^{D} \hat{\varphi}^k(\mathbf{x}) \odot (\hat{\varphi}^k(\mathbf{x}))^* + \lambda \mathbf{e}}, \tag{3}$$

where $\mathbf{e}$ is a vector whose components are all "1", and the division of two vectors yields a vector whose components are the quotients of the corresponding components in these two vectors. It is clear that the DCF is very efficient.

Given the feature vectors $\{\varphi^l(\mathbf{z})\}_{l=1}^D$ of an image patch $\mathbf{z}$ in a new frame, its correlation response map $g(\mathbf{z})$ for w is computed by [14]:

$$g(\mathbf{z}) = \mathcal{F}^{-1}\left(\sum_{l=1}^{D} \hat{\mathbf{w}}^{l*} \odot \hat{\varphi}^l(\mathbf{z}) \right), \tag{4}$$

where $\mathcal{F}^{-1}$ is the discrete Fourier inverse transform. The tracking in a new frame is carried out to search for the image patch with the maximum correlation response [14].

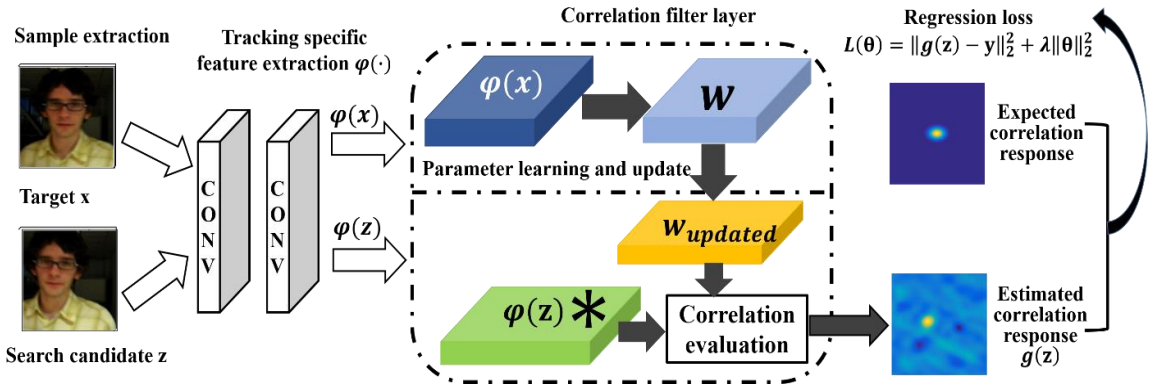


Fig. 1. The overall DCFNet architecture.

3. Discriminant Correlation Filter Network for Tracking

We propose a new discriminant correlation filter network (DCFNet) which consists of two convolutional layers for feature learning and a correlation filter layer for visual tracking, as shown in Fig. 1. Namely, the network is realized by cascading a feature extractor with a DCF module to obtain the response of the object location. The regression loss of the correlation response is back-propagated through the network to adjust the parameters for both feature extraction and object appearance modeling. Different from traditional DCF-based trackers which only tune the hyper parameters heuristically, our DCFNet tunes the DCF hyper parameters and the feature extraction parameters simultaneously. In the offline training stage, the DCFNet is trained from scratch and end-to-end using a video object detection dataset [41]. This enhances the representation power of the feature extraction module. In the online tracking stage, while the convolutional layers are frozen in order to save the processing time, avoid overfitting and reduce tracking drift, the correlation filter layer is updated continuously to tackle the variations in object appearance. The object size and position are simultaneously estimated according to the maximum of the correlation response. The key components of our DCFNet-based model are the derivation of the back propagation and the online model update.

3.1. DCFNet Derivation: back-propagation

The DCFNet consists of two branches: the filter learning branch and the tracking branch. The filter learning branch exploits target exemplars to learn the parameters in the correlation filter layer. The tracking branch works on the candidate search samples and calculates their correlation responses in the correlation filter layer. The network is trained by minimizing the differences between the real response and the expected 2D Gaussian-shaped response. In the training stage, each input to the DCFNet is a pair of images: $\mathbf{z} \leftrightarrow \mathbf{x}$. Let $\boldsymbol{\theta}$ be the current parameters of the CNN. We extract the features $\{\hat{\varphi}_{\boldsymbol{\theta}}^l(\mathbf{x})\}_{l=1}^D$ of the target image patch $\mathbf{x}$ from the CNN with parameters $\boldsymbol{\theta}$. Substitution of $\{\hat{\varphi}_{\boldsymbol{\theta}}^l(\mathbf{x})\}_{l=1}^D$ into (3) yields the filter $\{\mathbf{w}^l\}_{l=1}^D$. The features $\{\hat{\varphi}_{\boldsymbol{\theta}}^l(\mathbf{z})\}_{l=1}^D$ of the search patch $\mathbf{z}$ are extracted from the CNN with $\boldsymbol{\theta}$. The correlation response map $g_{\boldsymbol{\theta}}(\mathbf{z})$ of $\mathbf{z}$ is computed by substituting $\hat{\varphi}_{\boldsymbol{\theta}}^l(\mathbf{z})$ into (4). The objective for optimizing $\boldsymbol{\theta}$ is formulated as:

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \left(\| g_{\boldsymbol{\theta}}(\mathbf{z}) - \mathbf{y} \|_2^2 + \gamma \| \boldsymbol{\theta} \|_2^2 \right). \quad (5)$$

An explicit regularization $\|\boldsymbol{\theta}\|_2^2$ to the network parameters is incorporated for better convergence. We use the weight decay method [55] in the conventional parameter optimization to carry out this regularization. To restrict the magnitude of feature map values and increase the stability in the training process, we add a local response normalization (LRN) [42] at the end of the convolutional layers.

We derive the backward formulas for optimizing the CNN parameters $\boldsymbol{\theta}$ in the frequency domain [56] in Appendix A. Since g is a real-valued vector ($g = g^*$), the discrete Fourier transform and inverse discrete Fourier transform of the derivatives with respect to g have the following relations:

$$\begin{cases} \hat{g} = \mathcal{F}(g), \\ \dfrac{\partial L}{\partial \hat{g}^*} = \mathcal{F}\left(\dfrac{\partial L}{\partial g}\right), \\ \dfrac{\partial L}{\partial g} = \mathcal{F}^{-1}\left(\dfrac{\partial L}{\partial \hat{g}^*}\right). \end{cases} \tag{6}$$

Since the operations in the forward pass process only contain the element-based Hadamard product and division, we calculate the derivative in (6) per pixel (u, v) in the frequency domain:

$$\frac{\partial L}{\partial \hat{g}^*_{uv}} = \left(\mathcal{F}\left(\frac{\partial L}{\partial g}\right)\right)_{uv}. \tag{7}$$

For the back-propagation of the tracking branch, the partial differential $\partial L / \partial(\varphi^l(\mathbf{z}))$ for $\mathbf{z}$ is required to compute. According to (4),

$$\frac{\partial \hat{g}^*_{uv}(\mathbf{z})}{\partial(\hat{\varphi}^l_{uv}(\mathbf{z}))^*} = \hat{w}^l_{uv}. \tag{8}$$

Then,

$$\frac{\partial L}{\partial(\hat{\varphi}^l_{uv}(\mathbf{z}))^*} = \frac{\partial L}{\partial \hat{g}^*_{uv}} \frac{\partial \hat{g}^*_{uv}(\mathbf{z})}{\partial(\hat{\varphi}^l_{uv}(\mathbf{z}))^*} = \frac{\partial L}{\partial \hat{g}^*_{uv}} \hat{w}^l_{uv}. \tag{9}$$

It holds that

$$\frac{\partial L}{\partial \varphi^l(\mathbf{z})} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial(\hat{\varphi}^l(\mathbf{z}))^*}\right). \tag{10}$$

For the back-propagation of the branch of learning the correlation filter, the partial differential $\partial L / \partial(\varphi^l(\mathbf{x}))$ for $\mathbf{x}$ is required to compute. We compute $\partial L / \partial \hat{\varphi}^l_{uv}(\mathbf{x})$ and $\partial L / \partial(\hat{\varphi}^l_{uv}(\mathbf{x}))^*$ independently:

$$\frac{\partial L}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = \frac{\partial L}{\partial \hat{g}^*_{uv}} \frac{\hat{\mathbf{y}}^*_{uv}(\hat{\varphi}^l_{uv}(\mathbf{z}))^* - (\hat{\varphi}^l_{uv}(\mathbf{x}))^*(\hat{\varphi}^l_{uv}(\mathbf{z}))^* \hat{\mathbf{w}}^l_{uv}}{\sum_{k=1}^{D} \hat{\varphi}^k_{uv}(\mathbf{x})(\hat{\varphi}^k_{uv}(\mathbf{x}))^* + \lambda}, \tag{11}$$

$$\frac{\partial L}{\partial(\hat{\varphi}^l_{uv}(\mathbf{x}))^*} = \frac{\partial L}{\partial \hat{g}^*_{uv}} \frac{-\hat{\varphi}^l_{uv}(\mathbf{x})(\hat{\varphi}^l_{uv}(\mathbf{z}))^* \hat{\mathbf{w}}^l_{uv}}{\sum_{k=1}^{D} \hat{\varphi}^k_{uv}(\mathbf{x})(\hat{\varphi}^k_{uv}(\mathbf{x}))^* + \lambda}. \tag{12}$$

Then, $\partial L / \partial \hat{\varphi}^l_{uv}(\mathbf{x})$ and $\partial L / \partial(\hat{\varphi}^l_{uv}(\mathbf{x}))^*$ are combined to compute $\partial L / \partial(\varphi^l(\mathbf{x}))$:

$$\frac{\partial L}{\partial \varphi^l(\mathbf{x})} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial(\hat{\varphi}^l(\mathbf{x}))^*} + \left(\frac{\partial L}{\partial \hat{\varphi}^l(\mathbf{x})}\right)^*\right). \tag{13}$$

Once the error is propagated backwards to the real-value feature maps, the rest of the back-propagation is conducted as the traditional CNN optimization. Since all the operations of the back-propagation in the correlation filter layer are the Hadamard operations in the Fourier frequency domain, the efficiency property of the correlation filter is retained. The offline training can be applied on large-scale datasets. After the offline training has been completed, a feature extractor is obtained for online DCF tracking.

3.2. Online model update

In contrast with the fixed similarity metric in conventional Siamese networks, we update the filter $\mathcal{W}$ over time during the online tracking. As in [23], at frame T the optimization problem in (2) is formulated in an incremental mode:

$$\varepsilon = \sum_{t=1}^{T} \beta_t \left(\left\| \sum_{l=1}^{D} \mathrm{CirCorre}(\mathbf{w}^l, \varphi^l(\mathbf{x}_t)) - \mathbf{y} \right\|_2^2 + \lambda \left\| \sum_{l=1}^{D} \mathbf{w}^l \right\|_2^2 \right), \tag{14}$$

where t indexes a frame and parameter $\beta_t \geq 0$ corresponds to the impact of sample $\mathbf{x}_t$. The closed-form solution in (3) is extended to time series:

$$\hat{\mathbf{w}}^l_T = \frac{\sum_{t=1}^{T} \beta_t \hat{\mathbf{y}}^* \odot \hat{\varphi}^l(\mathbf{x}_t)}{\sum_{t=1}^{T} \beta_t \left(\sum_{k=1}^{D} \hat{\varphi}^k(\mathbf{x}_t) \odot (\hat{\varphi}^k(\mathbf{x}_t))^* + \lambda \mathbf{e} \right)}. \tag{15}$$

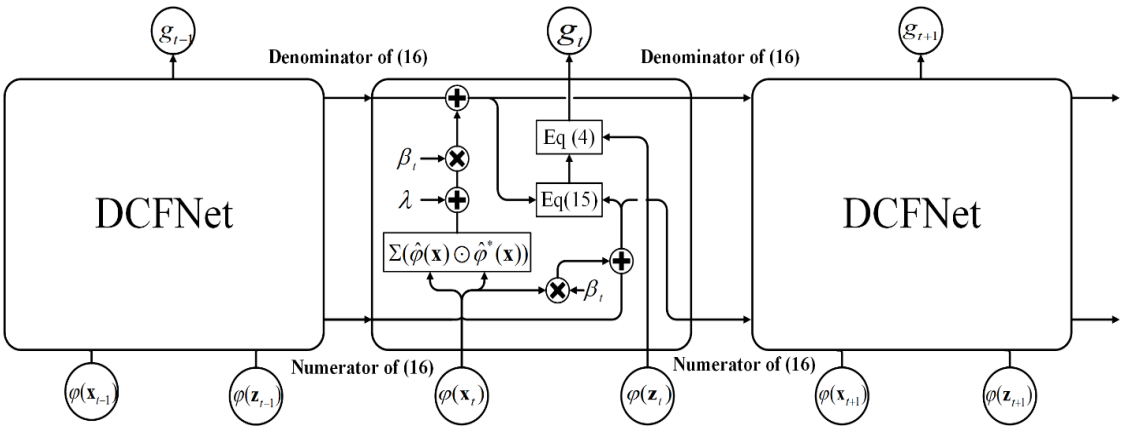


Fig. 2. The online tracking process of DCFNet: The numerator (output of the horizontal pipeline at the bottom) and denominator (output of the horizontal pipeline at the top) in (16) are recurrently forward-propagated and updated.

The filter can be updated in the following incremental way:

$$\hat{\mathbf{w}}_T^l = \frac{\beta_T \hat{\mathbf{y}}^* \odot \hat{\varphi}^l(\mathbf{x}_T) + \sum_{t=1}^{T-1} \beta_t \hat{\mathbf{y}}^* \odot \hat{\varphi}^l(\mathbf{x}_t)}{\beta_T \left(\sum_{k=1}^{D} \hat{\varphi}^k(\mathbf{x}_t) \odot (\hat{\varphi}^k(\mathbf{x}_t))^* + \lambda \mathbf{e} \right) + \sum_{t=1}^{T-1} \beta_t \left(\sum_{k=1}^{D} \hat{\varphi}^k(\mathbf{x}_t) \odot (\hat{\varphi}^k(\mathbf{x}_t))^* + \lambda \mathbf{e} \right)}. \quad (16)$$

This incremental update ensures that it is not necessary to maintain a large sample set. Only a small amount of memory is needed. The DCFNet in the online tracking process can be regarded as a recurrent neural network (RNN) as shown in Fig. 2.

4. Scale DCFNet for Tracking

We introduce joint scales and positions into the DCFNet, forming a scale DCFNet shown in Fig. 3. The scale DCFNet is lightweight with only two convolutional layers and one correlation filter layer. It is end-to-end trainable in the joint scale and position space of the object state, which permits task-driven feature extraction and adaptive appearance modeling in the correlation filter-based tracking framework. The main components of the scale DCFNet-based tracker include object size and position estimation according to the maximum of the joint scale-position correlation response, learning of the tracking task-driven features based on the regression loss of the joint scale-position correlation response, and the update of the correlation filter layer.

4.1. Joint scale position estimation

In order to enhance the discriminative power of the scale position correlation network for visual tracking, our correlation filter layer models the object and contextual appearance correlations in the joint scale position space instead of the pure position space. Thus, the simultaneous variations in object size and position are learnt by the scale DCFNet and accurately estimated in the tracking process.

The set of the parameters of the correlation filter layer contains one correlation filter $\mathbf{w}^l$ per feature channel: $\mathcal{W} = \{\mathbf{w}^l\}_{l=1}^D$. Let S be the number of scales. Let $\mathbf{x}_s$ be the target sample in the s-th scale space. Let $\mathbf{y}_s$ be the desired correlation output in the s-th scale space. The set $\{\mathbf{y}_s\}_{s=1}^S$ is constructed as S 2D Gaussian functions with their peaks at the object's center position (see the expected correlation response in Fig. 3). We extend the single scale ridge regression loss to the multi-scale ridge regression loss by linear extension, which has a closed solution in the Fourier frequency domain:

$$\begin{cases} \mathbf{w} = \arg\min_{\mathbf{w}} \sum_{s=1}^{S} \left\| \sum_{l=1}^{D} \mathrm{CirCorre}\left(\mathbf{w}^l, \varphi^l(\mathbf{x}_s)\right) - \mathbf{y}_s \right\|_2^2 + \lambda \sum_{l=1}^{D} \left\| \mathbf{w}^l \right\|_2^2, \\ \hat{\mathbf{w}}^l = \dfrac{\sum_{s=1}^{S} \hat{\mathbf{y}}_s^* \odot \hat{\varphi}^l(\mathbf{x}_s \mid \boldsymbol{\theta})}{\sum_{s=1}^{S} \sum_{k=1}^{D} \hat{\varphi}^k(\mathbf{x}_s \mid \boldsymbol{\theta}) \odot \left(\hat{\varphi}^k(\mathbf{x}_s \mid \boldsymbol{\theta})\right)^* + \lambda \mathbf{e}}, \end{cases} \quad (17)$$

where $\hat{\mathbf{y}}_s$ denotes the discrete Fourier transform for $\mathbf{y}_s$: $\hat{\mathbf{y}}_s = \mathcal{F}(\mathbf{y}_s)$, and $\hat{\mathbf{y}}_s^*$ is the complex conjugate of $\hat{\mathbf{y}}_s$.

Centered at the object position in the previous frame, we extract the candidate samples by cropping the object regions with different scales from the current frame. The samples are then resized to a fixed size with $M \times N$ pixels. Let $\mathbf{z}_s$ be the candidate sample in the s-th scale space. The set of the candidate samples is denoted as $\{\mathbf{z}_s\}_{s=1}^S$. The convolutional layers for feature extraction work on $\mathbf{z}_s$ and a set of D feature representations $\{\varphi^l(\mathbf{z}_s)\}_{l=1}^D$ are obtained. Then, the correlation response for sample $\mathbf{z}_s$, output from the correlation filter layer, is obtained by:

$$\mathrm{g}(\mathbf{z}_s) = \sum_{l=1}^{D} \mathrm{CirCorre}(\mathbf{w}^l, \varphi^l(\mathbf{z}_s)) = \mathcal{F}^{-1}\left(\sum_{l=1}^{D} \hat{\mathbf{w}}^{l*} \odot \hat{\varphi}^l(\mathbf{z}^s) \right). \quad (18)$$

The object size and position are estimated based on the maximum of the joint scale-position correlation response.

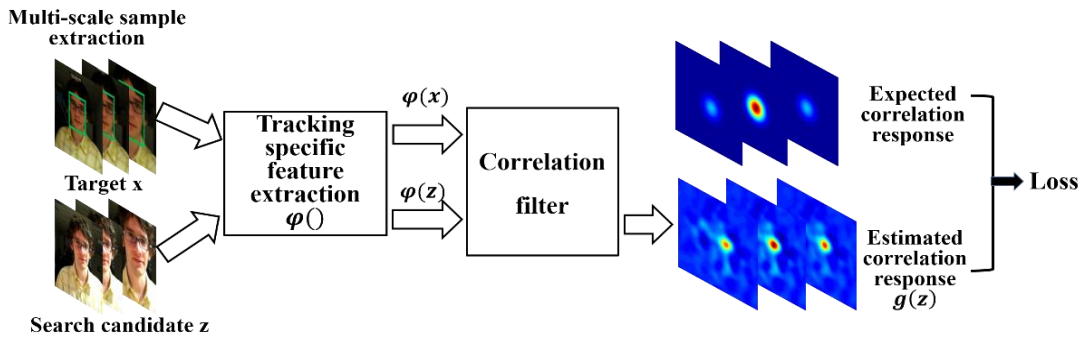


Fig. 3. The architecture of the scale DCFNet.

The parameter set $\mathcal{W}$ for the correlation filter is learnt and updated online in the joint scale-position space to adapt to variations in object appearance. The correlation filter layer is learnt using a set of the training samples in the joint scale-position space. The training samples are extracted from the tracking results in the previous frames and are resized to the same spatial size *M*×*N*. Let $\mathbf{x}_{s,t}$ be the training sample of the *s*-th scale from the *t*-th frame. The correlation responses of the training samples from the *t*-th frame estimated by the correlation filter layer are

$$\left\{ g(\mathbf{x}_{s,t}) = \sum_{l=1}^{D} \mathrm{CirCorre}(\mathbf{w}^l, \varphi^l(\mathbf{x}_{s,t})) \right\}_{s=1}^{S}. \quad (19)$$

By extending the learning of the correlation filter in **the** joint scale-position space to a temporal incremental form, the parameter set $\mathcal{W}$ of the correlation filter at frame *T* is optimized by minimizing the following ridge regression loss:

$$\varepsilon = \sum_{t=1}^{T} \beta_t \left(\sum_{s=1}^{S} \left\| g(\mathbf{x}_{s,t}) - \mathbf{y}_s \right\|_2^2 + \lambda \sum_{l=1}^{D} \left\| \mathbf{w}^l \right\|_2^2 \right), \quad (20)$$

where $\beta_t \geq 0$ is the impact of the training samples from the *t*-th frame and the constant $\lambda \geq 0$ controls the relative weight of the regularization term.

We transform the ridge regression problem in (20) to the Fourier domain using Parseval's formula. As the objective function is real-valued, positive, and convex, the global optimum is obtained by setting the partial derivative equal to zero. The solution is:

$$\hat{\mathbf{w}}^l = \frac{\sum_{t=1}^{T} \beta_t \left(\sum_{s=1}^{S} \hat{\mathbf{y}}_s^* \odot \hat{\varphi}^l \left(\mathbf{x}_{s,t} \right) \right)}{\sum_{t=1}^{T} \beta_t \left(\sum_{s=1}^{S} \sum_{k=1}^{D} \hat{\varphi}^k \left(\mathbf{x}_{s,t} \right) \odot \left(\hat{\varphi}^k \left(\mathbf{x}_{s,t} \right) \right)^* + \lambda \mathbf{e} \right)}. \quad (21)$$

Since this correlation filter layer is learnt in the Fourier frequency domain using several discrete Fourier transforms and element-wise multiplications, the computation in this layer is quite efficient. In particular, the provided closed-form solution avoids an expensive iterative optimization process.

4.2. Task-driven feature learning

The learnable parameters in the correlation filter and the CNN **are co-adapted** and **cooperated** for providing an expected correlation response. The regression loss of the correlation response in the joint scale-position space is back-propagated through the whole network to automatically learn the tracking task-driven features.

The scale position correlation network consists of the filter learning branch and the tracking branch in the joint scale-position space. The network is trained by minimizing the differences between the real response and the *S* expected 2D Gaussian-shaped responses $\{\mathbf{y}_s\}_{s=1}^{S}$. Let each training pair be represented by $\left(\{\mathbf{z}_s\}_{s=1}^{S} \leftrightarrow \{\mathbf{x}_s\}_{s=1}^{S} \right)$. The offline training problem is formulated as:

$$L(\boldsymbol{\theta}) = \sum_{s=1}^{S} \left\| g(\mathbf{z}_s) - \mathbf{y}_s \right\|_2^2 + \gamma \left\| \boldsymbol{\theta} \right\|_2^2. \quad (22)$$

The derivation for the back-propagation for the scale DCFNet is similar to that for the DCFNet. The main difference is that the partial differentials $\partial \hat{g}_{uv}^* / \partial \hat{\varphi}_{uv}^l(\mathbf{x}_s)$ and $\partial \hat{g}_{uv}^* / \partial (\hat{\varphi}_{uv}^l(\mathbf{x}_s))^*$ are derived from (17) instead of (3). For the back-propagation of the tracking branch, $\partial L / \partial (\varphi^l(\mathbf{z}_s))$ is computed by replacing **z** in (9) and (10) with $\mathbf{z}_s$. For the back-propagation of the filter learning branch, the partial differential $\partial L / \partial (\varphi^l(\mathbf{x}_s))$ for $\mathbf{x}_s$ is computed instead of $\partial L / \partial (\varphi^l(\mathbf{x}))$. The partial differentials $\partial \hat{g}_{uv}^* / \partial \hat{\varphi}_{uv}^l(\mathbf{x}_s)$ and $\partial \hat{g}_{uv}^* / \partial (\hat{\varphi}_{uv}^l(\mathbf{x}_s))^*$ are computed, respectively, by:

$$\frac{\partial \hat{g}_{uv}^*}{\partial \hat{\varphi}_{uv}^l(\mathbf{x}_s)} = \frac{(\hat{\varphi}_{uv}^l(\mathbf{z}_s))^* \mathbf{y}_{s,uv}^* - \hat{g}_{uv}^* (\hat{\varphi}_{uv}^l(\mathbf{x}_s))^*}{\sum_{s=1}^{S} \sum_{k=1}^{D} \hat{\varphi}_{uv}^k(\mathbf{x}_s) (\hat{\varphi}_{uv}^k(\mathbf{x}_s))^* + \lambda \mathbf{e}}, \quad (23)$$

$$\frac{\partial \hat{g}_{uv}^*}{\partial (\hat{\varphi}_{uv}^l(\mathbf{x}_s))^*} = \frac{-\hat{g}_{uv}^* \hat{\varphi}_{uv}^l(\mathbf{x}_s)}{\sum_{s=1}^{S} \sum_{k=1}^{D} \hat{\varphi}_{uv}^k(\mathbf{x}_s) (\hat{\varphi}_{uv}^k(\mathbf{x}_s))^* + \lambda \mathbf{e}}. \quad (24)$$

4.3. Online model update

Similar to the DCFNet tracker, the scale DCFNet carries out the online update of the correlation filter layer to make the tracker quite adaptive to continuous changes in object appearance. The solution (21) of the optimization problem (20) is posed as the following incremental update process:

$$\hat{\mathbf{w}}_T^l = \frac{\beta_T \left(\sum_{s=1}^{S} \hat{\mathbf{y}}_s^* \odot \hat{\varphi}^l \left(\mathbf{x}_{s,T} \right) \right) + \sum_{t=1}^{T-1} \beta_t \left(\sum_{s=1}^{S} \hat{\mathbf{y}}_s^* \odot \hat{\varphi}^l \left(\mathbf{x}_{s,t} \right) \right)}{\beta_T \left(\sum_{s=1}^{S} \sum_{k=1}^{D} \hat{\varphi}^k \left(\mathbf{x}_{s,T} \right) \odot \left(\hat{\varphi}^k \left(\mathbf{x}_{s,T} \right) \right)^* + \lambda \mathbf{e} \right)} \cdot (25)$$
$$+ \sum_{t=1}^{T-1} \beta_t \left(\sum_{s=1}^{S} \sum_{k=1}^{D} \hat{\varphi}^k \left(\mathbf{x}_{s,t} \right) \odot \left(\hat{\varphi}^k \left(\mathbf{x}_{s,t} \right) \right)^* + \lambda \mathbf{e} \right)$$

We update the correlation filter layer similarly to the RNN as shown in Fig. 2, while keeping it a much simpler architecture. At frame *t*, the latest updated correlation filter $\mathcal{W}_{t-1}$ is used to test the samples $\{\varphi(\mathbf{z}_{s,t})\}_{s=1}^{S}$ and obtain the response output $\{g(\mathbf{z}_{s,t})\}_{s=1}^{S}$. The object size and

position are simultaneously estimated by searching the joint scale-position space for where the maximum value of the response output exists. Based on the tracking result, the training samples $\{\varphi(\mathbf{x}_{s,t})\}_{s=1}^{S}$ are extracted **and used** to incrementally update $\mathcal{W}_{t-1}$ to a new filter $\mathcal{W}_t$, such that the new filter approximately outputs S expected Gaussian responses when it is correlated with the training samples $\{\varphi(\mathbf{x}_{s,t})\}_{s=1}^{S}$.

5. Convolutional-Deconvolutional DCFNet for Tracking

We incorporate a convolutional-deconvolutional architecture into the scale DCFNet, and propose a convolutional-deconvolutional correlation filter framework, named convolutional-deconvolutional DCFNet. The convolutional-deconvolutional architecture in the convolutional-deconvolutional DCFNet is used to effectively perceive the structural information about the object and then improve generalization performance of feature representation. The global context constraints about the negative samples are introduced into the scale DCFNet in order to suppress the influence from distractors. The convolutional-deconvolutional DCFNet is able to combine the low-level and high-level features for tracking. Its architecture is shown in Fig. 4, which consists of a convolutional network and a deconvolutional network. The convolutional features are extracted by comparing the search patch **z** with the target patch **x**. The shallow features from the first and second convolutional layers are used by the context-aware correlation filtering (CACF) (corresponding to the DCFNet) as a fine-grained representation for the image. Fine-grained object localization is carried out using the correlation filter working on the low-level fine-grained representations. This correlation filter, implemented as a differentiable layer, is regularized by a global negative sample context constraint. The deep features from the fifth convolutional layer form a high-level representation for the patch. These features are used for spatial correlation analysis in a learnt generic semantic embedding space without online update to avoid tracking drift. The embedding is constrained by a domain-independent reconstruction imposed by the deconvolution network, benefitting to preserve the geometric or structural information about images. The convolutional-deconvolutional DCFNet exploits multi-resolution representations for tracking. A multiple task learning method is used to train the entire network end to end. Then, the discriminative correlation filter and discriminative spatial semantic embedding imposed by the generative reconstruction are enhanced.

5.1. Semantic embedding space

In recent deep learning-based trackers [32, 33], the semantic embedding space is only used for discriminative learning. In contrast, the proposed convolutional-deconvolutional DCFNet fully utilizes the convolutional-deconvolutional Siamese network structure with multi-resolution feature representations to realize perception of the structural information about the object. A generative image reconstruction constraint is added into the traditional discriminative learning and a more generic semantic embedding space with more generic high-level feature representations is learnt. Less sensitiveness of the unsupervised image reconstruction to the training samples brings larger generalization capability to the learnt semantic

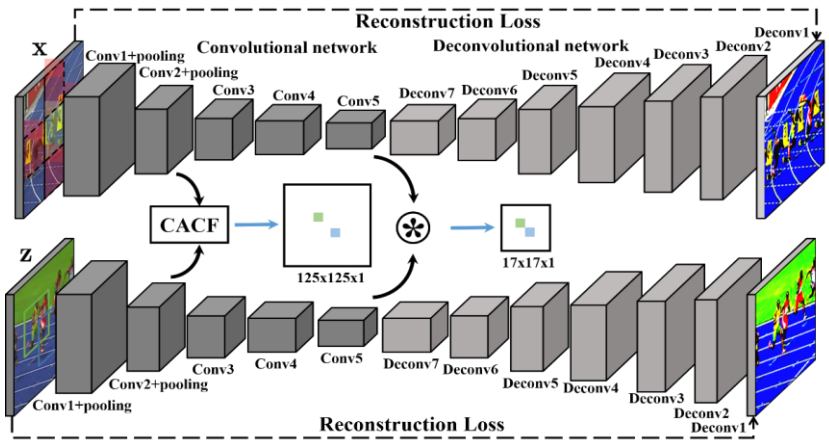


Fig. 4. The architecture of our convolutional-deconvolutional DCFNet: The "125*125*1" is the size of a correlation response map from CACF; The "17*17*1" is the size of a correlation response map from the high-level semantic correlation analysis.

embedding space. This makes tracking more robust. By using the reconstruction constraint, the geometric and structural information in the original image is preserved in the learnt semantic embedding space. This makes tracking more accurate.

A convolutional-deconvolutional architecture is utilized to learn the generic semantic embedding space. The convolutional mapping ϕ: $\mathbb{R}^{M\times N\times 3} \to \mathbb{R}^{P\times Q\times D}$ ($P < M$, $Q < N$) has five convolution layers. Either the first or second convolutional layer is accompanied with a max-pooling layer. This mapping produces a representation from the embedding space described in the fifth convolutional layer. The deconvolutional network ψ: $\mathbb{R}^{P\times Q\times D} \to \mathbb{R}^{M\times N\times 3}$ transforms the high-level representation which has a low resolution into the image space which has a high resolution, This transformation is carried out using seven stacked deconvolutional layers. We define the reconstruction loss L_{recons} and the spatial embedding loss L_{high} to optimize the semantic embedding space. Let $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_d$ be the sets of the parameters for the convolutional network and the deconvolutional network, respectively. The reconstruction loss L_{recons} for the target patch $\mathbf{x}$ and the search patch $\mathbf{z}$ is defined as:

$$L_{recons} = \sum_{s=1}^{S} \left\| \psi(\phi(\mathbf{x}_s \mid \boldsymbol{\theta}_c) \mid \boldsymbol{\theta}_d) - \mathbf{x}_s \right\|_2^2 + \left\| \psi(\phi(\mathbf{z}_s \mid \boldsymbol{\theta}_c) \mid \boldsymbol{\theta}_d) - \mathbf{z}_s \right\|_2^2, \quad (26)$$

where S is the number of scales.

The spatial embedding loss is defined as follows. Let $m \times n$ represent the spatial size for the correlation analysis in the semantic embedding space. Let $f_{p,q}$ represent the similarity of the target image and one search image which has a center away from the center of the target image by $p \times q$ pixels. The similarity between the target and search images is measured using the spatial correlation operation (the inner product) on the semantic embedding space:

$$f_{p,q} = \sum_{s=1}^{S} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \langle \phi_{p+i,q+j}(\mathbf{x}_s \mid \boldsymbol{\theta}_c), \phi_{i,j}(\mathbf{z}_s \mid \boldsymbol{\theta}_c) \rangle, \quad (27)$$

where $\phi_{i,j}(\mathbf{z}_s \mid \boldsymbol{\theta}_c)$ represents a D-dimensional channel feature vector for the position (i, j) in the representation for the search patch $\mathbf{z}_s$ in the s-th scale space and $\langle , \rangle$ is the inner product of vectors. Each search patch associates with a label $y(p,q) \in \{+1, -1\}$ where "+1" indicates that it is a positive sample and "-1" indicates it is a negative sample. Then, the global high level logistic loss is defined as:

$$L_{high} = \frac{1}{|\mathcal{R}|} \sum_{(p,q)\in\mathcal{R}} \log(1 + \exp(-y(p,q)\, f_{p,q})), \quad (28)$$

where $\mathcal{R}$ represents a search grid, and $|\mathcal{R}|$ is the number of the search patches. The combination loss is $L_{recons} + L_{high}$.

5.2. Context aware correlation filter

The features with high-resolutions at the low layers, i.e., the fine-grained representations of the image, are utilized for correlation filtering analysis to accurately localize the object. A global contextual constraint from negative samples is added into the spatial correlation analysis as a regularization in order to reduce the influences from distractors. This is carried out by adding a differentiable correlation filtering layer, which is trained from end-to-end and further adaptively updated online during tracking.

Inspired by the context-aware correlation filter [50], the correlation filter is regularized using the global negative sample context. In each frame, k context image patches $\{\mathbf{x}^i\}_{i=1}^k$ are sampled around the target image $\mathbf{x}^0$. Let $\varphi^l(\cdot)$ be a feature mapping based on the parameters $\boldsymbol{\theta}_{el}$ of the low-level convolutional layers (the first and second layers) in the convolutional network, i.e., $\varphi^l(.) = \varphi^l(. \mid \boldsymbol{\theta}_{cl})$. Let $\boldsymbol{\Phi}^l(\mathbf{x}_s^0)$ be the circulant feature matrices of the low-level fine-grained CNN features of channel l for patch $\mathbf{x}^0$ at scale s, corresponding to the features $\varphi^l(\mathbf{x}_s^0)$. Different from using S scales to represent the target, only one scale is used for negative samples. Let $\boldsymbol{\Phi}^l(\mathbf{x}^i)$ be the circulant feature matrices of the low-level fine-grained CNN features of channel l for context patch $\mathbf{x}^i$. The context patches are used as negative samples that include various distractors and backgrounds. A correlation filter is learnt in order that the target patch has a high response and the context patches have responses close to zero:

$$\min_{\mathbf{w}^l} \sum_{s=1}^{S} \left\| \boldsymbol{\Phi}^l(\mathbf{x}_s^0)\mathbf{w}^i - \mathbf{y}_s \right\|_2^2 + \lambda_1 \left\| \mathbf{w}^l \right\|_2^2 + \lambda_2 \sum_{i=1}^{k} \left\| \boldsymbol{\Phi}^l(\mathbf{x}^i)\mathbf{w}^l \right\|_2^2, \quad (29)$$

where λ_1 and λ_2 are regularization coefficients. The closed form solution for the context-aware correlation filter in the Fourier domain is:

$$\hat{\mathbf{w}}^l = \frac{\sum_{s=1}^{S} (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{y}}_s}{\sum_{s=1}^{S} (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\varphi}^l(\mathbf{x}_s^0) + \lambda_1 \mathbf{e} + \lambda_2 \sum_{i=1}^{k} (\hat{\varphi}^l(\mathbf{x}^i))^* \odot \hat{\varphi}^l(\mathbf{x}^i)}. \quad (30)$$

The correlation filter $\{\mathbf{w}^l\}_{l=1}^D$ is learnt based on the

representations $\{\varphi^l(\mathbf{x}_s^0)\}_{s=1}^S$ for the target image and the representations $\{\varphi^l(\mathbf{x}^i)\}_{i=1}^k$ for the global contexts.

The context-aware correlation filter in [50] utilizes hand-crafted features for correlation analysis. In contrast, we learn actively a low-level fine-grained representation for fitting to a correlation filter. This is achieved by transforming the correlation filter to a differentiable correlation filter layer and adding it after a low-level convolutional layer for convolutional mapping. In this way, the entire convolutional-deconvolutional network can be trained end-to-end. Furthermore, the representations from a low-level convolutional layer for convolutional mapping are fine-grained. The lower feature maps are more effective for accurately localizing the object. These representations are denoted as $\{\varphi^l(\mathbf{z}_s) = \varphi^l(\mathbf{z}_s \mid \boldsymbol{\theta}_{cl})\}_{s=1}^S$ for a search image $\mathbf{z}$. The correlation response maps for z are obtained by:

$$g(\mathbf{z}_s) = \sum_{l=1}^{D} \boldsymbol{\Phi}^l(\mathbf{z}_s)\mathbf{w}^l = \mathcal{F}^{-1}\left(\sum_{l=1}^{D} \hat{\varphi}^l(\mathbf{z}_s) \odot \hat{\mathbf{w}}^l\right), \quad (31)$$

where $\boldsymbol{\Phi}^l(\mathbf{z}_s)$ is the circulant matrix of the feature representation $\varphi^l(\mathbf{z}_s)$. The feature representations of the low-level convolutional layers are learnt using the low-level correlation filtering loss defined by:

$$L_{low} = \sum_{s=1}^{S} \left\| g(\mathbf{z}_s) - \mathbf{y}_s \right\|_2^2 = \sum_{s=1}^{S} \left\| \sum_{l=1}^{D} \boldsymbol{\Phi}^l(\mathbf{z}_s)\mathbf{w}^l - \mathbf{y}_s \right\|_2^2. \quad (32)$$

The low-level convolutional parameters $\boldsymbol{\theta}$ of the CNN are optimized by minimizing (32). We derive the backward formulas in the frequency domain in Appendix B. Let Re(·) be the real part of a complex-valued matrix. Let μ be the denominator of $\hat{\mathbf{w}}$ in (30):

$$\mu = \sum_{s=1}^{S} (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\varphi}^l(\mathbf{x}_s^0) + \lambda_1 + \lambda_2 \sum_{i=1}^{k} (\hat{\varphi}^l(\mathbf{x}^i))^* \odot \hat{\varphi}^l(\mathbf{x}^i). \quad (33)$$

The derivatives of L_{low} in (32) are obtained by:

$$\frac{\partial L_{low}}{\partial \hat{g}^*(\mathbf{z}_s)} = 2(\hat{g}(\mathbf{z}_s) - \hat{\mathbf{y}}_s), \quad (34)$$

$$\frac{\partial L_{low}}{\partial \varphi^l(\mathbf{z}_s)} = \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{g}^*(\mathbf{z}_s)} \odot \hat{\mathbf{w}}^{l*}\right), \quad (35)$$

$$\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^l} = \sum_{s=1}^{S} \frac{\partial L_{low}}{\partial \hat{g}^* \varphi^l(\mathbf{z}_s)} \odot (\hat{\varphi}^l(\mathbf{z}_s))^*, \quad (36)$$

$$\frac{\partial L_{low}}{\partial \varphi^l(\mathbf{x}_s^0)} = \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^l} \odot \frac{\hat{\mathbf{y}}_s^* - 2Re((\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{w}}^l)}{\mu}\right), \quad (37)$$

$$\frac{\partial L_{low}}{\partial \hat{\varphi}^l(\mathbf{x}^i)} = \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^l} \odot \frac{-2Re((\hat{\varphi}^l(\mathbf{x}^i))^* \odot \hat{\mathbf{w}}^l)}{\mu}\right). \quad (38)$$

5.3. Multiple task learning and online tracking

The two differentiable components, the semantic embedding learning described in Section 5.1 and the context-aware correlation filter described in Section 5.2, complement each other for localization and tracking using multi-resolution representations. By using multiple task learning, the low-level detailed representation and high-level semantic description are simultaneously learnt with complementary enhancement. The network is trained end-to-end. The multiple task loss is defined as:

$$L_{all} = L_{high} + L_{low} + L_{recons} + \gamma \left\| \boldsymbol{\theta} \right\|_2^2, \quad (39)$$

where L_{recons}, L_{high}, and L_{low} are defined in (26), (28), and (32) respectively. The ℓ_2-norm of the network weights is incorporated into the loss function in order that the network is regularized for increasing the generalization of the network.

During tracking inference, at frame *T*, large search patches with multiple scales are cropped, centered at the previous estimated object position, denoted as $\{\mathbf{z}_s\}_{s=1}^S$. The search patches are input into the convolutional network to yield the fine-grained representations and the semantic embedding representations. The fine-grained representations are input to the context-aware correlation filter shown in (31). The semantic embedding representations are estimated using the spatial correlation analysis shown in (27). Then, the object state is evaluated via searching for the maximum of the combined correlation response:

$$\arg\max_{(p,q,s)} f_{p,q}(\mathbf{z}_s) + g_{p,q}(\mathbf{z}_s), \quad (40)$$

where $g_{p,q}(\mathbf{z}_s)$ is, in the response map, the correlation response of the position which is of $p \times q$ pixels away from the center of $\mathbf{z}_s$. The bilinear interpolation method is used to up-sample the high level spatial correlation response map $f(\cdot)$ to ensure that its up-sampled resolution is the same as that of the low level map $g(\cdot)$. During online tracking, only the correlation analysis in the frequency domain and the neural network feed-forward pass are involved. This makes the tracker very efficient.

We make use of a fusion of updates of long and short terms. The semantic embedding representations $\{\phi(\mathbf{x}_s \mid \boldsymbol{\theta}_c)_{s=1}^S$ of the target image in (27) are only calculated for the target image in the initial frame, i.e., only

similarities of search images to the target image in the initial frame are used for the spatial correlation estimation. This is useful for generic long term tracking. Linear interpolation is used to update online the context-aware correlation filter $\mathcal{W}$ in (30) in order to adapt to changes in object appearance. Let $\tilde{f}(\mathbf{z}_o)$ be the maximum in the spatial correlation response map at the initial frame and $\tilde{f}(\mathbf{z}_T)$ be the maximum in the response map at the current frame. A dynamic learning rate α_t is defined as:

$$\alpha_T = \alpha \tilde{f}(\mathbf{z}_T) / \tilde{f}(\mathbf{z}_o), \tag{41}$$

where α is the basic learning rate. The context aware correlation filter $\mathcal{W}$ is updated by:

$$\mathcal{W}_T \leftarrow \alpha_T \mathcal{W}_T + (1-\alpha_T)\mathcal{W}_{T-1}. \tag{42}$$

As the representation for the target image at the initial frame is fixed, the correlation responses over time indicate changes in object appearance and background disturbance. By using this update method, the long-term update and the short-term update complement each other in the temporal domain.

6. Experiments

All the experiments were conducted on a workstation with an Intel Xeon 2630 at 2.4GHz and a NVIDIA GeForce GTX 1080 GPU. The proposed DCFNet tracker, scale DCFNet tracker, and convolutional-deconvolutional DCFNet tracker were implemented using MATLAB with MatConvNet [57]. the code of DCFNet is available at: at:https://github.com/foolwood/DCFNet. We used the following datasets to carry out ablation studies and overall performance evaluations for tracking:

- **The OTB datasets** [1, 2], OTB 2013 and OTB 2015, are standard benchmarks for visual tracking, containing, respectively, 50 and 100 fully annotated objects associated with 11 different attributes. Two evaluation metrics, distance precision and overlap precision, were exploited on the OTB datasets. The distance precision for a video is the proportion of the frames where the center location error is less than the threshold of 20 pixels. The overlap precision is the proportion of the frames in which the overlap ratio between the predicted bounding box and the ground truth bounding box is larger than the threshold of 0.5. Performance is also described using success plots in which overlap precisions are plotted in the range of the thresholds of intersection-over-union between the predicted bounding box and the ground truth box. In these plots, the areas under the curves (AUC) are used to rank trackers, displayed in the legends.
- **The VOT challenge** [3] is one of the most influential and largest annual events in the tracking field. On the VOT2015 dataset [3] and the VOT2017 dataset [54], the measure exploited to quantitatively analyze the tracking performance is the expected average overlap (EAO) that is an estimator of the average overlap that a tracker is expected to attain on a large collection of short-term sequences with the same visual properties as the given dataset. The measure addresses the problem of increased variance and bias of the average overlap measure due to variable sequence lengths.

In the following, the implementation settings are introduced. Ablation studies of the effectiveness and efficiency analysis of main components in the proposed trackers are described. The overall performance of our trackers was evaluated in comparison with the state-of-the-art trackers.

6.1. Implementation settings

The main implementation settings include the aspects of network architecture, training data, and parameter setting.

6.1.1. Network architecture

The feature extraction of our DCFNet and scale DCFNet consist of 2 convolutional layers [43] with the kernel size 3×3 and a Relu operation appended at the end of each convolutional layer. A local response normalization was added to output the final feature representation. The final output feature representation was forced to 32 channels.

In our convolutional-deconvolutional DCFNet, the convolutional network has the same structure as the baseline SiamFC [32], where the AlexNet was used and the fully connected layers were removed. The input size is 255×255×3. The output supplied from the Conv5 layer has a size of 22×22×256. The output is input for spatial correlation analysis. It is also input into the deconvolutional network for image reconstruction. The deconvolutional network with seven deconvolutional layers was removed during the tracking inference process. The fine-grained representations with size 125×125×8 from the Conv2 layer in the convolutional network are input into layer of the context-aware correlation filter for accurately localizing the

object.

6.1.2. Training dataset

The dataset of image sequences for ILSVRC (ImageNet large scale visual recognition challenge) [41, 58, 59, 60] was used to train our DCFNet, scale DCFNet, and convolutional-deconvolutional DCFNet from end-to-end. This training set consists of 7,911 objects and has little correlation with the OTB and VOT datasets. The dataset has more than 4000 vides and almost 2,000,000 annotated image patches of objects. For training the DCFNet and the scale DCFNet, in each video snippet of an object we collected each pair of frames within 10 nearest frames, and fed the cropped pair of target patches of 2×padding size to the network. The resulted 5,507,660 pairs in total were used to train the DCFNet and scale DCFNet. The cropped inputs were resized to a spatial resolution which is consistent between the offline training and online tracking phases. A case study of the tradeoff between the tracking accuracy and speed suggests a resolution of 125×125. For training the convolutional-deconvolutional DCFNet, pairs of frames which contain the same object were picked randomly. The target and search patches were cropped by the padding size of 2. Since positive and negative samples are included in each patch, the patch was resized to the input size of 255×255.

6.1.3. Parameter setting

For the correlation filter layer, the regularization coefficient λ in (2) and (20) was set to 10^{-4}. The online learning rate β_t in (14) and (20) was fixed to 0.008. The Gaussian spatial bandwidth was set to 0.1 for both online tracking and offline training. Similar to [49], we used a patch pyramid with the scale factors:

$$\left\{\sigma^s \mid \sigma = 1.02,\ s = \left\lfloor -\frac{S-1}{2} \right\rfloor, \left\lfloor -\frac{S-3}{2} \right\rfloor, ..., \left\lfloor \frac{S-1}{2} \right\rfloor \right\}. \quad (43)$$

The stochastic gradient descent solver with the momentum of 0.9 was used to train the networks from the scratch. The weight decay γ in (5) and (22) was set to 0.0005. The learning rate decays exponentially from 10^{-2} to 10^{-5}. The model was trained for 50 epochs where a mini-batch size is 32.

For the convolutional-deconvolutional DCFNet, when the model was trained, only the dynamic learning rate α_t in (41) affects the online tracking. We set α in (41) to 0.017. The regularization parameters in (29) were set as $\lambda_1 = 1e^{-4}$ and $\lambda_2 = 0.1$.

6.2. Ablation study

An ablation analysis was carried out in terms of network architectures of DCFNet, the number of scale levels on the scale DCFNet, and the effect of generic semantic embedding and fine-grained object localization for the convolutional-deconvolutional DCFNet.

For the network architectures, the number of the training parameters and the size of the receptive field gradually increase when the convolutional layers go deeper. Table 1 shows that on OTB2013 our DCFNet with only conv2 achieves better performance in contrast with deeper conv3. To give a better insight into this observation, we modified our DCFNet with conv2 using dilation convolution to approximate the receptive field of deeper conv3. This new variant with a small quantity of parameters also performs better than deeper conv3. The variants, DCFNet-3s, DCFNet-5s, and DCFNet-7s, enhance DCFNet with scale estimation at 3, 5, 7 adjacent scale levels only in the tracking process, i.e., there is one scale in the learning process. It is found that the design of 3 scales has a good balance between performance and tracking speed.

Table 1: Ablation study of the DCFNet with different architectures and different numbers of scale levels during tracking on OTB2013 [1] using mean overlap precision at the threshold of 0.5, the mean distance precision at 20 pixels, and the mean speed (frames per second).

Tracker	Overlap precision	Distance precision	Frames per second
DCFNet-conv1	61.0	70.4	211
DCFNet-conv2-dilation	66.3	77.3	120
DCFNet-conv2-1s	67.7	79.1	187
DCFNet-conv2-3s	78.5	86.7	109
DCFNet-conv2-5s	76.3	83.8	69
DCFNet-conv2-7s	77.4	88.0	53
DCFNet-conv3	64.3	75.3	61

We compared our DCFNet tracker and scale DCFNet tracker with some baselines including DCF+VGG and DCF+SiamFC and some variants including DCF (linear correlation filter version of [9]), SAMF [11], and DSST [14]. The results are shown in Table 2, where DCFNet does not consider scale factors in the both training and tracking processes, and DCFNet-3s enhances DCFNet with scale estimation at 3 adjacent scale levels only in the tracking process. The following points are noted:

- Compared with the traditional correlation filter-based tracker DCF using hand-crafted features, the DCFNet using the self-learnt features and linear correlation filters obtains a distance precision gain of 6.3~7.9%

and an overlap precision gain of 6.1~9.9%, while retaining a real time tracking speed. The variants, DCF+VGG and DCF+SiamFC, use the original feature extraction in VGG [43] and SiamFC [32], rather than the feature extraction in DCFNet. Compared with them, our specifically learnt feature representation in the DCFNet leads to better tracking performance by a notable margin. Therefore, the feature representation learnt from the end-to-end network pre-training is effective for tracking.

- By extending the search space from a single position space to the joint scale-position space, the scale DCFNet significantly boosts the tracking performance and outperforms the traditional multi-scale trackers SAMF [11] and DSST [14] by a large margin.
- Enhancing DCFNet with multiple scales in both the learning and tracking processes yields more accurate results than with multiple scales in the tracking process alone.
- Our scale DCFNet carries out appearance modeling in the joint scale-position space in both the training and tracking processes and ranks first in Table 2 for all the precision metrics.

Table 2. Ablation study of our DCFNet and scale DCFnet components on the OTB datasets using the mean overlap precision at the threshold of 0.5, the mean distance precision of 20 pixels, and the mean speed (frames per second).

Trackers		OTB-2013		OTB-2015		Frames per second
		Overlap precision	Mean distance precision	Overlap precision	Mean distance precision	
Our trackers	DCFNet	67.7	79.1	63.7	76.8	187
	DCFNet-3s	78.5	86.7	72.8	79.4	109
	Scale DCFNet	84.3	88.1	77.6	82.9	67
Variants	DCF+VGG	62.1	66.1	61.7	66.9	88
	DCF+SiamFC	66.8	74.2	64.0	68.0	77
Baselines	DCF [9]	61.6	72.8	54.8	68.9	292
	SAMF [11]	67.7	78.5	64.0	74.3	12
	DSST [14]	67.1	74.7	60.9	68.9	46

To highlight the tradeoff between tracking accuracy and speed for our scale DCFNet, we compared two kinds of different settings of the proposed scale DCFNet model: one kind was pretrained using 169×169, 125×125, 63×63, and 33×33 four different input spatial resolutions and the other was pretrained using 64, 32, 16, 8, and 4 five different numbers of the final output feature channels. Fig. 5 shows the tracking AUC accuracy and speed analyses of these scale DCFNet models and some other real time trackers. It is seen that decreasing input spatial resolution causes a large reduction in the AUC accuracy, although it provides a significant speedup. The AUC performance of a small number of output feature channels only falls by 4% while the run time cost is reduced by a factor of 3. Compared with the fast trackers in [9, 19, 32, 34, 40, 61, 62, 63], the proposed scale DCFNet achieves better performance both in accuracy and speed. According to the different computational resources available, a member in the scale DCFNet model with tracking speeds ranging from 60 FPS (frames per second) to 190 FPS is feasible for real applications.

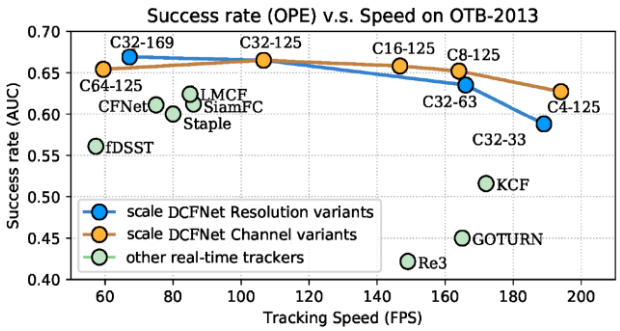


Fig. 5. The tracking speed and AUC performance on OTB-2013: The tracking speed ranges from 60 FPS (frames per second) to 190 FPS with different input spatial resolutions and different numbers of the output feature channels: "CX-Y" stands for a member in the scale DCFNet family consisting of X output feature channels and Y×Y input resolution. For example, "C32-169" means that our scale DCFNet has 32 channels and 169×169 resolution.

Table 3. Ablation study of effectiveness of tracking components of the convolutional-deconvolutional DCFnet (CD-DCFNet): On the OTB datasets, we used the mean overlap precision with the threshold of 0.5 and the mean distance precision with the threshold of 20 pixels; On the VOT2015 dataset, we used the EAO (expected average overlap) as well as the mean speed (frames per second).

Trackers		OTB-2013		OTB-2015		VOT2015	Frames per second
		Overlap precision	Distance precision	Overlap precision	Distance precision	EAO	
Our tracker	CD-DCFNet	84.2	88.5	78.5	83.6	0.315	65
Variants of our tracker	CDSiam	79.0	83.9	75.4	80.7	0.293	86
	CACFNet	83.8	87.6	77.7	82.7	0.271	109
	CACFNet+	83.9	88.3	78.0	83.1	0.277	109
The baselines	SiamFC [32]	77.8	80.9	73.0	77.0	0.289	86
	CFNet [40]	71.7	76.1	70.3	76.0	0.217	75
	CACF [50]	75.4	80.3	68.9	79.1	0.199	13

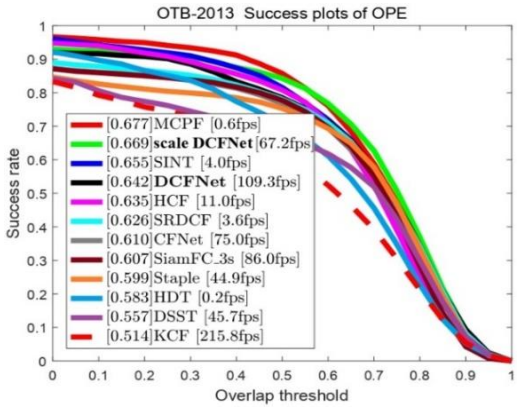
To show the effect of the generic semantic embedding space as well as the fine-grained object localization in the context-aware spatial correlation filter, we made comparison between the variants of our convolutional-deconvolutional DCFNet (CD-DCFNet) tracker and the baseline trackers. Table 3 shows the results. The following points are revealed:

- **Generic semantic embedding:** The variant, CDSiam, adds a convolutional-deconvolutional network

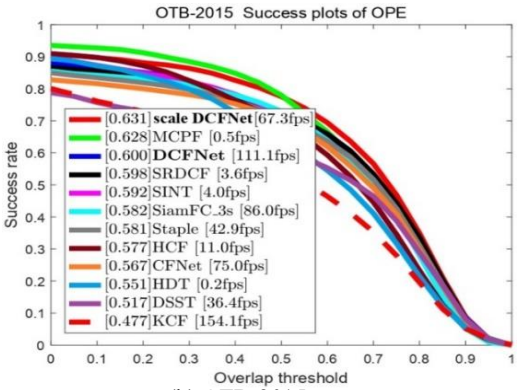
architecture-based image reconstruction constraint into the SiamFC tracker [32]. On the OTB2015 dataset, CDSiam obtains large distance precision gains of 3.7% compared with SiamFC. Such a domain-independent reconstruction constraint improves generalization capability for the learnt semantic embedding space as well as ensures the robustness of tracking.

- **Context-aware spatial correlation filter:** The variant, CACFNet, cascades the CACF [50] with CNNs. It learns fine-grained representations of the Conv2 layer, fitted to a context-aware correlation filter for tracking. On the OTB datasets, it obtains larger overlap precision by more than 8%, compared with the CACF [50] tracker which uses the previously widely used HoG features. This indicates that the learnt representations of features are more discriminative than the HOG features. In contrast with the CFNet tracker in [40] that learns Conv2 representations for a general correlation filter, our convolutional-deconvolutional DCFNet-based tracker obtains significant overlap precision gains of more than 10%. This indicates that exploiting fine-grained Conv2 representations with lower channels and incorporating a global constraint of contexts into the correlation filter lead to stable modeling of object appearance.
- **Multiple task learning:** The variant, CACFNet+, improves CACFNet by adding spatial correlation analysis to semantic embedding space, as well as adding a reconstruction constraint for training. During tracking inference, CACFNet+ estimates the object state using the spatial context-aware correlation filter which supplies the fine-grained correlation responses. The performance improvement of CACFNet+ in contrast with CACFNet indicates that a high-level constraint can reinforce the fine-grained correlation-based appearance modeling for discriminative tracking. The convolutional-deconvolutional DCFNet outperforms CACFNet+. This indicates effectiveness of combination of fine-grained representation with the semantic spatial correlation responses during the tracking inference.
- **Efficiency:** The convolutional-deconvolutional DCFNet-based tracker carries out tracking in real time while obtaining significant improvement of performances on the datasets.

6.3. State-of-the-art comparison



(a) OTB-2013

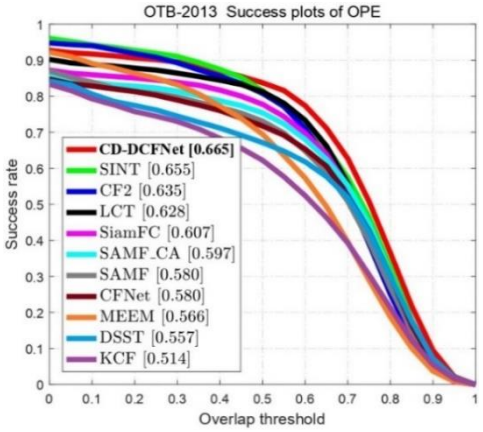


(b) OTB-2015

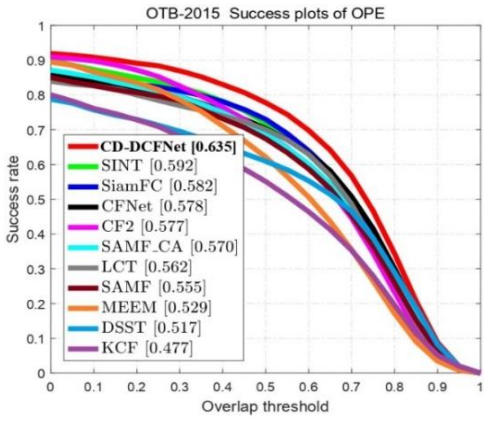
Fig. 6. The success plots on the OTB2013 [1] and OTB2015 [2] of the DCFNet and the scale DCFNet compared with correlation filter-based trackers and the-state-of-the-art trackers.

Fig. 6 shows the results of comparison between the DCFNet, the scale DCFNet, the correlation filter-based trackers including CSK [22], KCF [9], DSST [14], HCF [16], HDT [18], SRDCF [20], CFNet [40], and MCPF [64], and the-state-of-the-art trackers including TGPR [65], MEEM [44], Staple [19], CNN-SVM [29], SINT [33], and SiamFC [32], on the OTB2013 and OTB2015 datasets. From Fig. 6(a) and Fig. 6(b), it is seen that our simple feature training in a single scale position space leads to 10% and 6.2% gains in success plots on OTB2015 compared with KCF and DSST using HOG features respectively. Our automatic feature learning and appearance correlation modeling in the joint scale-position space lead to AUC gains of more than 11% in the success plots of one pass evaluation (OPE) on OTB-2013 and OTB-2015 compared with KCF and DSST that exploit HOG features and do not consider the scale factor in the training process. Although our feature learning network only contains two convolutional layers and is much shallower than [16, 18, 29], superior performance with much faster speed was achieved. Our algorithm is orders of

magnitude faster (100×) than the recent top ranked correlation filter-based tracker MCPF [64], while achieving a comparable performance. Because of a more adaptive online update strategy, the proposed methods work better than the recent SINT [33] and SiamFC [32]. Compared with CFNet [40] which is end-to-end pre-trained in the position space, the scale DCFNet achieves an AUC gain of more than 6% because it was learnt end-to-end in the joint scale-position space and a more appropriate regression loss was used.



(a) OTB2013



(b) OTB2015

Fig. 7. The success plots for the convolutional-deconvolutional DCFNet tracker (CD-DCFNet) compared with state-of-the-art trackers on (a) OTB2013 and (b) OTB2015.

The proposed convolutional-deconvolutional DCFNet (CD-DCFNet) tracker was compared with the state-of-the-art tracking algorithms SAMF CA [50], CFNet [40], SiamFC [32], SINT [33], LCT [47], MEEM [44], CF2 [16], SRDCF [20], KCF [9], and DSST [14] on the OTB2013 dataset and the OTB2015 dataset. The success plots are shown in Fig. 7. Compared with the competing trackers using deep learning features, the convolutional-deconvolutional DCFNet yields the best results. Its AUC scores on these two datasets are 63.5% and 66.5%, respectively. In contrast with the Siamese network tracking algorithms [32, 33, 40], our convolutional-deconvolutional DCFNet tracker increases the AUC score by more than 4.3% especially on OTB2015. Among the correlation filter trackers which use pre-trained features, the CF2 tracker obtains the AUC with 57.7% running at 15 frames per second on the OTB2015 dataset. Our convolutional-deconvolutional DCFNet tracker increases AUC by 10.4% and its tracking speed is faster than CF2 by more than 3 times. Among the tracking algorithms running in real time, the LCT, KCFM, EEM, and DSST less robustly and less accurately track the object, or lose the track under background clutters. The results indicate that accurate and robust object localization is achieved by the combination of the generic semantic embedding learning and the context-aware spatial correlation analysis.

On the VOT2015 challenge dataset, the DCFNet and the scale DCFNet were compared with the 62 participating trackers based on the expected average overlap (EAO) measure. They were ranked by EAO. The results are shown in Fig. 8, in which the horizontal coordinate indicates the ranks of the trackers, while the vertical coordinate indicates the EAO values of the trackers. The horizontal gray line in the figure is the VOT2015 state-of-the-art bound. Our scale DCFNet is ranked within the Top-10 trackers in the overall performance evaluation. Compared with the rest of the Top-10 trackers, the scale DCFNet has the fastest tracking speed on the speed evaluation unit called equivalent filter operations (EFO). In contrast with correlation filter-based trackers, such as DeepSRDCF [17], MUSTer [46], and KCF [9], our DCFNet and scale DCFNet achieve an excellent balance between accuracy and speed.

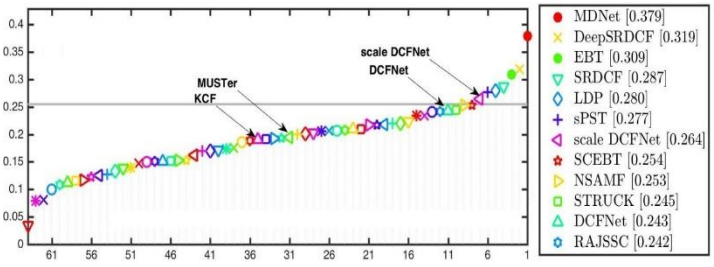
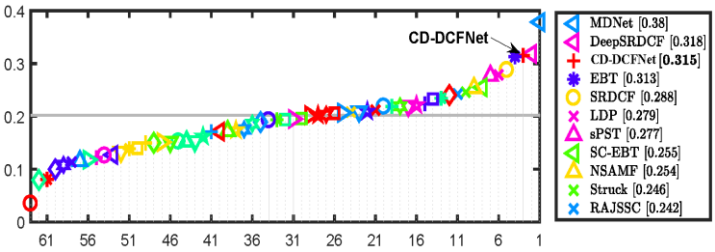


Fig. 8. The EAO plot for the proposed CD-DCFNet tracker and the participating trackers on the VOT2015 challenge: The Top-11 trackers are listed in the legend and their tracking speeds are shown in EFO values.

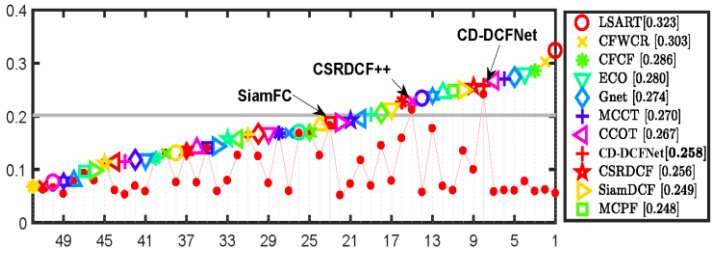
Fig. 9 compares our convolutional-deconvolutional DCFNet tracker with the state-of-the-art trackers on VOT2015 [3] and VOT2017 datasets [54] based on the EAO measure. In the figure, the horizontal coordinate indicates

the ranks of the trackers, while the vertical coordinate indicates the EAO values of the trackers. The horizontal grey lines show the bounds of the state-of-the-art. The convolutional-deconvolutional DCFNet ranks the third and eighth respectively on the two datasets in the overall performance evaluation. The red polygonal line in Fig. 9(b) shows the accuracies of the trackers indicated by the horizontal coordinate when the trackers run in real time. Our CD-DCFNet ranks first in the VOT2017 real-time experiment. Among the top ten competing tracking algorithms on the VOT2015 dataset, only the NSAMF tracking algorithm runs in real time while its EAO score is 0.254. The convolutional-deconvolutional DCFNet runs in real time (65 frames per second) while its EAO score is 0.315. It yields accuracy scores comparable to MDNet and Deep-SRDCF, while running faster by orders of magnitude. Compared with the baseline SiamFC [32] whose EAO score is 0.188 on the VOT2017 dataset, the convolutional-deconvolutional DCFNet substantially increases EAO by 7.0%, demonstrating its superiority in accuracy and robustness of tracking. To test the efficiency of the convolutional-deconvolutional DCFNet, we conducted the real time experiments on VOT2017. CSRDCF++ [51] obtains top performance in real time using an implementation with optimized C++. The convolutional-deconvolutional DCFNet yields state-of-the-art real-time performance while its EAO is 0.241 which is larger, by 14%, than the EAO obtained by the VOT2017 winner. The SiamDCF for the VOT2017 challenge is the initial version of the convolutional-deconvolutional DCFNet. It also uses multi-resolution representations to achieve correlation analysis. In contrast with SiamDCF, by using a domain-independent reconstruction constraint and a global context constraint, the convolutional-deconvolutional DCFNet carries out correlation analysis on a learnt semantic embedding space and achieves discriminative fine-grained object appearance modeling.

We compared our trackers with more recent real-time trackers [21, 66, 67, 68, 69]. The comparison is shown in Table 4. It is seen that our trackers are still comparable to the more recent real-time trackers, where some of them even extensively trained using transformers [66, 67]. We only used the AlexNet as the backbone network, so the size of the network in our trackers is much less than the sizes of the networks in the competing trackers.



(a) EAO on VOT2015



(b) EAO on VOT2017

Fig. 9. The EAO plot for the proposed CD-DCFNet tracker and the competing trackers on (a) VOT2015 and (b) VOT2017: Legends are shown for top performing trackers.

Table 4. Comparison with more recent real-time tackers on the OTB-2015 dataset in terms of AUC score.

scale-DCFNet	DC-DCFNet	HCAT [66]	E.T.Track [67]	LightTrack [68]	ECO [21]	ATOM [69]
63.1	63.5	68.1	67.8	66.2	69.1	66.9

6.4. Qualitative analysis

We show examples to validate the generality of the proposed trackers. In order to verify the improvement of the DCFNet and the scale-DCFNet in visual object tracking for changes in object shapes and scale estimation, we compared the DCFNet tracker and the scale-DCFNet tracker with the correlation filtering-based trackers, MCPF [64], SAMF [11], DSST [14], and the Siamese network-based trackers, CFNet [40] and SiamFC [32]. The comparison was carried out on the two challenging videos of human3 and skinning.

Fig. 10 shows the results of comparison between the DCFNet tracker, the scale-DCFNet tracker, and the competing trackers on the human3 sequence. Around Frame 100, the object is occluded frequently by electric poles and other pedestrians with appearances similar to the object. Our trackers well handle these occlusions and background Clutters. However, for SAMF [11], tracking drift occurs, because it only uses the HOG features which lack sufficient discriminative ability for pedestrians with similar appearances. At Frame 300, only our trackers and MCPF [64] can track the object well under the complex situations. For the DSST [14] which cascades one-dimensional scale correlation filters for estimating object scales, the object is

erroneously attracted by distractors. The Siamese network-based tracker SiamFC [32], which does not update the appearance model online, is also unable to adapt to the scenes with occlusions. From Frame 300 to Frame 600, the focal length of the camera undergoes significant changes, and the image scales of the objects in the images jump. It is seen that our scale DCFNet tracker has strong adaptability to changes in scales due to the learning in the joint scale-position space. Although the discriminative correlation learning algorithm MCPF [64] based on particle filtering uses more scale samples, it does not enough correctly estimate the scale of the object due to the lack of the joint learning of multi-scale samples.

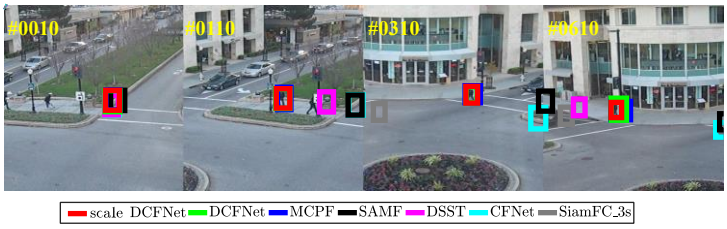


Fig. 10. The results of the DCFNet tracker, the scale-DCFNet tracker, and the competing trackers on the human3 sequence.

Fig. 11 shows the results of our DCFNet tracker, the scale DCFNet tracker, and the competing trackers on the skiing sequence in which there are not only the significant changes in object scales and large object deformations, but also, in the camera's perspective, fast motion of objects with low-resolutions. Because the multiple difficult scenes appear in the same video, this video poses great challenges for trackers. It is seen that SAMF [11] does not accurately track the object at the very beginning of tracking due to the use of very simple feature representation. After Frame 30, only the DCFNet tracker and the scale DCFNet-based tracker accurately track the objects. The scale DCFNet tracker more accurately estimates significant changes in scales. This validates the advantage of scale estimation obtained by learning in the joint position-scale space.

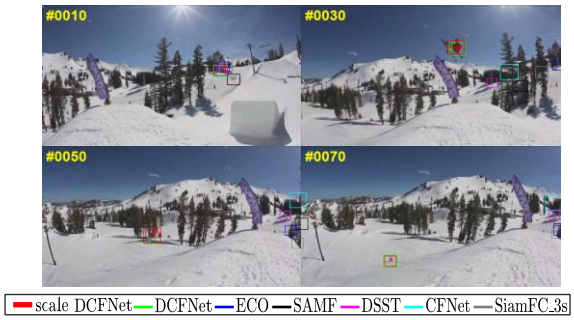


Fig. 11. The results of the DCFNet tracker, the scale DCFNet tracker, and the competing trackers on the skiing sequence.

7. Conclusion

In this paper, we focus on tracking in real time, which is essential in real applications. We have proposed a DCFNet tracker to unify the feature representation learning and correlation filter-based appearance modeling within an end-to-end learnable framework. The DCFNet is quite efficient benefiting from the lightweight feature learning network and the Fourier frequency domain-based fast correlation modeling in the correlation filter layer. The DCFNet has been extended to the scale DCFNet based on a joint scale-position space. The scale DCFNet enables feature learning to obtain accurate predictions of object scale and position. We have extended the scale DCFNet to the convolutional-deconvolutional DCFNet. A domain-independent image reconstruction constraint has been incorporated into the semantic embedding learning to generate high-level representations which maintain the structural information about images. A fine-grained context-aware correlation filter is learnt for accurately localizing the object. It is updated online for adaptive tracking. Evaluations on several benchmarks demonstrate that that the end-to-end learning improves the performance and our DCFNet, scale DCFNet, and convolutional-deconvolutional DCFNet obtain a great balance between accuracy and speed.

References

1. Wu Y, Lim J, and Yang M H, "Online object tracking: a benchmark," *in Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 2411-2418, 2013.
2. Wu Y, Lim J, and Yang M H, "Object tracking benchmark," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834-1848, 2015.
3. Kristan M, Matas J, Leonardis A, and Felsberg M, "The visual object tracking VOT2015 challenge results," *in Proc. of the IEEE International Conference on Computer Vision Workshops*, pp. 564-586, 2015.
4. Tan K and Wei Z, "Learning an orientation and scale adaptive tracker with regularized correlation filters," *IEEE Access*, vol. 7, pp. 53476-53486, 2019.
5. Wu Q, Yan Y, Liang Y, Liu Y, and Wang H, "DSNet: Deep and shallow feature learning for efficient visual tracking," *in Proc. of Asian Conference on Computer Vision*, pp. 119-134, Dec. 2018.
6. Zhong Z, Yang Z, Feng W, Wu W, Hu Y, and Liu C L, "Decision controller for object tracking with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 28069-28079, 2019.
7. Kalal Z, Mikolajczyk K, and Matas J, "Tracking learning-detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409-1422, 2012.
8. Hare S, Golodetz S, Saffari A, Vineet V, Cheng M M, Hicks S L, and Torr P H S, "Struck: Structured output tracking with kernels," *IEEE trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096-2109, 2016.
9. Henriques J F, Caseiro R, Martins P, and Batista J, "High-speed tracking with kernelized correlation filters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583-596, 2015.

10. Nam H and Han B, "Learning multi-domain convolutional neural networks for visual tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293-4302, 2016.
11. Li Y and Zhu J, "A scale adaptive kernel correlation filter tracker with feature integration," *in Proc. of European Conference on Computer Vision*, pp. 254-265, 2014.
12. Kalal Z, Matas J, and Mikolajczyk K, "P-N learning: Bootstrapping binary classifiers by structural constraints," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49-56, 2010.
13. Bolme D S, Beveridge J R, Draper B A, and Lui Y M, "Visual object tracking using adaptive correlation filters," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2544-2550, 2010.
14. Danelljan M, Hager G, Khan F, and Felsberg M, "Accurate scale estimation for robust visual tracking," *in Proc. of British Machine Vision Conference*, Nottingham, September 1-5, 2014.
15. Wang N, Shi J, Yeung D Y, and Jia J, "Understanding and diagnosing visual tracking systems," *in Proc. of IEEE International Conference on Computer Vision*, pp. 3101-3109, 2015.
16. Ma C, Huang J B, Yang X, and Yang M H, "Hierarchical convolutional features for visual tracking," *in Proc. of IEEE International Conference on Computer Vision*, pp. 3074-3082, 2015.
17. Danelljan M, Hager G, Khan F S, and Felsberg M, "Convolutional features for correlation filter based visual tracking," *in Proc. of IEEE International Conference on Computer Vision Workshops*, pp. 58-66, 2015.
18. Qi Y, Zhang S, Qin L, Yao H, Huang Q, Lim J, and Yang M H, "Hedged deep tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4303-4311, 2016.
19. Bertinetto L, Valmadre J, Golodetz S, Miksik O, and Torr P H S, "Staple: Complementary learners for real-time tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1401-1409, 2016.
20. Danelljan M, Hager G, Khan F S, and Felsberg M, "Learning spatially regularized correlation filters for visual tracking," *in Proc. of IEEE International Conference on Computer Vision*, pp. 4310-4318, 2015.
21. Danelljan M, Bhat G, Khan F S, and Felsberg M, "ECO: Efficient convolution operators for tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6638-6646, 2017.
22. Henriques J F, Caseiro R, Martins P, and Batista J, "Exploiting the circulant structure of tracking-by-detection with kernels," *in Proc. of European Conference on Computer Vision*, pp. 702-715, 2012.
23. Danelljan M, Khan F S, Felsberg M, and Weijer J V, "Adaptive color attributes for real-time visual tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090-1097, 2014.
24. He K, Zhang X, Ren S, and Sun J, "Deep residual learning for image recognition," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.
25. Girshick R, Donahue J, Darrell T, and Malik J, "Rich feature hierarchies for accurate object detection and semantic segmentation," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.
26. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C Y, and Berg A C, "SSD: Single shot multibox detector," *in Proc. of European Conference on Computer Vision*, pp. 21-37, 2016.
27. R. Girshick, "Fast R-CNN," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1440-1448, 2015.
28. Long J, Shelhamer E, and Darrell T, "Fully convolutional networks for semantic segmentation," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015.
29. Hong S, You T, Kwak S, and Han B, "Online tracking by learning discriminative saliency map with convolutional neural network," *in Proc. of International Conference on Machine Learning*, pp. 597-606, 2015.
30. Wang N and Yeung D Y, "Learning a deep compact image representation for visual tracking," *in Proc. of Advances in neural information processing systems*, pp. 809-817, 2013.
31. Wang L, Ouyang W, Wang X, and Lu H, "Visual tracking with fully convolutional networks," *in Proc. of IEEE International Conference on Computer Vision*, pp. 3119-3127, 2015.
32. Bertinetto L, Valmadre J, Henriques J F, Vedaldi A, and Torr P H, "Fully-convolutional siamese networks for object tracking," *in Proc. of European Conference on Computer Vision*, pp. 850-865, 2016.
33. Tao R, Gavves E, and Smeulders A W M, "Siamese instance search for tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1420-1429, 2016.
34. Held D, Thrun S, and Savarese S, "Learning to track at 100 fps with deep regression networks," *in Proc. of European Conference on Computer Vision*, pp. 749-765, 2016.
35. Yan B, Wang D, Lu H, and Yang X, "Cooling-shrinking attack: Blinding the tracker with imperceptible noises," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 990-999, 2020.
36. Guo Q, Xie X, Juefei-Xu F, Ma L, Li Z, Xue W, Feng W, and Liu Y., "Spark: Spatial-aware online incremental attack against visual tracking," *in Proc. of European Conference on Computer Vision*, pp. 202-219, 2020.
37. Jia S, Ma C, Song Y, and Yang X, "Robust tracking against adversarial attacks," *in Proc. of European Conference on Computer Vision*, pp. 69-84, 2020.
38. Liang S, Wei X, Yao S, and Cao X, "Efficient adversarial attacks for visual object tracking," *in Proc. of European Conference on Computer Vision*, pp. 34-50, 2020.
39. Nakka K K and Salzmann M, "Temporally-transferable perturbations: efficient, one-shot adversarial attacks for online visual object trackers," arXiv preprint arXiv:2012.15183, 2020.
40. Valmadre J, Bertinetto L, Henriques J, Vedaldi A, and Torr P H S, "End-to-end representation learning for correlation filter based tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2805-2813, 2017.
41. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, December 2015.
42. Krizhevsky A, Sutskever I, and Hinton G E, "Imagenet classification with deep convolutional neural networks," *in Proc. of Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.
43. Simonyan K and Zisserman A, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
44. Zhang J, Ma S, and Sclaroff S, "MEEM: Robust tracking via multiple experts using entropy minimization," *in Proc. of European Conference on Computer Vision*, pp. 188-203, 2014.
45. Choi J, Chang H J, Yun S, Fischer T, and Demiris Y., "Attentional correlation filter network for adaptive visual tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4807-4816, 2017.
46. Hong Z, Chen Z, Wang C, Mei X, Prokhorov D, and Tao D, "Multi-store tracker (MUSTer): a cognitive psychology inspired approach to object tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 749-758, 2015.
47. Ma C, Yang X, Zhang C, and Yang M H, "Long-term correlation tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5388-5396, 2015.
48. Fan H and Ling H, "Parallel tracking and verifying: a framework for real-time and high accuracy visual tracking," *in Proc. of IEEE International Conference on Computer Vision*, pp. 5486-5494, 2017.
49. Zhang M, Xing J, Gao J, and Hu W, "Robust visual tracking using joint scale-spatial correlation filters," *in Proc. of IEEE International Conference on Image Processing*, pp. 1468-1472, 2015.

50. Mueller M, Smith N, and Ghanem B, "Context-aware correlation filter tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1396-1404, 2017.
51. Lukezic A, Vojir T, Zajc L C, Matas J, and Kristan M, "Discriminative correlation filter with channel and spatial reliability," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6309-6318, 2017.
52. Galoogahi H K, Fagg A, and Lucey S, "Learning background-aware correlation filters for visual tracking," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1135-1143, 2017.
53. Wang Q, Gao J, Zhang M, Xing J, and Hu W, "SPCNet: Scale position correlation network for end-to-end visual tracking," *in Proc. of International Conference on Pattern Recognition*, pp. 1803-1808, 2018.
54. Kristan M, Leonardis A, Matas J, Felsberg M, Pflugfelder R, and Zajc L C. "The visual object tracking VOT2017 challenge results," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1949-1972, 2017.
55. Goyal P, Dollar P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, and He K, "Accurate, large minibatch SGD: training Imagenet in 1 hour," arXiv preprint arXiv:1706.02677, 2017.
56. Boeddeker C, Hanebrink P, Drude L, Heymann J, and Hab-Umbach R, "On the computation of complex valued gradients with application to statistically optimum beamforming," arXiv preprint arXiv:1701.00392, 2017.
57. Vedaldi A and Lenc K, "Matconvnet: Convolutional neural networks for Matlab," *in Proc. of ACM International Conference on Multimedia*, pp. 689-692, 2015.
58. Li A, Lin M, Wu Y, Yang M H, and Yan S, "NUS-PRO: A new visual tracking challenge," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 335-349, 2016.
59. Liang P, Blasch E, and Ling H, "Encoding color information for visual tracking: algorithms and benchmark," *IEEE Trans. on Image Processing*, vol. 24, no. 12, pp. 5630–5644, 2015.
60. Mueller M, Smith N, and Ghanem B "A benchmark and simulator for UAV tracking," *in Proc. of European Conference on Computer Vision*, pp. 445-461, 2016.
61. Danelljan M, Hager G, Khan F S, and Felsberg M, "Discriminative scale space tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1561-1575, 2016.
62. Wang M, Liu Y, and Huang Z, "Large margin object tracking with circulant feature maps," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4021-4029, 2017.
63. Gordon D, Farhadi A, and Fox D, "Re3: Real-time recurrent regression networks for visual tracking of generic objects," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788-795, 2018.
64. Zhang T, Xu C, and Yang M H, "Multi-task correlation particle filter for robust object tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4335-4343, 2017.
65. Gao J, Wang Q, Xing J, Ling H, Hu W, and Maybank S, "Tracking-by-fusion via Gaussian processes regression based transfer learning," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 939-955, April 2020.
66. Chen X, Kang B, Wang D, Li D and Lu H, "Efficient visual tracking via hierarchical cross-attention transformer," *in Proc. of European Conference on Computer Vision Workshops*, pp. 461-477, Feb. 2023.
67. Blatter P, Kanakis M, Danelljan M, and Gool L V, "Efficient visual tracking with exemplar transformers," *in Proc. of IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1571-1581, 2023.
68. Yan B, Peng H, Wu K, Wang D, Fu J, and Lu H, "LightTrack: Finding lightweight neural networks for object tracking via one-shot architecture search", *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 15180-15189, 2021.
69. Danelljan M, Bhat G, Khan F S, and Felsberg M, "ATOM: Accurate tracking by overlap maximization," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4660-4669, 2019.

Weiming Hu received the Ph.D. degree from the department of computer science and engineering, Zhejiang University in 1998. From April 1998 to March 2000, he was a postdoctoral research fellow with the Institute of Computer Science and Technology, Peking University. Now he is a professor in the Institute of Automation, Chinese Academy of Sciences. His research interests are in visual motion analysis, recognition of web objectionable information, and network intrusion detection.

Qiang Wang received the B.S. degree in automation from the University of Science and Technology Beijing, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the Institute of Automation, University of Chinese Academy of Sciences (UCAS). His research interest includes the theory and applications of single object tracking.

Jin Gao Jin Gao received his Bachelor's degree from the Beihang University, Beijing, China in 2010 and the Ph.D. degree from the Institute of Automation, University of Chinese Academy of Sciences (CAS), in 2015. Now he is an associate professor with the National Laboratory of Pattern Recognition, Institute of Automation, CAS. His research interests include visual tracking, augmented reality, semi-supervised learning.

Bing Li received the PhD degree from the Department of Computer Science and Engineering, Beijing Jiaotong University, China, in 2009. Currently, he is a professor in the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. His research interests include color constancy, visual saliency and web content mining.

Stephen Maybank received a BA in Mathematics from King's College Cambridge in 1976 and a PhD in computer science from Birkbeck college, University of London in 1988. Now he is a professor in the Department of Computer Science and Information Systems, Birkbeck College. His research interests include the geometry of multiple images, camera calibration, visual surveillance etc.

Acknowledgements

This work is supported by the national key R&D program of China (No. 2020AAA0105802，2020AAA0105800), the Natural Science Foundation of China (Grant No. 62036011, 62192782, 61721004, U2033210), Beijing Natural Science Foundation (L223003), the Major Projects of Guangdong Education Department for Foundation Research and Applied Research (Grant: 2017KZDXM081, 2018KZDXM066), Guangdong Provincial University Innovation Team Project (Project No.: 2020KCXTD045).

Appendix A: Derivation for DCFNet

We derive the backward formulas for optimizing the CNN parameters θ for DCFNet. We treat the elements in $g_{\boldsymbol{\theta}}(\mathbf{z})$ as variables which are relevant with $\varphi^l(\mathbf{z})$ and $\varphi^l(\mathbf{x})$ and simplify $\partial g_{\boldsymbol{\theta}}(\mathbf{z})$ as ∂g. It is assumed that the channels are independent. By using the chain rule to return the errors to the two branches of the network individually, we can obtain:

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \sum_{l=1}^{D} \frac{\partial L(\boldsymbol{\theta})}{\partial \varphi^l(\mathbf{z})} \frac{\partial \varphi^l(\mathbf{z})}{\partial \boldsymbol{\theta}} + \sum_{l=1}^{D} \frac{\partial L(\boldsymbol{\theta})}{\partial \varphi^l(\mathbf{x})} \frac{\partial \varphi^l(\mathbf{x})}{\partial \boldsymbol{\theta}} + 2\gamma\boldsymbol{\theta} \\ &= \sum_{l=1}^{D} \mathcal{F}^{-1}\left(\frac{\partial L(\boldsymbol{\theta})}{\partial \hat{\varphi}^l(\mathbf{z})^*} \right) \frac{\partial \varphi^l(\mathbf{z})}{\partial \boldsymbol{\theta}} + \sum_{l=1}^{D} \mathcal{F}^{-1}\left(\frac{\partial L(\boldsymbol{\theta})}{\partial \hat{\varphi}^l(\mathbf{x})^*} \right) \frac{\partial \varphi^l(\mathbf{x})}{\partial \boldsymbol{\theta}} + 2\gamma\boldsymbol{\theta}.\end{aligned} \tag{a}$$

According to (a), it is necessary to compute $\partial L / \partial(\varphi^l(\mathbf{z}))$ and $\partial L / \partial(\varphi^l(\mathbf{x}))$. We start with $\partial L / \partial g$. According to (4) and (5) in the main paper,

$$\frac{\partial L}{\partial g} = 2\left(\sum_{l=1}^{D} \mathbf{w}^l \star \varphi^l(\mathbf{x}) - \mathbf{y} \right). \tag{b}$$

The chain rule is a little complicated since the intermediate variables are complex-valued variables. The definitions of the discrete Fourier transform and inverse discrete Fourier transform are used to derive their gradients [29]. Let N be the number of components in vector g. It is apparent that

$$\hat{g}_f = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_n e^{-j\frac{2\pi}{N}nf} = \underset{n \to f}{\mathcal{F}}(g_n), \tag{c}$$

$$g_n = \frac{1}{\sqrt{N}} \sum_{f=0}^{N-1} \hat{g}_f e^{-j\frac{2\pi}{N}nf} = \underset{f \to n}{\mathcal{F}^{-1}}(\hat{g}_f), \tag{d}$$

with g_n as the discrete time signal, $\hat{g}_f$ as the transformed signal in the frequency domain, and n and f having the range $\{0, ..., N\}$. Partial derivatives of $\hat{g}_f$ with respect to g_n and g_n^* are:

$$\begin{cases} \dfrac{\partial \hat{g}_f}{\partial g_n} = \dfrac{1}{\sqrt{N}} e^{-j\frac{2\pi}{N}nf}, \\ \dfrac{\partial \hat{g}_f}{\partial g_n^*} = 0, \end{cases} \tag{e}$$

Partial derivatives of g_n with respect to $\hat{g}_f$ and $\hat{g}_f^*$ are:

$$\begin{cases} \dfrac{\partial g_n}{\partial \hat{g}_f} = \dfrac{1}{\sqrt{N}} e^{j\frac{2\pi}{N}nf}, \\ \dfrac{\partial g_n}{\partial \hat{g}_f^*} = 0. \end{cases} \tag{f}$$

Then, we can obtain the following inferences:

$$\frac{\partial L}{\partial g_n^*} = \sum_{f=0}^{N-1}\left(\left(\frac{\partial L}{\partial \hat{g}_f^*}\right)^* 0 + \frac{1}{\sqrt{N}}\frac{\partial L}{\partial \hat{g}_f^*}\left(e^{-j\frac{2\pi}{N}nf}\right)^*\right) = \frac{1}{\sqrt{N}}\sum_{f=0}^{N-1}\frac{\partial L}{\partial \hat{g}_f^*}e^{j\frac{2\pi}{N}nf} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial \hat{g}_f^*}\right), \tag{g}$$

$$\frac{\partial L}{\partial \hat{g}_f^*} = \sum_{n=0}^{N-1}\left(\left(\frac{\partial L}{\partial \hat{g}_n^*}\right)^* 0 + \frac{1}{\sqrt{N}}\frac{\partial L}{\partial \hat{g}_n^*}\left(e^{j\frac{2\pi}{N}nf}\right)^*\right) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1}\frac{\partial L}{\partial \hat{g}_n^*}e^{-j\frac{2\pi}{N}nf} = \mathcal{F}\left(\frac{\partial L}{\partial \hat{g}_f^*}\right). \tag{h}$$

Since g is the correlation response which is a real-valued vector, it holds that $g = g^*$. The discrete Fourier transform and inverse discrete Fourier transform of the derivatives with respect to g have the following relations:

$$\begin{cases} \hat{g} = \mathcal{F}(g), \\ \dfrac{\partial L}{\partial \hat{g}^*} = \mathcal{F}\left(\dfrac{\partial L}{\partial g^*}\right) = \mathcal{F}\left(\dfrac{\partial L}{\partial g}\right), \\ \dfrac{\partial L}{\partial g} = \dfrac{\partial L}{\partial g^*} = \mathcal{F}^{-1}\left(\dfrac{\partial L}{\partial \hat{g}^*}\right). \end{cases} \tag{i}$$

Since the operations in the forward pass process only contain the element-based Hadamard product and division, we calculate the derivative in (i) per pixel (u, v) in the frequency domain:

$$\frac{\partial L}{\partial \hat{g}_{uv}^*} = \left(\mathcal{F}\left(\frac{\partial L}{\partial g}\right)\right)_{uv}. \tag{j}$$

For the back-propagation of the tracking branch, the partial differential $\partial L / \partial(\varphi^l(\mathbf{z}))$ for $\mathbf{z}$ is required to compute. According to (4) in the main paper,

$$\frac{\partial \hat{g}_{uv}^*(\mathbf{z})}{\partial(\hat{\varphi}_{uv}^l(\mathbf{z}))^*} = \hat{w}_{uv}^l. \tag{k}$$

Then,

$$\frac{\partial L}{\partial(\hat{\varphi}_{uv}^l(\mathbf{z}))^*} = \frac{\partial L}{\partial \hat{g}_{uv}^*}\frac{\partial \hat{g}_{uv}^*(\mathbf{z})}{\partial(\hat{\varphi}_{uv}^l(\mathbf{z}))^*} = \frac{\partial L}{\partial \hat{g}_{uv}^*}\hat{w}_{uv}^l, \tag{l}$$

where $\partial L / \partial \hat{g}_{uv}^*$ is computed using (b) and (j). It holds that

$$\frac{\partial L}{\partial \varphi^l(\mathbf{z})} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial(\hat{\varphi}^l(\mathbf{z}))^*}\right). \tag{m}$$

For the back-propagation of the branch of learning the correlation filter, the partial differential $\partial L / \partial(\varphi^l(\mathbf{x}))$ for $\mathbf{x}$ is required to compute. We compute $\partial L / \partial \hat{\varphi}^l_{uv}(\mathbf{x})$ and $\partial L / \partial(\hat{\varphi}^l_{uv}(\mathbf{x}))^*$ independently, and then combine them to compute $\partial L / \partial(\varphi^l(\mathbf{x}))$. It is apparent that

$$\begin{cases} \dfrac{\partial L}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = \dfrac{\partial L}{\partial \hat{g}^*_{uv}} \dfrac{\partial \hat{g}^*_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})}, \\ \dfrac{\partial \hat{g}^*_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = \dfrac{\partial \hat{g}^*_{uv}}{\partial \hat{\mathbf{w}}^l_{uv}} \dfrac{\partial \hat{\mathbf{w}}^l_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = (\hat{\varphi}^l_{uv}(\mathbf{z}))^* \dfrac{\partial \hat{\mathbf{w}}^l_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})}, \end{cases} \tag{n}$$

where according to (4) in the main paper,

$$\frac{\partial \hat{g}^*_{uv}}{\partial \hat{\mathbf{w}}^l_{uv}} = (\hat{\varphi}^l_{uv}(\mathbf{z}))^*. \tag{o}$$

Let μ be the denominator of (3) in the main paper:

$$\mu = \sum_{k=1}^{D} \hat{\varphi}^k_{uv}(\mathbf{x})(\hat{\varphi}^k_{uv}(\mathbf{x}))^* + \lambda. \tag{p}$$

According to (3) in the main paper,

$$\frac{\partial \hat{\mathbf{w}}^l_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = \frac{\hat{\mathbf{y}}^*_{uv}\mu - \hat{\mathbf{y}}^*_{uv}\hat{\varphi}^l_{uv}(\mathbf{x})(\hat{\varphi}^l_{uv}(\mathbf{x}))^*}{\mu^2}. \tag{q}$$

Substitution of (q) into (n) yields

$$\frac{\partial \hat{g}^*_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = \frac{\hat{\mathbf{y}}^*_{uv}\mu(\hat{\varphi}^l_{uv}(\mathbf{z}))^* - \hat{\mathbf{y}}^*_{uv}\hat{\varphi}^l_{uv}(\mathbf{x})(\hat{\varphi}^l_{uv}(\mathbf{x}))^*(\hat{\varphi}^l_{uv}(\mathbf{z}))^*}{\mu^2}. \tag{r}$$

According to (3) in the main paper,

$$\mu = \frac{\hat{\varphi}^l_{uv}(\mathbf{x})\hat{\mathbf{y}}^*_{uv}}{\hat{\mathbf{w}}^l_{uv}}, \tag{s}$$

and $\mu\hat{\mathbf{w}}^l_{uv} = \hat{\varphi}^l_{uv}(\mathbf{x})\hat{\mathbf{y}}^*_{uv}$. Substitution of (s) into one μ in the denominator of (r) yields

$$\frac{\partial \hat{g}^*_{uv}}{\partial \hat{\varphi}^l_{uv}(\mathbf{x})} = \frac{\hat{\mathbf{y}}^*_{uv}\mu(\hat{\varphi}^l_{uv}(\mathbf{z}))^*\hat{\mathbf{w}}^l_{uv} - \hat{\mathbf{y}}^*_{uv}\hat{\varphi}^l_{uv}(\mathbf{x})(\hat{\varphi}^l_{uv}(\mathbf{x}))^*(\hat{\varphi}^l_{uv}(\mathbf{z}))^*\hat{\mathbf{w}}^l_{uv}}{\mu\hat{\varphi}^l_{uv}(\mathbf{x})\hat{\mathbf{y}}^*_{uv}}. \tag{t}$$

Replacing $\mu\hat{\mathbf{w}}^l_{uv}$ in the numerator of (t) with $\hat{\varphi}^l_{uv}(\mathbf{x})\hat{\mathbf{y}}^*_{uv}$ yields

$$\frac{\partial \hat{g}_{uv}^{*}}{\partial \hat{\varphi}_{uv}^{l}(\mathbf{x})}=\frac{\hat{\mathbf{y}}_{uv}^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\mathbf{y}}_{uv}^{*}-\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\mathbf{y}}_{uv}^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\mathbf{w}}_{uv}^{l}}{\mu\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\mathbf{y}}_{uv}^{*}}$$
$$=\frac{\hat{\mathbf{y}}_{uv}^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}-(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\mathbf{w}}_{uv}^{l}}{\mu}, \tag{u}$$

where $\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\mathbf{y}}_{uv}^{*}$ in the denominator and the numerator is canceled. Substitution of (p) into (u) yields

$$\frac{\partial \hat{g}_{uv}^{*}}{\partial \hat{\varphi}_{uv}^{l}(\mathbf{x})}=\frac{\hat{\mathbf{y}}_{uv}^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}-(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\mathbf{w}}_{uv}^{l}}{\sum_{k=1}^{D}\hat{\varphi}_{uv}^{k}(\mathbf{x})(\hat{\varphi}_{uv}^{k}(\mathbf{x}))^{*}+\lambda}. \tag{v}$$

Substitution of (v) and (j) into the upper part of (n) yields the solution for the partial differential $\partial L/\partial\hat{\varphi}_{uv}^{l}(\mathbf{x})$.

We derive $\partial L/\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}$ below. It is apparent that

$$\begin{cases}\dfrac{\partial L}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}=\dfrac{\partial L}{\partial \hat{g}_{uv}^{*}}\dfrac{\partial \hat{g}_{uv}^{*}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}, \\ \dfrac{\partial \hat{g}_{uv}^{*}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}=\dfrac{\partial \hat{g}_{uv}^{*}}{\partial \hat{\mathbf{w}}_{uv}^{l}}\dfrac{\partial \hat{\mathbf{w}}_{uv}^{l}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}.\end{cases} \tag{w}$$

According to (3) in the main paper,

$$\frac{\partial \hat{\mathbf{w}}_{uv}^{l}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}=-\frac{\hat{\mathbf{y}}^{*}\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\varphi}_{uv}^{l}(\mathbf{x})}{\mu^{2}}. \tag{x}$$

Substitution of (o) and (x) into the lower part of (w) yields

$$\frac{\partial \hat{g}_{uv}^{*}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}=-\frac{\hat{\mathbf{y}}_{uv}^{*}\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\varphi}_{uv}^{l}(\mathbf{x})(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}}{\mu^{2}}. \tag{y}$$

Substitution of (s) into one μ in the denominator of (y) yields

$$\frac{\partial \hat{g}_{uv}^{*}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}=-\frac{\hat{\mathbf{y}}_{uv}^{*}\hat{\varphi}_{uv}^{l}(\mathbf{x})\hat{\varphi}_{uv}^{l}(\mathbf{x})(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\mathbf{w}}_{uv}^{l}}{\mu\hat{\mathbf{y}}_{uv}^{*}\hat{\varphi}_{uv}^{l}(\mathbf{x})}=-\frac{\hat{\varphi}_{uv}^{l}(\mathbf{x})(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\mathbf{w}}_{uv}^{l}}{\mu}, \tag{z}$$

where $\hat{\mathbf{y}}_{uv}^{*}\hat{\varphi}_{uv}^{l}(\mathbf{x})$ in the denominator and the numerator is canceled. Substitution of (p) into (z) yields

$$\frac{\partial \hat{g}_{uv}^{*}}{\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}}=\frac{-\hat{\varphi}_{uv}^{l}(\mathbf{x})(\hat{\varphi}_{uv}^{l}(\mathbf{z}))^{*}\hat{\mathbf{w}}_{uv}^{l}}{\sum_{k=1}^{D}\hat{\varphi}_{uv}^{k}(\mathbf{x})(\hat{\varphi}_{uv}^{k}(\mathbf{x}))^{*}+\lambda}. \tag{A}$$

Substitution of (j) and (A) into the upper part of (w) yields the solution for the partial differential $\partial L/\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}$. Combination of $\partial L/\partial\hat{\varphi}_{uv}^{l}(\mathbf{x})$ and $\partial L/\partial(\hat{\varphi}_{uv}^{l}(\mathbf{x}))^{*}$ yields

$$\frac{\partial L}{\partial \varphi^l(\mathbf{x})} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial(\hat{\varphi}^l(\mathbf{x}))^*} + \left(\frac{\partial L}{\partial \hat{\varphi}^l(\mathbf{x})}\right)^*\right). \tag{B}$$

Appendix B: Derivation for Convolutional-Deconvolutional DCFNet

The parameters θ of the CNN in the convolutional-deconvolutional DCFNet are optimized by minimizing (32) in the main paper. The derivatives of L_{low} in (32) in the main paper are obtained:

$$\frac{\partial L_{low}}{\partial \hat{g}^*(\mathbf{z}_s)} = 2(\hat{g}(\mathbf{z}_s) - \hat{\mathbf{y}}_s). \tag{C}$$

The back-propagation of the tracking branch for search image $\mathbf{z}$ is represented by:

$$\frac{\partial L_{low}}{\partial \varphi^l(\mathbf{z}_s)} = \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{g}^*(\mathbf{z}_s)} \odot \hat{\mathbf{w}}^{l*}\right). \tag{D}$$

For the back-propagation of the learning branch for the target image $\mathbf{x}$, according to (31) in the main paper, the gradient of back-propagation of the filter w is computed by:

$$\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^l} = \sum_{s=1}^{S} \frac{\partial L_{low}}{\partial \hat{g}^* \varphi^l(\mathbf{z}_s)} \odot (\hat{\varphi}^l(\mathbf{z}_s))^*. \tag{E}$$

Let μ be the denominator of $\hat{\mathbf{w}}$ in (30) in the main paper:

$$\mu = \sum_{s=1}^{S} (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\varphi}^l(\mathbf{x}_s^0) + \lambda_1 + \lambda_2 \sum_{i=1}^{k} (\hat{\varphi}^l(\mathbf{x}^i))^* \odot \hat{\varphi}^l(\mathbf{x}^i). \tag{F}$$

We compute the gradients for the object image patch's features $\varphi(\mathbf{x}_s^0)$ and the context image patch's features $\varphi(\mathbf{x}^i)$ as follows: According to (30) in the main paper, the following partial differentials hold:

$$\frac{\partial \hat{\mathbf{w}}^l}{\partial(\hat{\varphi}^l(\mathbf{x}_s^0))^*} = \frac{\mu\hat{\mathbf{y}}_s - \hat{\varphi}^l(\mathbf{x}_s^0) \odot (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{y}}_s}{\mu^2} = \frac{\hat{\mathbf{y}}_s - \hat{\varphi}^l(\mathbf{x}_s^0) \odot \hat{\mathbf{w}}^l}{\mu}, \tag{G}$$

$$\frac{\partial \hat{\mathbf{w}}^l}{\partial \hat{\varphi}(\mathbf{x}_s^0)} = \frac{-\hat{\varphi}^l(\mathbf{x}_s^0)^* \odot \hat{\varphi}^l(\mathbf{x}_s^0)^* \odot \hat{\mathbf{y}}_s}{\mu^2} = \frac{-\hat{\varphi}^l(\mathbf{x}_s^0)^* \odot \hat{\mathbf{w}}^l}{\mu}, \tag{H}$$

$$\frac{\partial \hat{\mathbf{w}}^l}{\partial(\hat{\varphi}^l(\mathbf{x}^i))^*} = \frac{\sum_{s=1}^{S} -(\hat{\varphi}^l(\mathbf{x}^i))^* \odot (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{y}}_s}{\mu^2} = \frac{\sum_{s=1}^{S} -(\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{w}}^l}{\mu}, \tag{I}$$

$$\frac{\partial \hat{\mathbf{w}}^l}{\partial \hat{\varphi}^l(\mathbf{x}^i)} = \frac{\sum_{s=1}^{S} -(\hat{\varphi}^l(\mathbf{x}^i))^* \odot (\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{y}}_s}{\mu^2} = \frac{\sum_{s=1}^{S} -(\hat{\varphi}^l(\mathbf{x}_s^0))^* \odot \hat{\mathbf{w}}^l}{\mu}. \tag{J}$$

Let Re(·) be the real part of a complex-valued matrix. The back-propagation of the learning branch is carried out by:

$$\begin{aligned}\frac{\partial L_{low}}{\partial \varphi^{l}(\mathbf{x}_s^0)} &= \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^{l}}\frac{\partial \hat{\mathbf{w}}}{\partial (\hat{\varphi}^{l}(\mathbf{x}_s^0))^{*}} + \left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^{l}}\frac{\partial \hat{\mathbf{w}}^{l}}{\partial \hat{\varphi}(\mathbf{x}_s^0)}\right)^{*}\right) \\ &= \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^{l}} \odot \frac{\hat{\mathbf{y}}_s^{*} - 2\mathrm{Re}((\hat{\varphi}^{l}(\mathbf{x}_s^0))^{*} \odot \hat{\mathbf{w}}^{l})}{\mu}\right),\end{aligned} \tag{K}$$

$$\begin{aligned}\frac{\partial L_{low}}{\partial \hat{\varphi}^{l}(\mathbf{x}^{i})} &= \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^{l}}\frac{\partial \hat{\mathbf{w}}^{l}}{\partial (\hat{\varphi}^{l}(\mathbf{x}^{i}))^{*}} + \left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^{l}}\frac{\partial \hat{\mathbf{w}}^{l}}{\partial \hat{\varphi}(\mathbf{x}^{i})}\right)^{*}\right) \\ &= \mathcal{F}^{-1}\left(\frac{\partial L_{low}}{\partial \hat{\mathbf{w}}^{l}} \odot \frac{-2\mathrm{Re}((\hat{\varphi}^{l}(\mathbf{x}^{i}))^{*} \odot \hat{\mathbf{w}}^{l})}{\mu}\right).\end{aligned} \tag{L}$$