ROI Pooled Correlation Filters for Visual Tracking

Yuxuan Sun[1], Chong Sun[2], Dong Wang[1]*, You He[3], Huchuan Lu[1,4]
[1]School of Information and Communication Engineering, Dalian University of Technology, China
[2]Tencent Youtu Lab, China
[3]Naval Aviation University, China
[4]Peng Cheng Laboratory, China
rumsyx@mail.dlut.edu.cn, waynecsun@tencent.com, heyou_f@126.com, {wdice,lhchuan}@dlut.edu.cn

Abstract

The ROI (region-of-interest) based pooling method performs pooling operations on the cropped ROI regions for various samples and has shown great success in the object detection methods. It compresses the model size while preserving the localization accuracy, thus it is useful in the visual tracking field. Though being effective, the ROI-based pooling operation is not yet considered in the correlation filter formula. In this paper, we propose a novel ROI pooled correlation filter (RPCF) algorithm for robust visual tracking. Through mathematical derivations, we show that the ROI-based pooling can be equivalently achieved by enforcing additional constraints on the learned filter weights, which makes the ROI-based pooling feasible on the virtual circular samples. Besides, we develop an efficient joint training formula for the proposed correlation filter algorithm, and derive the Fourier solvers for efficient model training. Finally, we evaluate our RPCF tracker on OTB-2013, OTB-2015 and VOT-2017 benchmark datasets. Experimental results show that our tracker performs favourably against other state-of-the-art trackers.

1. Introduction

Visual tracking aims to localize the manually specified target object in the successive frames, and it has been densely studied in the past decades for its broad applications in the automatic drive, human-machine interaction, behavior recognition, etc. Till now, visual tracking is still a very challenging task due to the limited training data and plenty of real-world challenges, such as occlusion, deformation and illumination variations.

In recent years, the correlation filter (CF) has become one of the most widely used formulas in visual tracking for its computation efficiency. The success of the correlation filter mainly comes from two aspects: first, by exploiting the property of circulant matrix, the CF-based algorithms do not need to construct the training and testing samples explicitly, and can be efficiently optimized in the Fourier domain, enabling it to handle more features; second, optimizing a correlation filter can be equivalently converted to solving a system of linear functions, thus the filter weights can either be obtained with the analytic solution (e.g., [9, 8]) or be solved via the optimization algorithms with quadratic convergence [9, 7]. As is well recognized, the primal correlation filter algorithms have limited tracking performance due to the boundary effects and the over-fitting problem. The phenomenon of boundary effects is caused by the periodic assumptions of the training samples, while the over-fitting problem is caused by the unbalance between the numbers of model parameters and the

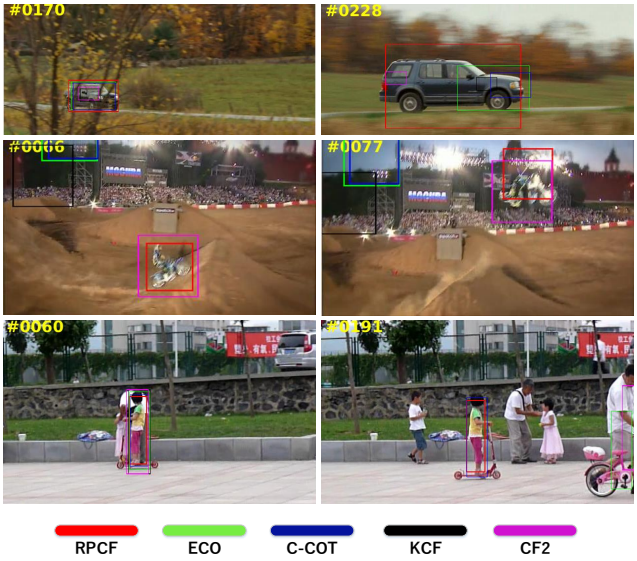


Figure 1. Visualized tracking results of our method and other four competing algorithms. Our tracker performs favourably against the state-of-the-art.

*Corresponding Author: Dr. Wang

training samples. Though the boundary effects have been well addressed in several recent papers (*e.g*., SRDCF [9], DRT [29], BACF [12] and ASRCF [5]), the over-fitting problem is still not paid much attention to and remains to be a challenging research hotspot.

The average/max-pooling operation has been widely used in the deep learning methods via the pooling layer, which is shown to be effective in handling the over-fitting problem and deformations. Currently, two kinds of pooling operations are widely used in deep learning methods. The first one performs average/max-pooling on the entire input feature map and obtains a feature map with reduced spatial resolutions. In the CF formula, the pooling operation on the input feature map can lead to fewer available synthetic training samples, which limits the discriminative ability of the learned filter. Also, the smaller size of the feature map will significantly influence the localization accuracy. However, the ROI (Region of Interest)-based pooling operation is an alternative, which has been successfully embedded into several object detection networks (*e.g*., [14, 26]). Instead of directly performing the average/max-pooling on the entire feature map, the ROI-based pooling method first crops large numbers of ROI regions, each of which corresponds to a target candidate, and then performs average/max-pooling for each candidate ROI region independently. The ROI-based pooling operation has the merits of a pooling operation as mentioned above, and at the same time retains the number of training samples and the spatial information for localization, thus it is meaningful to introduce the ROI-based pooling into the CF formula. Since the CF algorithm has no access to real-world samples, it remains to be investigated on how to exploit the ROI-based pooling in a correlation filter formula.

In this paper, we study the influence of the pooling operation in visual tracking, and propose a novel ROI pooled correlation filters algorithm. Even though the ROI-based pooling algorithm has been successfully applied in many deep learning-based applications, it is seldom considered in the visual tracking field, especially in the correlation filter-based methods. Since the correlation filter formula does not really extract positive and negative samples, it is infeasible to perform the ROI-based pooling like Fast R-CNN [14]. Through mathematical derivation, we provide an alternative solution to implement the ROI-based pooling. We propose a correlation filter algorithm with equality constraints, through which the ROI-based pooling can be equivalently achieved. We propose an Alternating Direction Method Of Multipliers (ADMM) algorithm to solve the optimization problem, and provide an efficient solver in the Fourier domain. Large number of experiments on the OTB-2013 [31], OTB-2015 [32] and VOT-2017 [20] datasets validate the effectiveness of the proposed method (see Figure 1 and Section 5). The contributions of this paper are three-fold:

- This paper is the first attempt to introduce the idea of ROI-based pooling in the correlation filter formula. It proposes a correlation filter algorithm with equality constraints, through which the ROI-based pooling operation can be equivalently achieved without the need for real-world ROI sample extraction. The learned filter weights are insusceptible to the over-fitting problem and are more robust to deformations.
- This paper proposes a robust ADMM method to optimize the proposed correlation filter formula in the Fourier domain. With the computed Lagrangian multipliers, the paper aims to use the conjugate gradient method for filter learning, and develops efficient optimization strategy for each step.
- This paper conducts large amounts of experiments on three available public datasets. The experimental results validate the effectiveness of the proposed method.

2. Related Work

The recent papers on visual tracking are mainly based on the correlation filters and deep networks [21], many of which have impressive performance. In this section, we primarily focus on the algorithms based on the correlation filters and briefly introduce related issues of the pooling operations.

Discriminative Correlation Filters. Trackers based on correlation filters have been the focus of researchers in recent years, which have achieved the top performance in various datasets. The correlation filter algorithm in visual tracking can be dated back to the MOSSE tracker [2], which takes the single-channel gray-scale image as input. Even though the tracking speed is impressive, the accuracy is not satisfactory. Based on the MOSSE tracker, Henriques *et al*. advance the state-of-the-art by introducing the kernel functions [18] and higher dimensional features [19]. Ma *et al*. [24] exploit the rich representation information of deep features in the correlation filter formula, and fuse the responses of various convolutional features via a coarse-to-fine searching strategy. Qi *et al*. [25] extend the work of [24] by exploiting the Hedge method to learn the importance for each kind of feature adaptively. Apart from the MOSSE tracker, the aforementioned algorithms learn the filter weights in the dual space, which have been attested to be less effective than the primal space-based algorithms [8, 9, 19]. However, correlation filters learned in the primal space are severely influenced by the boundary effects and the over-fitting problem. Because of this, Danelljan *et al*. [9] introduce a weighted regularization constraint on the learned filter weights, encouraging the algorithm to learn more weights on the central region of the target object. The SRDCF tracker [9] has become a baseline algorithm for many latter trackers, *e.g*., CCOT [11] and SRD-

CFDecon [10]. The BACF tracker [12] provides another feasible way to address the boundary effects, which generates real-world training samples and greatly improves the discriminant power of the learned filter. Though the above methods have well addressed the boundary effects, the over-fitting problem is rarely considered. The ECO tracker [7] jointly learns a projection matrix and the filter weights, through which the model size is greatly compressed. Different from the ECO tracker, our method introduces the ROI-based pooling operation into a correlation filter formula, which does not only address the over-fitting problem but also makes the learned filter weights more robust to deformations.

Pooling Operations. The idea of the pooling operation has been used in various fields in computer vision, *e.g.*, feature extraction [6, 22], convolutional neural networks [27, 16], to name a few. Most of the pooling operations are performed on the entire feature map to either obtain more stable feature representations or rapidly compress the model size. In [6], Dalal *et al*. divide the image window into dozens of cells, and compute the histogram of gradient directions in each divided cell. The computed feature representations are more robust than the ones based on individual pixels. In most deep learning-based algorithms (*e.g.*, [6, 22]), the pooling operations are performed via a pooling layer, which accumulates the multiple response activations over a small neighbourhood region. The localization accuracy of the network usually decreases after the pooling operation. Instead of the primal max/average-pooling layer, the faster R-CNN method [14] exploits the ROI pooling layer to ensure the localization accuracy and at the same time compress the model size. The method firstly extracts the ROI region for each candidate target object via a region of proposal network (RPN), and then performs the max-pooling operation on the ROI region to obtain more robust feature representations. Our method is inspired by the ROI pooling proposed in [14], and is the first attempt to introduce the ROI-based pooling operation into the correlation filter formula.

3. Correlation Filter and Pooling

In this section, we briefly revisit the two key technologies closely related to our approach (*i.e.*, the correlation filter and pooling operation).

3.1. Revisit of Correlation Filter

To help better understand our method, we first introduce the primal correlation filter algorithm. Given an input feature map, a correlation filter algorithm aims at learning a set of filter weights to regress the Gaussian-shaped response. We use $y_d \in \mathbb{R}^N$ to denote the desired Gaussian-shaped response, and x to denote the input feature map with D feature channels $x_1, x_2, ..., x_D$. For each feature channel

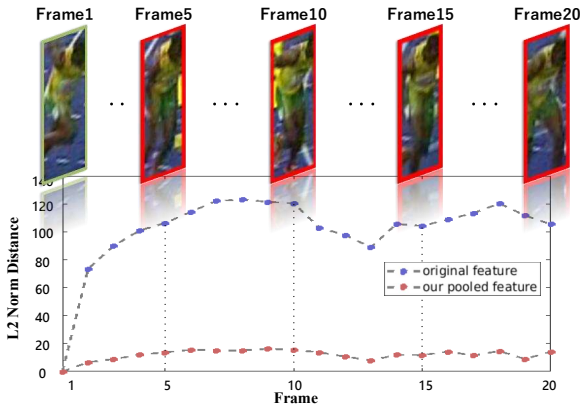


Figure 2. Illustration showing that ROI pooled features are more robust to target deformations than the original ones. For both features, we compute the ℓ_2 loss between features extracted from Frames 2-20 and Frame 1, and visualize the distances via red and blue dots respectively.

$x_d \in \mathbb{R}^N$, a correlation filter algorithm computes the response by convolving x_d with the filter weight $w_d \in \mathbb{R}^N$. Based on the above-mentioned definitions and descriptions, the optimal filter weights can be obtained by optimizing the following objective function:

$$E(w) = \frac{1}{2}\left\| y - \sum_{d=1}^{D} w_d * x_d \right\|_2^2 + \frac{\lambda}{2}\sum_{d=1}^{D}\|w_d\|_2^2, \quad (1)$$

where $*$ denotes the circular convolution operator, $w = [w_1, w_2, ..., w_D]$ is concatenated filter vector, λ is a trade-off parameter to balance the importance between the regression and the regularization losses. According to the Parseval's theorem, Eq. 1 can be equivalently written in the Fourier domain as

$$E(\hat{w}) = \frac{1}{2}\left\| \hat{y} - \sum_{d=1}^{D} \hat{w}_d \odot \hat{x}_d \right\|_2^2 + \frac{\lambda}{2}\sum_{d=1}^{D}\|\hat{w}_d\|_2^2, \quad (2)$$

where $\odot$ is the Hadamard product. We use $\hat{y}$, $\hat{w}_d$, $\hat{x}_d$ to denote the Fourier domain of vector y, w_d and x_d.

3.2. Pooling Operation in Visual Tracking

As is described by many deep learning methods [27, 13], the pooling layer plays a crucial rule in addressing the over-fitting problem. Generally speaking, a pooling operation tries to fuse the neighbourhood response activations into one, through which the model parameters can be effectively compressed. In addition to addressing the over-fitting problem, the pooled feature map becomes more robust to deformations (Figure 2). Currently, two kinds of pooling operations are widely used, *i.e.*, the pooling operation based on the entire feature map (*e.g.*, [27, 16]) and the pooling operation based on the candidate ROI region (*e.g.* [26]). The

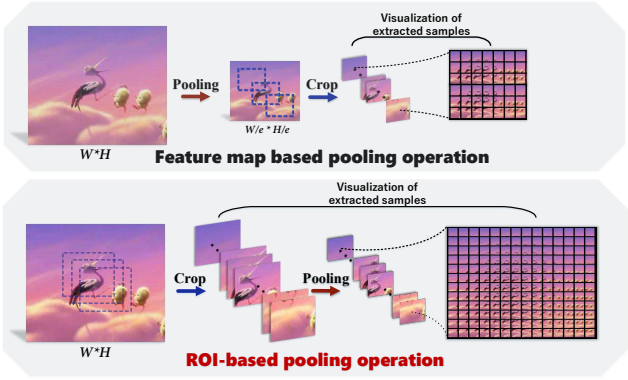


Figure 3. Illustration showing the difference between the feature map based and the ROI-based pooling operations. For clarity, we use 8 as the stride for sample extraction on the original image. This corresponds to a stride = 2 feature extraction in the HOG feature with 4 as the cell size. The pooling kernel size is set as e = 2 in this example.

former one has been widely used in the CF trackers with deep features, as a contrast, the ROI-based pooling operation is seldom considered. As is described in Section 1, directly performing average/max-pooling on the input feature map will result in fewer training/testing samples and worse localization accuracy. We use an example to show how different pooling methods influence the sample extraction process in Figure 3, wherein the extracted samples are visualized on the right-hand side. For simplicity, this example is based on the dense sampling process. The conclusion is also applicable to the correlation filter method, which is essentially trained via densely sampled circular candidates. In the feature map based pooling operation, the feature map size is first reduced to $W/e \times H/e$, thus leading to fewer samples. However, the ROI-based pooling first crop samples from the $W \times H$ feature map and then performs pooling operations upon them, thus does not influence the training number. Fewer training samples will lead to inferior discrimination ability of the learned filter, while fewer testing samples will result in inaccurate target localizations. Thus, it is meaningful to introduce the ROI-based pooling operation into the correlation filter algorithms. Since the max-pooling operation will introduce the non-linearity that makes the model intractable to be optimized, the ROI-based average-pooling operation is preferred in this paper.

4. Our Approach

4.1. ROI Pooled Correlation Filter

In this section, we propose a novel correlation tracking method with ROI-based pooling operation. Like the previous methods [18, 11], we introduce our CF-based tracking algorithm in the one-dimensional domain, and the conclusions can be easily generalized to higher dimensions. Since the correlation filter does not explicitly extract the training samples, it is impossible to perform the ROI-based pooling operation following the pipeline in Figure 3. In this paper, we derive that the ROI-based pooling operation can be implemented by adding additional constraints on the learned filter weights.

Given a candidate feature vector v corresponding to the target region with L elements, we perform the average-pooling operation on it with the pooling kernel size e. For simplicity, we set $L = eM$, where M is a positive integer (the padding operation can be used if L cannot be divided by e evenly). The pooled feature vector $v' \in \mathbb{R}^M$ can be computed as $v' = \frac{1}{e}Uv$, where the matrix $U \in \mathbb{R}^{M \times Me}$ is constructed as:

$$U = \begin{bmatrix} \mathbf{1}^e & \mathbf{0}^e & \cdots & \mathbf{0}^e & \mathbf{0}^e \\ \mathbf{0}^e & \mathbf{1}^e & \cdots & \mathbf{0}^e & \mathbf{0}^e \\ \vdots & \vdots & \ddots & \mathbf{0}^e & \mathbf{0}^e \\ \mathbf{0}^e & \mathbf{0}^e & \cdots & \mathbf{1}^e & \mathbf{0}^e \\ \mathbf{0}^e & \mathbf{0}^e & \cdots & \mathbf{0}^e & \mathbf{1}^e \end{bmatrix}, \tag{3}$$

where $\mathbf{1}^e \in \mathbb{R}^{1 \times e}$ denotes a vector with all the entries set as 1, and $\mathbf{0}^e \in \mathbb{R}^{1 \times e}$ is a zero vector. Based on the pooled vector, we compute the response as:

$$r = w'^{\top} v' = w'^{\top} Uv/e = \left(U^{\top} w'\right)^{\top} v/e, \tag{4}$$

wherein w' is the weight corresponding to the pooled feature vector, $U^{\top} w' = [w'(1)\mathbf{1}^e, w'(2)\mathbf{1}^e, ..., w'(M)\mathbf{1}^e]^{\top}$. It is easy to conclude that average-pooling operation can be equivalently achieved by constraining the filter weights in each pooling kernel to have the same value. Based on the discussions above, we define our ROI pooled correlation filter as follows:

$$\begin{aligned} E(w) = \tfrac{1}{2} \left\| y - \sum_{d=1}^{D} (p_d \odot w_d) * x_d \right\|_2^2 + \tfrac{\lambda}{2} \sum_{d=1}^{D} \|g_d \odot w_d\|_2^2 \\ \text{s.t.} \quad w_d(i_\eta) = w_d(j_\eta), \quad (i_\eta, j_\eta) \in \mathcal{P}, \eta = 1, .., K \end{aligned} \tag{5}$$

where we consider K equality constraints to ensure that filter weights in each pooling kernel have the same value, $\mathcal{P}$ denotes the set that two filter elements belong to the same pooling kernel, i_η and j_η denote the indexes of elements in weight vector w_d. In Eq. 5, $p_d \in \mathbb{R}^N$ is a binary mask which crops the filter weights corresponding to the target region. By introducing p_d, we make sure that the filter only has the response for the target region of each circularly constructed sample [12]. The vector $g_d \in \mathbb{R}^N$ is a regularization weight that encourages the filter to learn more weights in the central part of the target object. The idea to introduce p_d and g_d has been previously proposed in [9, 12], while our tracker is the first attempt to integrate them. In the equality constraints, we consider the relationships between two arbitrary weight elements in a pooling kernel,

thus $K = \frac{e!}{(e-2)!2!}(\lfloor (L-e)/e \rfloor + 1)$ for each channel d, where L is the number of nonzero values in p_d. Note that the constraints are only performed in the filter coefficients corresponding to the target region of each sample, and the computed K is based on the one-dimensional case.

According to the Parseval's formula, the optimization in Eq. 5 can be equivalently written as:

$$E(\hat{w}) = \frac{1}{2}\left\|\hat{y} - \sum_{d}^{D} \hat{P}_d \hat{w}_d \odot \hat{x}_d\right\|_2^2 + \frac{\lambda}{2}\sum_{d=1}^{D}\left\|\hat{G}_d w_d\right\|_2^2, \quad \text{s.t. } V_d^1 \mathcal{F}_d^{-1}\hat{w}_d = V_d^2 \mathcal{F}_d^{-1}\hat{w}_d \tag{6}$$

where $\mathcal{F}_d$ denotes the Fourier transform matrix, and $\mathcal{F}_d^{-1}$ denotes the inverse transform matrix. The vectors $\hat{p}_d \in \mathbb{C}^{N\times 1}$, $\hat{y} \in \mathbb{C}^{N\times 1}$, $\hat{x}_d \in \mathbb{C}^{N\times 1}$ and $\hat{w}_d \in \mathbb{C}^{N\times 1}$ denote the Fourier coefficients of the corresponding signal vectors y, x_d, p_d and w_d. Matrices $\hat{P}_d$ and $\hat{G}_d$ are the Toeplitz matrices, whose (i,j)-th elements are $\hat{p}_d((N+i-j)\%N+1)$ and $\hat{g}_d((N+i-j)\%N+1)$, where $\%$ denotes the modulo operation. They are constructed based on the convolution theorem to ensure that $\hat{P}_d\hat{w}_d = \hat{p}_d * \hat{w}_d$, $\hat{G}_d w_d = \hat{g}_d * \hat{w}_d$. Since the discrete Fourier coefficients of a real-valued signal are Hermitian symmetric, *i.e.*, $\hat{p}_d((N+i-j)\%N+1) = \hat{p}_d((N+j-i)\%N+1)^*$ in our case, we can easily conclude that $\hat{P}_d = \hat{P}_d^H$ and $\hat{G}_d = \hat{G}_d^H$, where H denotes the conjugate-transpose of a complex matrix. In the constraint term, $V_d^1 \in \mathbb{R}^{K\times N}$ and $V_d^2 \in \mathbb{R}^{K\times N}$ are index matrices with either 1 or 0 as the entries, $V_d^1\mathcal{F}_d^{-1}\hat{w}_d = [w_d(i_1), ..., w_d(i_K)]^\top$ and $V_d^2\mathcal{F}_d^{-1}\hat{w}_d = [w_d(j_1), ..., w_d(j_K)]^\top$.

Eq. 6 can be rewritten in a compact formula as:

$$E(\hat{w}) = \frac{1}{2}\left\|\hat{y} - \sum_{d=1}^{D} \hat{E}_d \hat{w}_d\right\|_2^2 + \frac{\lambda}{2}\sum_{d=1}^{D}\left\|\hat{G}_d\hat{w}_d\right\|_2^2, \quad \text{s.t.} \quad V_d\mathcal{F}_d^{-1}\hat{w}_d = \mathbf{0} \tag{7}$$

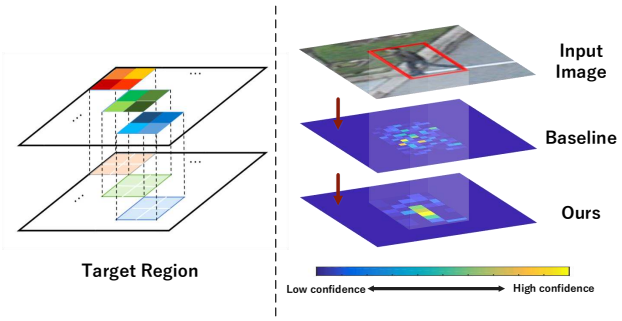
where $\hat{E}_d = \hat{X}_d\hat{P}_d$, $\hat{X}_d = \mathrm{diag}(\hat{x}_d(1), ..., \hat{x}_d(N))$ is a diagonal matrix, $V_d = V_d^1 - V_d^2$.

4.2. Model Learning

Since Eq. 7 is a quadratic programming problem with linear constraints, we use the Augmented Lagrangian Method for efficient model learning. The Lagrangian function corresponding to Eq. 7 is defined as:

$$\mathcal{L}(\hat{w}, \xi) = \frac{1}{2}\left\|\hat{y} - \sum_{d=1}^{D} \hat{E}_d\hat{w}_d\right\|_2^2 + \frac{\lambda}{2}\sum_{d=1}^{D}\left\|\hat{G}_d\hat{w}_d\right\|_2^2 + \sum_{d=1}^{D}\xi_d^\top V_d\mathcal{F}_d^{-1}\hat{w}_d + \frac{1}{2}\sum_{d=1}^{D}\gamma_d\left\|V_d\mathcal{F}_d^{-1}\hat{w}_d\right\|_2^2, \tag{8}$$

where $\xi_d \in \mathbb{R}^K$ denotes the Lagrangian multipliers for the d-th channel, γ_d is the penalty parameter, $\xi = [\xi_1^\top, ..., \xi_D^\top]^\top$. The ADMM method is used to alternately optimize $\hat{w}$ and ξ.



(a) (b)

Figure 4. Comparison between filter weights of the baseline method (*i.e.*, the correlation filter algorithm without ROI-based pooling) and the proposed method. (a) A toy model showing that our learned filter elements are identical in each pooling kernel. (b) Visualizations of the filter weights learned by the baseline and our method. Our algorithm learns more compact filter weights than the baseline method, and thus can better address the over-fitting problem.

Though the optimization objective function is non-convex, it becomes a convex function when either $\hat{w}$ or ξ is fixed.

When ξ is fixed, $\hat{w}$ can be computed via the conjugate gradient descent method [4]. We compute the gradient of the objective function with respects to $\hat{w}_d$ in Eq. 8 and obtain a number of linear equations by setting the gradient to be a zero vector:

$$(\hat{A} + \mathcal{F}\overline{V}^\top\overline{V}\mathcal{F}^{-1} + \lambda\hat{G}^H\hat{G})\hat{w} = \hat{E}^H y - \mathcal{F}V^\top\xi, \tag{9}$$

where $\mathcal{F} \in \mathbb{C}^{DN\times DN}$, $\hat{G} \in \mathbb{C}^{DN\times DN}$, $V \in \mathbb{R}^{DK\times DN}$ and $\overline{V} \in \mathbb{R}^{DK\times DN}$ are block diagonal matrices with the d-th matrix block set as $\mathcal{F}_d$, $\hat{G}_d$, V_d and $\sqrt{\gamma_d}V_d$, $E = [E_1, E_2, ..., E_D]$, $\hat{A} = E^H E$. In the conjugate gradient method, the computation load lies in the three terms $\hat{A}\hat{u}$, $\mathcal{F}\overline{V}^\top\overline{V}\mathcal{F}^{-1}\hat{u}$ and $\lambda\hat{G}^H\hat{G}\hat{u}$ given the search direction $\hat{u} = [u_1^\top, ..., u_D^\top]^\top$. In the following, we present more details on how we compute these three terms efficiently. Each of the three terms can be regarded as a vector constructed with D sub-vectors. The d-th sub-vector of $\hat{A}\hat{u}$ is computed as $\hat{P}_d^H X_d^H \sum_{j=1}^{D}\hat{X}_j(\hat{P}_j\hat{u}_j)$ wherein $P_d^H = P_d$ as described above. Since the Fourier coefficients of p_d (a vector with binary values) are densely distributed, it is time consuming to directly compute $\hat{P}_d\hat{v}$ given an arbitrary complex vector $\hat{v}$. In this work, the convolution theorem is used to efficiently compute $\hat{P}_d\hat{v}$. The d-th sub-vector of the second term is $\mathcal{F}_d\overline{V}_d{}^\top\overline{V}_d u_d = \gamma_d\mathcal{F}_d V_d{}^\top V_d u_d$. As the matrices V_d and $V_d^\top$ only consists of 1 and -1, thus the computation of $V_d^\top V_d u_d$ can be efficiently conducted via table lookups. The third term corresponds to the convolution operation, whose convolution kernel is usually smaller than 5, thus it can also be efficiently computed.

When $\hat{w}$ is computed, ξ_d can be updated via:

$$\xi_d^{i+1} = \xi_d^i + \gamma_d V_d \mathcal{F}_d^{-1} \hat{w}_d, \tag{10}$$

where we use ξ_d^i to denote the value of ξ_d in the i-th iteration. According to [3], the value of γ_d can be updated as:

$$\gamma_d^{i+1} = \min(\gamma_{\max}, \alpha \gamma_d^i), \tag{11}$$

again we use i to denote the iteration index.

4.3. Model Update

To learn more robust filter weights, we update the proposed RPCF tracker based on several training samples (T samples in total) like [11, 7]. We extend the notations $\hat{A}$ and $\hat{E}$ in Eq. 9 with superscript t, and reformulate Eq. 9 as follows:

$$(\sum_{t=1}^{T} \mu_t \hat{A}^t + \mathcal{F} V^\top V \mathcal{F}^{-1} + \lambda \hat{G}^H \hat{G}) \hat{w} = b, \tag{12}$$

where $b = \sum_{t=1}^{T} \mu_t (\hat{E}^t)^H y - \mathcal{F} V^\top \xi$, and μ_t denotes the importance weight for each training sample t. Most previous correlation filter trackers update the model iteratively via a weighted combination of the filter weights in various frames. Different from them, we exploit the sparse update mechanism, and update the model every N_t frames [7]. In each updating frame, the conjugate gradient descent method is used, and the search direction of the previous update process is input as a warm start. Our training samples are generated following [7], and the weight (*i.e.*, learning rate) for the newly added sample is set as ω, while the weights of previous samples are decayed by multiplying $1-\omega$. In Figure 4, we visualize the learned filter weights of different trackers with and without ROI-based pooling, our tracker can learn more compact filter weights and focus on the reliable regions of the target object.

4.4. Target Localization

In the target localization process, we first crop the candidate samples with different scales, *i.e.*, $x_d^s, s \in \{1, ..., S\}$. Then, we compute the response $\hat{r}^s$ for the feature in each scale in the Fourier domain:

$$\hat{r}^s = \sum_{d=1}^{D} \hat{x}_d^s \hat{w}_d. \tag{13}$$

The computed responses are then interpolated with trigonometric polynomial following [9] to achieve the sub-pixel target localization.

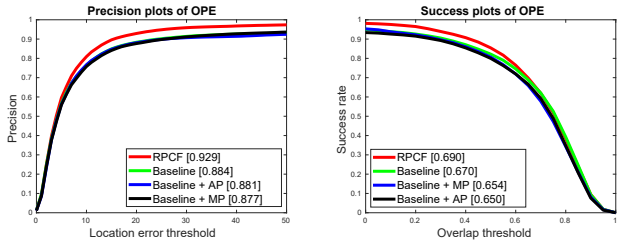
5. Experiments

In this section, we evaluate the proposed RPCF tracker on the OTB-2013 [31], OTB-2015 [32] and VOT2017 [20] datasets. We first evaluate the effectiveness of the method, and then further compare our tracker with the recent state-of-the-art.

5.1. Experimental Setups

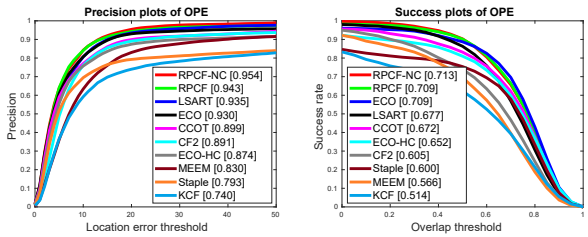
Implementation Details. The proposed RPCF method is mainly implemented in MATLAB on a PC with an i7-4790K CPU and a Geforce 1080 GPU. Similar to the ECO method [7], we use a combination of CNN features from two convolution layers, HOG and color names for target representation. For efficiency, the PCA method is used to compress the features. We set the learning rate ω, the maximum number of training samples T, $\gamma_{\max}$ and α as 0.02, 50, 1000 and 10 respectively, and we update the model in every N_t frame. As to γ_d, we set a relative small value γ_1 (*e.g.*, 0.1) for the high-level feature (*i.e.*, the second convolution layer), and a larger value $\gamma_2 = 3\gamma_1$ for the other feature channels. The kernel size e is set as 2 in the implementation. We use the conjugate gradient descent for model initialization and update, 200 iterations are used in the first frame, and the following update frame uses 6 iterations. Our tracker runs at about 5fps without optimization.

Evaluation Metric. We follow the one-pass evaluation (OPE) rule on the OTB-2013 and OTB-2015 datasets, and report the precision plots as well as the success plots for the performance measure. The success plots demonstrate the overlaps between tracked bounding boxes and ground truth with varying thresholds, while the precision plots measure the accuracy of the estimated target center positions. In the precision plots, we exploit the distance precision (DP) rate at 20 pixels for the performance report, while we exploit the area-under-curve (AUC) score for performance report in success plots. On the VOT-2017 dataset, we evaluate our tracker in terms of the Expected Average Overlap (EAO), accuracy raw value (A) and robustness raw value (R) measure the overlap, accuracy and robustness respectively.



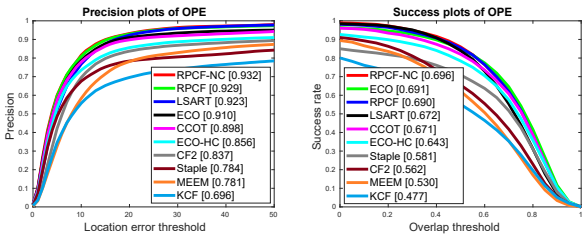
(a) (b)

Figure 5. Precision and success plots of 100 sequences on the OTB-2015 dataset. The distance precision rate at the threshold of 20 pixels and the AUC score for each tracker is presented in the legend.



(a) (b)

Figure 6. Precision and success plots of 50 sequences on the OTB-2013 dataset. The distance precision rate at the threshold of 20 pixels and the AUC score for each tracker is presented in the legend.



(a) (b)

Figure 7. Precision and success plots of 100 sequences on the OTB-2015 dataset. The distance precision rate at the threshold of 20 pixels and the AUC score for each tracker is presented in the legend.

5.2. Ablation Study

In this subsection, we conduct experiments to validate the contributions of the proposed RPCF method. We set the tracker that does not consider the pooling operation as the baseline method, and use Baseline to denote it. It essentially corresponds to Eq. 5 without equality constraints. To validate the superiority of our ROI-based pooling method over feature map based average-pooling and max-pooling, we also implement the trackers that directly performs average-pooling and max-pooling on the input feature map, which are named as Baseline+AP and Baseline+MP.

We first compare the Baseline method with Baseline+AP and Baseline+MP, which shows that the tracking performance decreases when feature map based pooling operations are performed. Directly performing pooling operations on the input feature map will not only influence the extraction of the training samples but also lead to worse target localization accuracy. In addition, the over-fitting problem is not well addressed in such methods since the ratio between the numbers of model parameters and available training samples do not change compared with the Baseline method. We validate the effectiveness of the proposed method by comparing our RPCF tracker with the Baseline method. Our tracker improves the Baseline method by 4.4% and 2.0% in precision and success plots respectively. By exploiting the ROI-based pooling operations , our learned filter weights are insusceptible to the over-fitting problem and are more robust to deformations.

5.3. State-of-the-art Comparisons

OTB-2013 Dataset. The OTB-2013 dataset contains 50 videos annotated with 11 various attributes including illumination variation, scale variation, occlusion, deformation and so on. We evaluate our tracker on this dataset and compare it with 8 state-of-the-art methods that are respectively ECO [7], CCOT [11], LSART [28], ECO-HC [7], CF2 [24], Staple [1], MEEM [33] and KCF [19]. We demonstrate the precision and success plots for different trackers in Figure 6.

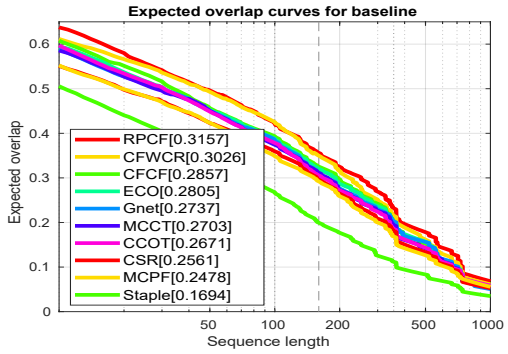


Figure 8. Expected Average Overlap (EAO) curve for 10 state-of-the-art trackers on the VOT-2017 dataset.

Our RPCF method has a 94.3% DP rate at the threshold of 20 pixels and a 70.9% AUC score. Compared with other correlation filter based trackers, the proposed RPCF method has the best performance in terms of both precision and success plots. Our method improves the second best tracker ECO by 1.9% in terms of DP rates, and has comparable performance according to the success plots. When the features are not compressed via PCA, the tracker (denoted as RPCF-NC) has a 95.4% DP rate at the threshold of 20 pixels and a 71.3% AUC score in success plots, and it runs at 2fps without optimization.

OTB-2015 Dataset. The OTB-2015 dataset is an extension of the OTB-2013 dataset and contains 50 more video sequences. On this dataset, we also compare our tracker with the above mentioned 8 state-of-the-art trackers, and present the results in Fiugre 7(a)(b). Our RPCF tracker has a 92.9% DP rate and a 69.0% AUC score. It improves the second best tracker ECO by 1.9% in terms of the precision plots. With the non-compressed features, our RPCF-NC tracker achieves the 93.2% DP rate and 69.6% AUC score, which again has the best performance among all the compared trackers.

The OTB-2015 dataset divides the image sequences into 11 attributes, each of which corresponds to a challenging factor. We compare our RPCF tracker against other 8 state-

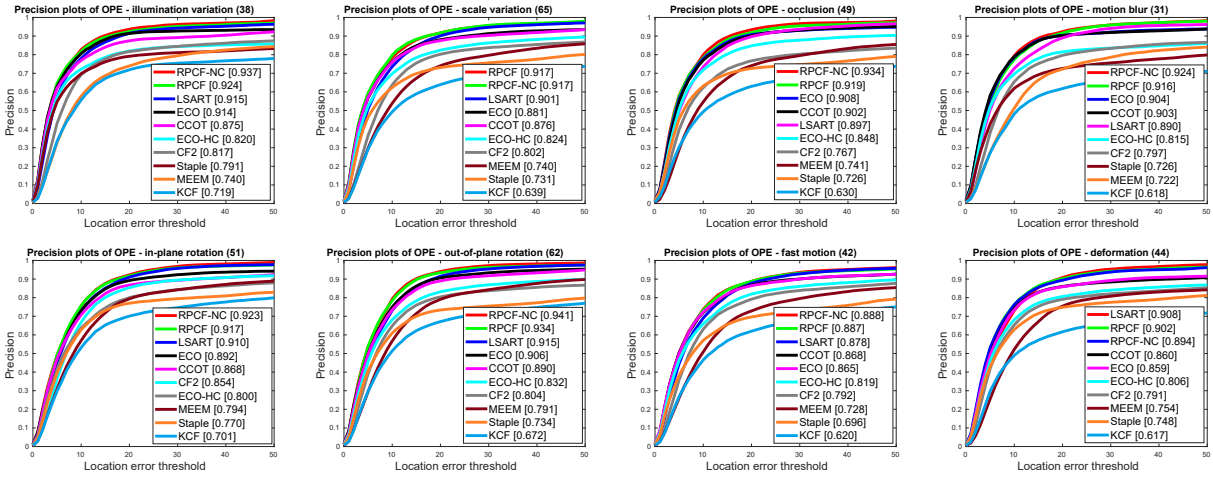


Figure 9. Precision plots of different algorithms on 8 attributes, which are respectively illumination variation, scale variation, occlusion, motion blur, in-plane rotation, out-of-plane rotation, fast motion and deformation.

Table 1. Performance evaluation for 10 state-of-the-art algorithms on the VOT-2017 public dataset. The best three results are marked in red, blue and green fonts, respectively.

	RPCF	CFWCR	CFCF	ECO	Gnet	MCCT	CCOT	CSR	MCPF	Staple
EAO	0.316	0.303	0.286	0.281	0.274	0.270	0.267	0.256	0.248	0.169
A	0.500	0.484	0.509	0.483	0.502	0.525	0.494	0.491	0.510	0.530
R	0.234	0.267	0.281	0.276	0.276	0.323	0.318	0.356	0.427	0.688

of-the-art trackers and present the precision plots for different trackers in Figure 9. As is illustrated in the figure, our RPCF tracker has good tracking performance in all the listed attributes. Especially, the RPCF tracker improves the ECO method by 3.6%, 2.5%, 2.8%, 2.2% and 4.3% in the attributes of scale variation, in-plane rotation, out-of-plane rotation, fast motion and deformation. The ROI pooled features become more consistent across different frames than the original ones, which contributes to robust target representation when the target appearance dramatically changes (see Figure 2 for example). In addition, by exploiting the ROI-based pooling operations, the model parameters are greatly compressed, which makes the proposed tracker insusceptible to the over-fitting problem. In Figure 9, we also present the results of our RPCF-NC tracker for reference.

VOT-2017 Dataset. We test the proposed tracker on the VOT-2017 dataset for more thorough performance evaluations. The VOT-2017 dataset consists of 60 sequences with 5 challenging attributes, *i.e.*, occlusion, illumination change, motion change, size change, camera motion. Different from the OTB-2013 and OTB-2015 datasets, it focuses on evaluating the short-term tracking performance and introduces a reset based experiment setting. We compare our RPCF tracker with 9 state-of-the-art trackers including CFWCR [17], ECO [7], CCOT [11], MCCT [30], CFCF [15], CSR [23], MCPF [34], Gnet [20] and Staple [1]. The tracking performance of different trackers in terms of EAO, A and R are provided in Table 1 and Figure 8. Among all the compared trackers, our RPCF method has a 31.6% EAO score which improves the ECO method by 3.5%. Also, our tracker has the best performance in terms of robustness measure among all the compared trackers.

6. Conclusion

In this paper, we propose the ROI pooled correlation filters for visual tracking. Since the correlation filter algorithm does not extract real-world training samples, it is infeasible to perform the pooling operation for each candidate ROI region like the previous methods. Based on the mathematical derivations, we provide an alternative solution for the ROI-based pooling with the circularly constructed virtual samples. Then, we propose a correlation filter formula with equality constraints, and develop an efficient ADMM solver in the Fourier domain. Finally, we evaluate the proposed RPCF tracker on OTB-2013, OTB-2015 and VOT-2017 benchmark datasets. Extensive experiments demonstrate that our method performs favourably against the state-of-the-art algorithms on all the three datasets.

Acknowledgement. This paper is supported in part by National Natural Science Foundation of China #61725202, #61829102, #61872056 and #61751212, and in part by the Fundamental Research Funds for the Central Universities under Grant #DUT18JC30. This work is also sponsored by CCF-Tencent Open Research Fund.

References

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

[2] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

[4] Angelika Bunse-Gerstner and Ronald Stöver. On a conjugate gradient-type method for solving complex symmetric linear systems. *Linear Algebra and its Applications*, 287(1-3):105–123, 1999.

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

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, Michael Felsberg, et al. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017.

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

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

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

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

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

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[14] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[15] Erhan Gundogdu and A Aydın Alatan. Good features to correlate for visual tracking. *IEEE Transactions on Image Processing*, 27(5):2526–2540, 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[17] Zhiqun He, Yingruo Fan, Junfei Zhuang, Yuan Dong, and HongLiang Bai. Correlation filters with weighted convolution responses. In *ICCV Workshops*, 2017.

[18] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012.

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

[20] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman P. Pflugfelder, Luka Cehovin Zajc, Tomás Vojír, and Gustav Häger. The visual object tracking vot2017 challenge results. In *ICCV Workshops*, 2017.

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

[22] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[23] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017.

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

[25] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming-Hsuan Yang. Hedged deep tracking. In *CVPR*, 2016.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] Chong Sun, Huchuan Lu, and Ming-Hsuan Yang. Learning spatial-aware regressions for visual tracking. In *CVPR*, 2018.

[29] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, pages 489–497, 2018.

[30] Ning Wang, Wengang Zhou, Qi Tian, Richang Hong, Meng Wang, and Houqiang Li. Multi-cue correlation filters for robust visual tracking. In *CVPR*, 2018.

[31] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

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

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

[34] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *CVPR*, 2017.