Fast-deepKCF Without Boundary Effect

Linyu Zheng, Ming Tang, Yingying Chen, Jinqiao Wang, Hanqing Lu
National Lab of Pattern Recognition, Institute of Automation, CAS, Beijing 100190, China
University of Chinese Academy of Sciences, Beijing, China
{linyu.zheng, tangm, yingying.chen, jqwang, luhq}@nlpr.ia.ac.cn

Abstract

In recent years, correlation filter based trackers (CF trackers) have received much attention because of their top performance. Most CF trackers, however, suffer from low frame-per-second (fps) in pursuit of higher localization accuracy by relaxing the boundary effect or exploiting the high-dimensional deep features. In order to achieve real-time tracking speed while maintaining high localization accuracy, in this paper, we propose a novel CF tracker, fdKCF, which casts aside the popular acceleration tool, i.e., fast Fourier transform, employed by all existing CF trackers, and exploits the inherent high-overlap among real* (i.e., *noncyclic) and dense samples to efficiently construct the kernel matrix. Our fdKCF* enjoys the following three advantages. (i) It is efficiently trained in kernel space and spatial domain without the boundary effect. (ii) Its fps is almost independent of the number of feature channels. Therefore, it is almost real-time, i.e., 24 fps on OTB-2015, even though the high-dimensional deep features are employed. (iii) Its localization accuracy is state-of-the-art. Extensive experiments on four public benchmarks, OTB-2013, OTB-2015, VOT2016, and VOT2017, show that the proposed fdKCF* achieves the state-of-the-art localization performance with remarkably faster speed than C-COT and ECO.*

1. Introduction

Visual object tracking is one of the fundamental problems in computer vision with many applications. In the model free tracking problem, the goal is to estimate the states (e.g., position and size) of the target in a whole image sequence only with the initial frame [45, 46]. Model free tracking is very challenging because the tracker has to learn the robust appearance model from a very limited training samples to resist extremely challenging interference, such as occlusions, large appearance changes, illumination variation, fast motion, and background clutters. In general, the key problem of model free tracking is how to construct a tracker which can not only tolerate appearance variation of target, but also exclude background interference, while maintaining the processing speed as fast as possible.

Figure 1: Comparison of sampling methods in KCF [22] (1st row), BACF [24] (2nd row), and our fdKCF* (last row). Training samples of KCF come from the cyclic shift of a base sample (*i.e.* learning region), and they are all virtual except for the base one. BACF obtains its training samples with target size (cyan boxes) by clipping the middle parts of all training samples of KCF, and some of them are virtual. Different from them, in fdKCF*, training samples with target size (red boxes) are densely sampled from the learning region in the traditional sliding window way, and they are all real. We call our sampling method as ***real and dense sampling***.

In recent years, correlation filter based trackers (CF trackers) have received much attention because of their top performances. Since MOSSE [3], almost all CF trackers [22, 39, 18, 28, 10, 1, 12, 13, 14, 24, 8, 31] have been relying on fast Fourier transform (FFT) to accelerate their computations. Unfortunately, while modern CF trackers' localization accuracies continue to improve, their fps' become lower and lower. We believe that the following two reasons cause this phenomenon. On the one hand, to improve the robustness of tracking algorithm itself, some representative CF trackers [12, 24] exploit techniques to relax the boundary effect which is introduced by FFT [3]. These techniques, however, inevitably destroy entire cyclicity of the training samples, resulting in much slower training speed. On the other hand, to improve the robustness of features to

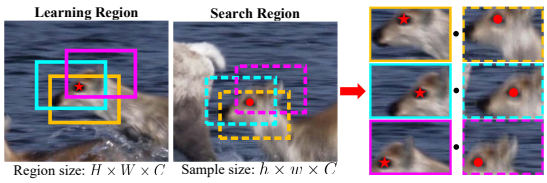


Figure 2: Illustration of redundant computations in the construction of linear kernel matrix by the brute-force approach. The color bounding boxes are three pairs of real and dense samples, $\bullet$ denotes dot product, and the red pentagram and the red dot are shared C-dimensional feature vectors by the solid and dotted line samples, respectively. The two samples of identical color are considered as a pair because the relative position of the red pentagram in the solid sample is the same as that of the red dot in the dotted sample. We have to calculate the dot product of the red pentagram and the red dot three times in this example. In practice, we need to calculate the above dot product $h \times w$ times when the dense and real samples and the brute-force approach are employed. In contrast, our fCKM calculates it only once.

appearance variations of targets, deep features which are always high-dimensional are employed by modern CF trackers [30, 14, 8, 11, 37, 5]. Although these trackers benefit from deep features remarkably, their computational costs increase significantly. Particularly, C-COT [14] which not only relaxes the boundary effect but also employs deep features can run at only 0.3 fps on GPU. Further, despite lots of techniques have been employed to accelerate its computation, ECO [8] can run at only 6 fps. Naturally, it is asked whether or not we can design a CF tracker which is able to efficiently relax or even avoid the boundary effect inherently, *i.e*., does not employ FFT to accelerate its training, and efficiently exploit deep features at the same time?

To solve the above problem, in this paper, we propose a novel CF tracker, fdKCF* which not only has not the boundary effect inherently, but also can run in real-time even though deep features are employed. First, we introduce the real and dense sampling method to avoid the boundary effect radically. As shown in Fig. 1, this sampling method is based on the traditional sliding window where all training samples are real, and it is different from the cyclic shift based sampling method used in existing CF trackers, such as KCF [22] and BACF [24], where training set contains virtual samples, resulting in the negative boundary effect. Second, we design a novel algorithm, fCKM, to construct the kernel matrix in spatial domain efficiently even if the high-dimensional deep features are employed after investigating the inherent high-overlap among real and dense samples. Finally, a Gauss-Seidel based iterative method is employed to efficiently optimize in dual space.

It is observed that there exist vast redundant computations in the construction of kernel matrix by using the brute-force approach because of the high-overlap among real and dense samples. Take the linear kernel as an example. Given the $H \times W \times C$ feature maps of learning region and detection region where C is the number of channels, we have to calculate the dot product of any two C-dimensional feature vectors, which come from the two feature maps respectively, K times where K is the number of pairs of samples which contain the above two feature vectors respectively. Indeed, this dot product need to be calculated only once. Fig. 2 shows an example. Inspired by this observation, we propose a novel algorithm, fCKM, which can construct the linear kernel matrix efficiently by eliminating redundant calculations, *i.e*., the dot product of any two C-dimensional feature vectors is only calculated once, instead of K times. fCKM conducts the following two steps to construct the matrix of linear kernel: (i) building the dot product value table of any two C-dimensional feature vectors which come from the input two feature maps respectively; (ii) obtaining each element in the linear kernel matrix by looking up the table and summing. This way, the redundant calculations of dot product of any two C-dimensional feature vectors can be replaced with looking up table with time complexity $O(1)$ rather than $O(C)$ in brute-force approach. Consequently, fCKM enjoys the following two advantages: (i) It is performed in spatial domain without the boundary effect. (ii) Its running speed is fast and insensitive to the number of feature channels. In our experiments, only a few milliseconds are taken to construct the matrix of linear kernel even though the number of feature channels up to 1024. Additionally, it can also be employed to construct many matrices of typical non-linear kernels with very little increase in time-consuming by modifying the first step and adding non-linear mapping after the second step.

Experiments are performed on four public benchmarks: OTB-2013, OTB-2015, VOT2016, and VOT2017. Our fdKCF* achieves the state-of-the-art localization performance, while running at 24 fps. As a fair comparison, when C-COT [14] and fdKCF* employ deep features of the same dimensional, run on same GPU, and do not use any other acceleration techniques, the localization accuracy of fdKCF* is higher than that of C-COT, while the mean fps of fdKCF* is about ***80 times*** that of C-COT. To the best of our knowledge, our fdKCF* is the first CF tracker which achieves both high localization accuracy and real-time speed.

2. Related Work

For the first time, KCF [21, 22] establishes the relationship between correlation filters and ridge regression. Compared to MOSSE, KCF is modeled in kernel or dual space and it can make use of multi-channel features without an increase in parameters. Besides, another important contribution of KCF is the fast calculation of kernel matrix in frequency domain. In order to improve the localization performance of KCF, HCF [30] introduces the higher dimensional deep features into KCF, however, the boundary effect become a bottleneck for its localization performance. Our

fdKCF* is also modeled in kernel or dual space. It is, however, mainly different from KCF and HCF in two aspects. First, samples of our fdKCF* are real (*i.e.*, noncyclic) and dense ones, rather than the cyclic shifts of a real base sample as in KCF and HCF (See Fig. 1). In other words, there is no boundary effect inherently in our fdKCF*, whereas KCF and HCF suffer from it. Therefore, the localization performance of our fdKCF* exceeds HCF with a large margin (about 10 percent point on OTB-2013 and OTB-2015). Second, our fdKCF* is accelerated in the spatial domain by exploiting the inherent high-overlap among real and dense samples, rather than in frequency domain as done in KCF and HCF. It is worth noting that compared to HCF, the tracking speed of our fdKCF* is much faster than its (24fps vs. 11fps on GPU) even though the search region of fdKCF* is larger than HCF (4 vs. 1.8 times target size) and fdKCF* does not exploit the cyclic samples structure to accelerate.

By exploiting the real and dense samples, LSART [37] solves for the dual variables of the KCF with linear kernel through propagating messages forward and backward in a network. It does not construct the kernel matrix explicitly. Different from LSART, our fdKCF* constructs the kernel matrix first, and then solve for the dual variables by an iterative method. The efficiency of our fdKCF* is remarkably higher than that of LSART for the following two reasons. First, in order to solve for the dual variables, every update for them requires to propagate messages forward and backward in the network of LSART, and this is time-consuming. Whereas in our fdKCF*, we only construct the kernel matrix once by our fCKM, then solve for the dual variables by an iterative method. Both steps are efficient. Second, only the first-order convergence method like SGD can be employed in LSART, while more efficient ones such as Gauss-Seidel can be used in our fdKCF*. It is worth noting that LSART can only employ the linear kernel, whereas nonlinear kernels can also be employed in our fdKCF*.

In addition, Siamese networks based trackers [2, 27, 20] achieved state-of-the-art performance in recent years. They treat tracking as a similarity learning problem and train their models by vast offline data. For completeness, we also compare our fdKCF* with typical ones in the experiment.

3. KCF without Boundary Effect

We will start with the kernel ridge regression problem, and suggest readers referring to [22, 44] for the relation between the ridge regression and the kernel ridge regression.

Let $\mathbf{X} \in R^{H\times W\times C}$ and $\mathbf{Z} \in R^{H\times W\times C}$ be the feature maps of learning region and search region, respectively, where H and W are the height and width of the feature maps, respectively, C is the number of channels. All training samples $\{\mathbf{x}_i\}_{i=1}^{N}$ where $\mathbf{x}_i \in R^{h\times w\times C}$ are sampled from $\mathbf{X}$, as shown in Fig.3, and all test samples $\{\mathbf{z}_i\}_{i=1}^{N}$ where $\mathbf{z}_i \in R^{h\times w\times C}$ are sampled from $\mathbf{Z}$ in the same way, where $h \leq H$ and $w \leq W$ are the height and width of the feature map of target, respectively, and $N = (H-h+1)\times(W-w+1)$. Further, we define the kernel matrix $\mathbf{K_{ZX}}$ as follows:

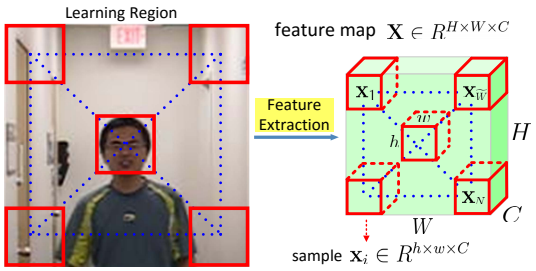


Figure 3: Sampling in feature map $\mathbf{X}$ (green cuboid). Samples $\left\{\mathbf{x}_i \mid i = 1...\widetilde{W}...N\right\}$ are obtained by using real and dense sampling method (see Fig. 1) where $\widetilde{W} = W - w + 1$ and $N = (H-h+1)\times(W-w+1)$. See Sec.3 for details.

$$\mathbf{K_{ZX}} = \begin{bmatrix} \kappa(\mathbf{z}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{z}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{z}_N, \mathbf{x}_1) & \cdots & \kappa(\mathbf{z}_N, \mathbf{x}_N) \end{bmatrix}, \tag{1}$$

where $\kappa(\cdot,\cdot)$ is a kernel. $\mathbf{K_{XX}}$ is the Gram matrix with respect to $\{\mathbf{x}_i\}_{i=1}^{N}$ if $\mathbf{Z} = \mathbf{X}$. In the rest of this paper, we will use $\mathbf{K}^L_{\mathbf{ZX}}$ and $\mathbf{K}^G_{\mathbf{ZX}}$ to indicate the linear and Gaussian kernel matrices with $\kappa^L(\mathbf{z}_i, \mathbf{x}_j) = \langle \mathbf{z}_i, \mathbf{x}_j \rangle$ and $\kappa^G(\mathbf{z}_i, \mathbf{x}_j) = g(\mathbf{z}_i, \mathbf{x}_j)$ as their elements, respectively, where $g(\cdot,\cdot)$ is Gaussian function.

According to Eq.(1), the optimization problem of KCF without the boundary effect (KCF*) can be formulated in dual space as

$$\min_{\boldsymbol{\alpha}} \|\mathbf{y} - \mathbf{K_{XX}}\boldsymbol{\alpha}\|_2^2 + \lambda \boldsymbol{\alpha}^T \mathbf{K_{XX}} \boldsymbol{\alpha}, \tag{2}$$

where $\mathbf{y} = [y_1, y_2, ..., y_N]$ is the vector of gaussian labels, λ is the regularization parameter, and $\boldsymbol{\alpha} \in R^{N\times 1}$ is the vector of dual variables. The optimization solution of Problem (2) can be expressed as

$$\boldsymbol{\alpha}^* = (\mathbf{K_{XX}} + \lambda \mathbf{I})^{-1} \mathbf{y}. \tag{3}$$

Further, given $\mathbf{X}$ and $\boldsymbol{\alpha}^*$, the process of detection in $\mathbf{Z}$ can be expressed as

$$f(\mathbf{Z}) = \mathbf{K_{ZX}} \boldsymbol{\alpha}^*. \tag{4}$$

It is clear that in order to calculate $\boldsymbol{\alpha}^*$ in Eq.(3), we have to construct $\mathbf{K_{XX}}$ with $\{\mathbf{x}_i\}_{i=1}^{N}$ first. Constructing $\mathbf{K_{XX}}$, however, is extremely time-consuming when dense samples and deep features are employed where N and C are generally large. For example, when $\mathbf{K}^L_{\mathbf{XX}}$ is constructed, its each element $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ has to be calculated by the formula

$$\begin{aligned} \kappa(\mathbf{x}_i, \mathbf{x}_j) &= \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ &= \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} \sum_{d=0}^{C-1} \mathbf{X}_{\lfloor i/\widetilde{W} \rfloor + m, (i \bmod \widetilde{W}) + n, d} \\ &\quad \cdot \mathbf{X}_{\lfloor j/\widetilde{W} \rfloor + m, (j \bmod \widetilde{W}) + n, d}, \end{aligned} \tag{5}$$

where $\widetilde{W} = W - w + 1$ and $\mathbf{X}_{p,q,d}$ is the element of $\mathbf{X}$ at the p-th row, q-th column and d-th channel. Therefore, the time complexity of constructing $\mathbf{K}^L_{\mathbf{XX}}$ with Eq.(5) is $O\left(N^2Chw\right)$, and so is $\mathbf{K}^L_{\mathbf{ZX}}$. Suppose $H = \beta h$, $H = W$, $h = w$, and replace N with $(H-h+1)\times(W-w+1)$. Then, the above complexity can be simplified to $O\left(C\beta^4h^6\right)$ [1]. It is noted that this complexity is extremely high because h and C often belong to 10 and 10^3 orders of magnitude, respectively, when deep features are employed.

4. Fast Calculation of Kernel Matrix (fCKM)

In this section, we first introduce our novel algorithm fCKM for efficient construction of $\mathbf{K_{ZX}}$, then show how it works with linear kernel and Gaussian kernel as two special cases. Finally, the analysis of complexities of fCKM and its comparison with the brute-force approach is presented.

4.1. fCKM for General Kernels

Our fCKM can construct the kernel matrix efficiently where kernel $\kappa(\cdot,\cdot)$ can be expressed as

$$\kappa(\mathbf{z},\mathbf{x}) = \psi\left(\sum_{m=0}^{h-1}\sum_{n=0}^{w-1}\sum_{d=0}^{C-1}\phi\left(\mathbf{z}_{m,n,d},\mathbf{x}_{m,n,d}\right)\right), \quad (6)$$

where $\psi(\cdot)$ and $\phi(\cdot,\cdot)$ are two functions with time complexities $O(\gamma)$ and $O(\eta)$, respectively, $\mathbf{z} \in R^{h\times w\times C}$ and $\mathbf{z}_{m,n,d}$ is the element of $\mathbf{z}$ at the m-th row, n-th column and d-th channel. We call such $\kappa(\cdot,\cdot)$ as (ψ,ϕ) kernel.

It can be observed from Fig. 2 and Fig. 3 that most elements of any sample are also elements of its spatially adjacent ones for real and dense samples. Such large shared elements will lead to large redundant computations in constructing $\mathbf{K_{ZX}}$ of (ψ,ϕ) kernel with brute-force approach, *i.e.*, calculating $\kappa(\mathbf{z}_i,\mathbf{x}_j)$'s with Eq.(6). This is because there are K pairs of samples, $(\mathbf{z},\mathbf{x})$'s, which contain $(\mathbf{Z}_{m,n,*},\mathbf{X}_{i,j,*})$ and the relative position of $\mathbf{Z}_{m,n,*}$ in $\mathbf{z}$ is the same as that of $\mathbf{X}_{i,j,*}$ in $\mathbf{x}$, where $K \in [1, h\times w]$ and $\mathbf{Z}_{m,n,*}$ is the C-dimensional feature vector at the m-th row and n-th column of $\mathbf{Z}$, leading to $\sum_{d=0}^{C-1}\phi\left(\mathbf{Z}_{m,n,d},\mathbf{X}_{i,j,d}\right)$ has to calculate K times. In order to reduce these redundant computations, we design a novel algorithm fCKM to construct $\mathbf{K_{ZX}}$ of (ψ,ϕ) kernel efficiently. Our fCKM consists of the following three steps.

(1) Building Base Table. The base table $\mathbf{T} \in R^{HW\times HW}$ of (ψ,ϕ) kernel is constructed with expression:

$$\mathbf{T}(i,j) = \sum_{d=0}^{C-1}\phi\left(\mathbf{Z}_{\lfloor j/W\rfloor,j \bmod W,d},\mathbf{X}_{\lfloor i/W\rfloor,i \bmod W,d}\right). \quad (7)$$

Consequently, $\mathbf{T}$ contains $\sum_{d=0}^{C-1}\phi\left(\mathbf{Z}_{m,n,d},\mathbf{X}_{i,j,d}\right)$ for all (m,n,i,j), and any one is calculated only once. The time complexity of this step is $O\left(\eta C\beta^4h^4 + C\beta^4h^4\right)$.

[1] Here, we use β instead of $\beta-1$ for convenience in this paper, and this is reasonable because $\beta \in [4,5]$ in general [14, 18, 12, 8, 11, 13].

(2) Constructing Summation Matrix. The summation matrix $\mathbf{S} \in R^{N\times N}$ is constructed through looking up the base table $\mathbf{T}$ and summing. Specifically,

$$\mathbf{S}(i,j) = \sum_{m=0}^{h-1}\sum_{n=0}^{w-1}\mathbf{T}(p,q), \quad (8)$$

where

$$\begin{aligned} p &= (\lfloor i/\widetilde{W}\rfloor + m)\times W + (i \bmod \widetilde{W}) + n,\\ q &= (\lfloor j/\widetilde{W}\rfloor + m)\times W + (j \bmod \widetilde{W}) + n,\\ \widetilde{W} &= W - w + 1. \end{aligned} \quad (9)$$

Consequently, $\mathbf{S}(i,j) = \psi^{-1}\left(\kappa\left(\mathbf{z}_i,\mathbf{x}_j\right)\right)$ for all (i,j). The time complexity of this step is $O\left(\beta^4h^6\right)$.

(3) Mapping. $\mathbf{K_{ZX}} \in R^{N\times N}$ of (ψ,ϕ) kernel can be obtained by mapping $\mathbf{S}$ with function $\psi(\cdot)$. Specifically,

$$\mathbf{K_{ZX}}(i,j) = \psi\left(\mathbf{S}(i,j)\right). \quad (10)$$

Consequently, $\mathbf{K_{ZX}}(i,j) = \kappa(\mathbf{z}_i,\mathbf{x}_j)$ for all (i,j). The time complexity of this step is $O\left(\gamma\beta^4h^4\right)$.

According to above steps, $\sum_{d=0}^{C-1}\phi\left(\mathbf{Z}_{m,n,d},\mathbf{X}_{i,j,d}\right)$ for all (m,n,i,j) is calculated only once, rather than K times, in constructing $\mathbf{K_{ZX}}$ of (ψ,ϕ) kernel, resulting in the high efficiency of our fCKM.

Last, we would like to discuss the key difference between KII [23] and our fCKM. In short, KII needs to validate whether the contribution function satisfies the necessary and sufficient condition, and it can only accelerate the filtering of a single filter, rather than a group of highly overlapping filters. Whereas, our fCKM focus on accelerating the filtering of such a group of filters.

4.2. fCKM for Linear Kernel Matrix

(ψ,ϕ) kernel is linear if $\psi(x) = x$ and $\phi(x,y) = xy$. Therefore, the linear kernel matrix $\mathbf{K}^L_{\mathbf{ZX}}$ can be constructed efficiently with fCKM. Specifically, according to Sec.4.1, $\mathbf{K}^L_{\mathbf{ZX}}$ can be constructed through the following two steps.

(1) Building the base table $\mathbf{T}^L$ as follows.

$$\mathbf{T}^L(i,j) = \sum_{d=0}^{C-1}\mathbf{Z}_{\lfloor j/W\rfloor,j \bmod W,d}\mathbf{X}_{\lfloor i/W\rfloor,i \bmod W,d}. \quad (11)$$

(2) Constructing the summation matrix $\mathbf{S}^L$ with Eq.(8), where $\mathbf{T}$ is replaced with $\mathbf{T}^L$ in Eq. 11. Finally, $\mathbf{K}^L_{\mathbf{ZX}} = \mathbf{S}^L$ because $\psi(x) = x$.

4.3. fCKM for Gaussian Kernel Matrix

Not only the linear kernel, but also many commonly used non-linear kernels, such as Gaussian, multi-quadric, and sigmoid ones, are (ψ,ϕ) kernel. As the most commonly used one in CF trackers, Gaussian kernel is used to show how fCKM works to construct the Gaussian kernel matrix.

(ψ, ϕ) kernel is Gaussian if $\psi(x) = \exp\left(-\frac{\sqrt{x}}{\sigma^2}\right)$ and $\phi(x, y) = (x - y)^2$. Therefore, the Gaussian kernel matrix $\mathbf{K}^G_{\mathbf{ZX}}$ can be constructed efficiently with fCKM. Specifically, according to Sec.4.1, $\mathbf{K}^G_{\mathbf{ZX}}$ can be constructed through the following three steps.

(1) Building the base table $\mathbf{T}^G$ as follows.

$$\mathbf{T}^G(i, j) = \sum_{d=0}^{C-1} \left(\mathbf{Z}_{\lfloor j/W \rfloor, j \bmod W, d} - \mathbf{X}_{\lfloor i/W \rfloor, i \bmod W, d}\right)^2. \tag{12}$$

(2) Constructing the summation matrix $\mathbf{S}^G$ with Eq.(8), where $\mathbf{T}$ is replaced with $\mathbf{T}^G$ in Eq. 12.

(3) Mapping $\mathbf{S}^G$ to $\mathbf{K}^G_{\mathbf{ZX}}$ with $\psi(x) = \exp\left(-\frac{\sqrt{x}}{\sigma^2}\right)$, *i.e.*,

$$\mathbf{K}^G_{\mathbf{ZX}}(i, j) = \exp\left(-\frac{\sqrt{\mathbf{S}^G(i, j)}}{\sigma^2}\right). \tag{13}$$

4.4. Complexity Analysis

In this section, we analyze and compare the time and space complexities of our fCKM against those of the brute-force approach in detail when the linear kernel is employed.

According to Sec. 4.1, the time complexities of the brute-force approach with Eq.(6) and our fKCM in constructing the kernel matrix $\mathbf{K}_{\mathbf{ZX}}$ of (ψ, ϕ) kernel are $O(\eta C\beta^4h^6 + C\beta^4h^6 + \gamma\beta^4h^4)$ and $O(\eta C\beta^4h^4 + C\beta^4h^4 + \beta^4h^6 + \gamma\beta^4h^4)$, respectively. Therefore, $\mathbf{K}^L_{\mathbf{ZX}}$ can be constructed in the time complexities $O\left(C\beta^4h^6\right)$ and $O\left(C\beta^4h^4 + \beta^4h^6\right)$ by the brute-force approach with Eq.(5) and our fKCM, respectively, because the time complexity of $\phi(\cdot, \cdot)$ and $\psi(\cdot)$ are $O(1)$ and $O(0)$, respectively, *i.e.*, $\eta = 1$ and $\gamma = 0$. Further, their proportional relation is

$$\frac{C\beta^4h^6}{C\beta^4h^4 + \beta^4h^6} = \frac{Ch^2}{C + h^2} = \frac{h^2}{1 + h^2/C} \tag{14}$$

In practice, when the high-dimensional deep features are employed, $C > h^2 \gg h \gg 1$. Therefore, the time complexities of constructing $\mathbf{K}^L_{\mathbf{ZX}}$ by brute-force approach is about h^2 times that of our fKCM.

Fig. 4 shows the effect of our fCKM on the reduction of computational costs compared to the brute-force approach with common $H = W = 60$ and $h = w = 15$ which are the case if the cell size (stride) of features is 4×4. It can be concluded from the figure that, when the number of channels increases, the increment of FLOPs with fCKM is much slower than that with the brute-force approach, and the more the number of channels is, the more the acceleration of fCKM is. Therefore, our fCKM can construct $\mathbf{K}^L_{\mathbf{ZX}}$ efficiently even though the high-dimensional deep features are exploited. For example, on a TITAN X GPU, when typically $H = W = 60$ and $h = w = 15$, the execution time of the Step **(1)** only increases 8μs with C increasing 1, and it always takes 4ms to perform Step **(2)** whatever C is. Therefore, when $H = W = 60$, $h = w = 15$, and $C = 600$, the time-consuming of our fCKM is about 8.8ms, whereas that of brute-force approach is about 600ms.

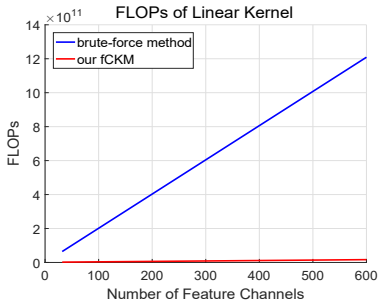


Figure 4: Comparison between the amount of computations of our fCKM and the brute-force method with Eq.(5) in calculating the linear kernel matrix $\mathbf{K}^L_{\mathbf{ZX}}$ with $H = W = 60$ and $h = w = 15$.

On the other hand, fCKM has $O\left(H^2W^2\right)$ higher space complexity than the brute-force approach. This requirement of extra space is used to store the base table $\mathbf{T}$. However, it is negligible on the current GPU and RAM. For example, fCKM only requires 50MB more than the brute-force approach does under $H = W = 60$ and float data type.

5. Fast-deepKCF without Boundary Effect

5.1. Fast Training

In Sec. 3, we have shown how to efficiently construct $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t}$ by our fCKM, where $\mathbf{X}_t$ is the $\mathbf{X}$ in frame t. Achieving optimal $\boldsymbol{\alpha}^*_t$ by directly using Eq.(3), however, is time-consuming, because the time complexity of matrix inversion is $O\left(N^3\right)$. Even though $\boldsymbol{\alpha}^*_t$ is achieved through solving a system of linear equations with Gauss elimination method, the complexity is still $O(\frac{1}{3}N^3)$. In order to achieve $\boldsymbol{\alpha}^*_t$ more efficiently, we adopt the iterative approach [12] based on the Gauss-Seidel. Specifically, we decompose $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \lambda\mathbf{I}$ into a lower triangular $\mathbf{L}_t$ and a strictly upper triangular $\mathbf{U}_t$, *i.e.*, $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \lambda\mathbf{I} = \mathbf{L}_t + \mathbf{U}_t$. Then, $\boldsymbol{\alpha}^*_t$ can be efficiently solved by the following iterative expressions:

$$\boldsymbol{\alpha}^{*(j)}_t \leftarrow \boldsymbol{\alpha}^*_{t-1}, \qquad j = 0, \tag{15a}$$

$$\boldsymbol{\alpha}^{*(j)}_t \leftarrow \mathbf{L}_t \setminus \left(\mathbf{y} - \mathbf{U}_t\boldsymbol{\alpha}^{*(j-1)}_t\right), \qquad j > 0, \tag{15b}$$

where j indicates the number of iterations, and $\boldsymbol{\alpha}^*_{t-1}$ is the optimal solution at frame $t - 1$. In practice, 5 iterations are enough for the satisfactory solution $\boldsymbol{\alpha}^*_t$. Note that this iterative method is efficient because Eq. (15b) can be solved efficiently with forward substitution, and the time complexity of each iterations is $O(N^2)$.

As a comparison to KCF, it is easy to know that the time complexities of constructing kernel matrix and solving linear system are $O\left(C\beta^2h^2\log(\beta h)\right)$ and $O(N)$, respectively, in KCF, being significantly lower than those of our

fdKCF*. This is because KCF exploits circulant samples which cause the negative boundary effect.

5.2. Update

To robustly locating the target object, updating the appearance model of a tracker is often necessary. Similar to other CF trackers [18, 38, 22], we update $\mathbf{X}_t$ in Sec. 5.1 by means of the following linear weighting approach, *i.e.*,

$$\begin{aligned} \mathbf{X}_1 &= \widetilde{\mathbf{X}}_1, \\ \mathbf{X}_t &= (1-\delta)\,\mathbf{X}_{t-1} + \delta\widetilde{\mathbf{X}}_t, \qquad t > 1, \end{aligned} \tag{16}$$

where $\widetilde{\mathbf{X}}_t$ is the actual sampled feature map of learning region in frame t, and δ is the learning rate.

5.3. Fast Multi-scale Detection

Scale-pyramid based method [28] is employed to locate the target object and estimate its proper scale simultaneously. Specifically, given $\mathbf{X}$, $\boldsymbol{\alpha}^*$, and the scale-pyramid $\left\{\mathbf{Z}^i\right\}_{i=1}^{S}$ of feature maps of detection regions, where S is the level of the scale-pyramid, the fast detection of target object on each scale can be expressed as

$$f\left(\mathbf{Z}^i\right) = \mathbf{K}_{\mathbf{Z}^i\mathbf{X}}\boldsymbol{\alpha}^*, \quad \forall i. \tag{17}$$

Note that any (ψ, ϕ) kernel can be used in Eq.(17).

Particularly, if (ψ, ϕ) kernel is linear, the optimal scale as well as the target location can be achieved with a more efficient approach. That is,

$$\mathbf{w} = \sum_{i=1}^{N} \boldsymbol{\alpha}^*(i)\mathbf{x}_i, \tag{18a}$$

$$\hat{f}\left(\mathbf{Z}^i\right) = \hat{\mathbf{Z}}^i \odot \hat{\mathbf{w}}^*, \quad \forall i, \tag{18b}$$

where $\boldsymbol{\alpha}^*(i)$ is the i-th element of $\boldsymbol{\alpha}^*$ and $\hat{\bullet}$ denotes the DFT of $\bullet$ in $H \times W$ dimension. Before conducting DFT for $\bullet$, 0's are padded to $\bullet$'s bottom right to make its dimensionality be $H \times W$ if the dimensionality of $\bullet$ is less than $H \times W$. Eq.(18) is more efficient than Eq.(17) in the case of multi-scale detection with linear (ψ, ϕ) kernel because Eq.(18a) is executed only once.

6. Experiments

We evaluate our fdKCF* on four public benchmarks, OTB-2013 [45], OTB-2015 [46], VOT2016 [26] and VOT2017 [25], and compare its performance with the state-of-the-art and representative trackers. All parameters of fdKCF* are kept consistent in all experimental comparisons.

6.1. Implementation Details

Platform. Our fCKM is implemented in C++, and the rest of our fdKCF* is implemented in PyTorch [33]. Experiments are performed on Linux with single TITAN X GPU.

Features. Similar to C-COT [14], our fdKCF* only employs deep features to show performances of the algorithm itself. Specifically, we adopt the VGG-M-BN [4] trained on ImageNet [16] for feature extraction. We first change original strides of Conv-2 and Conv-5 from 2 to 1 to improve the localization accuracy. Then, the output maps of Conv-1 followed by an average pooling layer with kernel size 2×2 and stride 2×2 are employed as shallow level features (96 channels), and the output maps of Conv-5 followed by a $2\times$ bilinear interpolation layer are employed as deep level features (512 channels). As a result, both shallow level features and deep level features are of 4×4 cell size (stride).

Parameters. We set different learning rates (Sec. 5.2) for shallow level features and deep level features. Specifically, $\delta = \delta_s = 0.01$ for shallow ones and $\delta = \delta_d = 0.005$ for deep ones. The regularization parameter λ in Eq. 2 is set to 0.01. The maximum number of iterations j in Eq. 15 is set to 5. Similarly to SRDCF [12], we set the image area of the square sampling region to 4^2 times the target area, and it is re-scaled to the area of 200^2 if its area is less than 200^2, and to the area of 240^2 if its area is greater than 240^2.

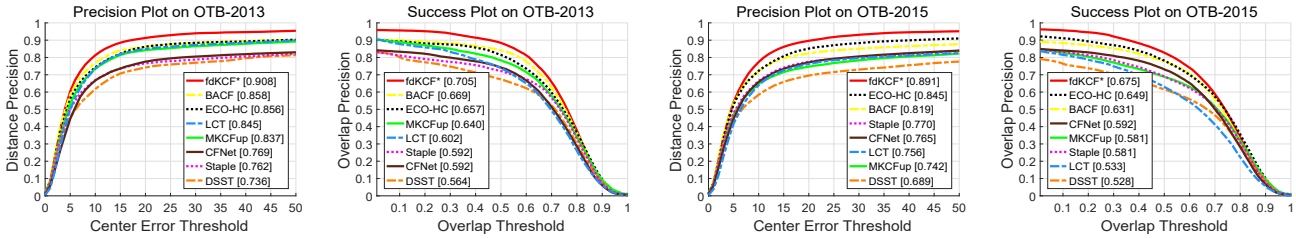
Scaling. In order to balance the localization accuracy and tracking speed, we set 5 levels scale-pyramid (Sec. 5.3).

Kernel. We only employ the linear kernel in fdKCF* in our current experiments. The reasons are (1) most of state-of-the-art CF trackers, such as BACF [18] and ECO [8], can only employ the linear kernel, (2) fdKCF* with linear kernel runs slightly faster than that with Gaussian kernel.

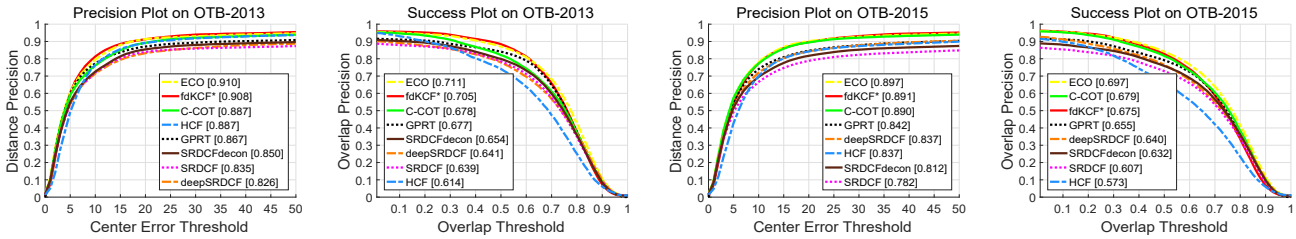
6.2. Evaluation on OTB datasets

In our OTB-2013 and OTB-2015 experiments, we compare our fdKCF* with state-of-the-art CF trackers and non-CF trackers, respectively. When comparing with CF trackers, following the standard benchmark protocols in the OTB-2015 [46], all trackers are quantitatively evaluated by five metrics, namely precision plot, success plot, distance precision (DP), overlap precision (OP), and AUC. In addition, as pointed out in [32], the definition of DP in OTB-2015 is defective because it is sensitive to the size of bounding boxes, and they propose the normalized precision, P_{norm}, to measure the localization accuracy. Based on their work, we evaluate all trackers with $P_{norm}@0.2$ which is computed as the percentage of frames in a video where P_{norm} is smaller than 0.2. When comparing with non-CF trackers, all trackers are quantitatively evaluated by AUC metric because they all reported AUCs in their original papers and there is no way to obtain the detailed tracking results of some of them to evaluate them with other metrics.

Comparion with CF trackers. We divide state-of-the-art CF trackers into two groups for a thorough comparison. The first group consists of seven trackers which can run at real-time speed, *i.e.* beyond 20 fps. These trackers are MKCFup [39], BACF [18], ECO-HC [8], LCT [31], Sta-



(a) Comparison with real-time CF trackers.



(b) Comparison with high localization performance CF trackers.

Figure 5: The mean precision and success plots of our fdKCF* and (a) seven modern real-time CF trackers, (b) seven modern CF trackers that produce state-of-the-art localization accuracy, on OTB-2013 and OTB-2015, respectively. The mean distance precisions and AUCs are reported in the legends. fdKCF* outperforms all other real-time CF trackers by large margins.

	fdKCF*	MKCFup	BACF	ECO-HC	LCT	Staple	DSST	CFNet
mOP-13	0.884	0.784	0.841	0.815	0.739	0.721	0.673	0.742
mOP-15	0.828	0.689	0.776	0.782	0.630	0.691	0.615	0.731
mPN-13	0.846	0.760	0.816	0.772	0.773	0.711	0.649	0.725
mPN-15	0.820	0.684	0.771	0.762	0.691	0.720	0.628	0.728

(a) Comparison with real-time CF trackers.

	fdKCF*	C-COT	ECO	GPRT	HCF	SRDCF	decon	deep
mOP-13	0.884	0.821	0.871	0.841	0.741	0.785	0.799	0.779
mOP-15	0.828	0.816	0.842	0.791	0.661	0.728	0.759	0.765
mPN-13	0.846	0.782	0.832	0.818	0.783	0.748	0.772	0.740
mPN-15	0.820	0.805	0.819	0.793	0.735	0.713	0.753	0.755
mFPS-15	24	0.3	6	5	11	8	1	<1

(b) Comparison with high localization performance CF trackers.

Table 1: The mean OP (mOP) and mean P_{norm}@0.2 (mPN) of our fdKCF* and (a) seven modern real-time CF trackers, (b) seven modern CF trackers that produce state-of-the-art localization accuracy, on OTB-2013 and OTB-2015. decon and deep are denote SRDCFdecon and deepSRDCF, respectively. The best three results on each line are shown in red, blue and magenta, respectively. fdKCF* outperforms all other real-time CF trackers by large margins. In addition, we also report the mean fps' (mFPS) of trackers in (b) on OTB-2015, and fdKCF* is visibly faster than them.

ple [1], DSST [9], and CFNet [41]. The second group is formed by other seven trackers which produce state-of-the-art localization accuracy but can not run at real-time. These trackers are ECO [8], C-COT [14], deepSRDCF [11], SRDCFdecon [13], SRDCF [12], GPRT [48], and HCF [30].

Fig. 5a and Table 1a show the comparison of our fdKCF* with the first group of CF trackers. As shown in Fig. 5a, our fdKCF* obtains the mean DP and AUC of 90.8% and 70.5% on OTB-2013, as well as 89.1% and 67.5% on OTB-2015, respectively, leading the second best trackers with a significant gain of 5.0%, 3.6%, 4.6% and 2.6%, respectively. In addition, as shown in Table 1a, our fdKCF* obtains the mean P_{norm}@0.2 and mean OP of 84.6% and 88.4% on OTB-2013, as well as 82.0% and 82.8% on OTB-2015, respectively, leading the second best trackers with a significant gain of 3.0%, 4.3%, 4.9% and 4.6%, respectively. These results show that our fdKCF* significantly outperforms all other state-of-the-art real-time CF trackers in localization performance. We believe that this is because our efficient fdKCF* not only avoids the boundary effect inherently, but also utilizes the powerful deep features.

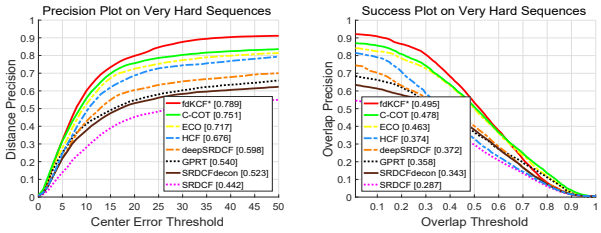


Figure 6: The mean precision and success plots of our fdKCF* and seven modern CF trackers that produce state-of-the-art localization accuracy on the very hard sequences [19] of OTB-2015. The mean distance precisions and AUCs are reported in the legends.

Fig. 5b and Table 1b show the comparison of our fdKCF* with the second group of CF trackers. Judging from the mean DPs and AUCs as shown in Fig. 5b, the localization performance of our fdKCF* is comparable to that of C-COT and slightly worse than that of ECO. However, judging from the mean P_{norm}@0.2 and mean OP as shown in Table 1b, the localization performance of our fdKCF* is competitive with that of ECO and obviously better than that of C-COT. Besides, as shown in the last line of Table 1b, the mean fps of our fdKCF* is visibly higher than those of

	where	AUC-2013	AUC-2015	Real-Time
SINT+	CVPR2016	0.655	0.571	no
SINT++	CVPR2018	0.624	0.574	no
RASNet	CVPR2018	0.670	0.642	yes
SASiam	CVPR2018	0.677	0.657	yes
SiamRPN	CVPR2018	0.658	0.636	yes
DaSiamRPN	ECCV2018	0.655	0.658	yes
StruckSiam	ECCV2018	0.638	0.621	yes
CREST	ICCV2017	0.673	0.623	no
DSLT	ECCV2018	0.683	0.660	no
DAT	NIPS2018	0.704	0.668	no
PTAV	ICCV2017	0.663	0.635	yes
TRACA	CVPR2018	0.652	0.603	yes
FlowTrack	CVPR2018	0.689	0.655	no
LSART	CVPR2018	0.701	0.672	no
VITAL	CVPR2018	0.710	0.682	no
fdKCF*	ours	0.705	0.675	yes

Table 2: The AUCs of our fdKCF* and other state-of-the-art non-CF trackers on OTB-2013 and OTB-2015. The best two results are shown in red and blue, respectively.

all trackers in the second group (including ECO), although there is not sparse update and feature dimension reduction, which are employed by ECO to accelerate, in fdKCF*.

We believe that the following three reasons cause the localization performance of our fdKCF* to be slightly worse than that of ECO. (1) The clustering algorithm GMM employed by ECO improves its robust, whereas there is no similar component in our fdKCF*. (2) The self-adaptive dimensionality reduction and weighting different features are used in ECO, whereas our fdKCF* does not use the similar components. (3) Both deep features and hand-crafted features, i.e., HOG [7] and Color-Names [15], are employed in ECO, whereas our fdKCF* only employs deep features. In fact, hand-crafted features can improve the localization performance of trackers on the sequences with easy challenges. To illustrate this fact, Han, *et al*. [19] divide the sequences of OTB-2015 into three sets: easy, hard, and very hard, according to the localization performance of most state-of-the-art trackers. It is not hard to find that trackers employed hand-crafted features always achieve high localization performances on the easy sequences. To further illustrate the above point, we show the comparison of our fdKCF* with the second group of CF trackers on the very hard sequences in Fig. 6. It is seen that the localization performance of our fdKCF* is obviously better than that of all other CF trackers (including ECO) on the very hard set. Note that C-COT, which mainly use deep features, also outperforms other trackers except fdKCF*. It is concluded that employing hand-crafted features may not improve but weaken the localization performance of trackers on very hard sequences.

Comparion with non-CF trackers. We compare fdKCF* with state-of-the-art non-CF trackers, including SINT+ [40], SINT++ [43], RASNet [42], SASiam [20], SiamRPN [27], DaSiamRPN [49], StruckSiam [47], CREST [35], DSLT [29], DAT [34], PTAV [17], TRACA [6], FlowTrack [50], LSART [37], and VITAL [36], on OTB-2013 and OTB-2015. Table 2 shows the results. It is seen that the localization accuracy of our fdKCF* outperforms most non-CF trackers, and outperforms all other real-time ones.

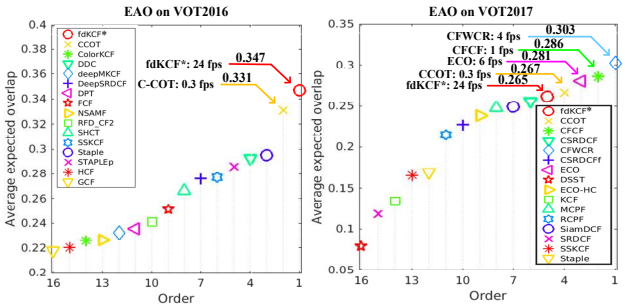


Figure 7: Expected average overlap on VOT2016 and VOT2017. Best trackers are closer to the top-right corner.

6.3. Evaluation on VOT datasets

We present the evaluation results on VOT2016 [26] and VOT2017 [25] datasets which contain 60 sequences, respectively. We follow the VOT challenge protocol to compare trackers, where mainly reports the expected average overlap (EAO) and rank trackers based on it.

Fig. 7 shows the EAO ranking plots where we compare our fdKCF* against the top-15 CF trackers on VOT2016 and VOT2017, respectively. The performances of these trackers come from the VOT report. On the whole, the EAO score of our fdKCF* is competitive to that of C-COT, which is the winner of VOT2016 challenge, and slightly worse than that of ECO. However, the tracking speed of our fdKCF* is visibly faster than those of C-COT and ECO. These conclusions are consistent with those obtained on the OTB datasets. Note that CFWCR and CFCF are improved versions based on ECO and C-COT, respectively. On the contrary, our fdKCF* is a novel tracking framework without any further improvements and tricks.

7. Conclusions and Future work

A novel CF tracker, fdKCF*, with the state-of-the-art localization accuracy and real-time speed is proposed in this paper. fdKCF* achieves the state-of-the-art accuracy because there is no boundary effect with it and powerful deep features are also employed. fdKCF* is able to run at real-time speed because a novel acceleration method, fCKM, is developed in spatial domain. Through exploiting the inherent high-overlap among real and dense samples, fCKM is able to construct the kernel matrix efficiently even though the high-dimensional deep features are employed. Future work can be found in the supplementary material.

Acknowledgement. This work was supported by National Natural Science Foundation of China under Grants 61772527, 61806200, and 61702510.

References

[1] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409, 2016.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[3] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010.

[4] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[5] Kai Chen and Wenbing Tao. Convolutional regression for visual tracking. *IEEE Transactions on Image Processing*, 27(7):3611–3620, 2018.

[6] Jongwon Choi, Hyung Jin Chang, Tobias Fischer, Sangdoo Yun, Kyuewang Lee, Jiyeoup Jeong, Yiannis Demiris, and Jin Young Choi. Context-aware deep feature compression for high-speed visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 479–488, 2018.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[8] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, Michael Felsberg, et al. Eco: Efficient convolution operators for tracking. In *CVPR*, volume 1, page 3, 2017.

[9] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[10] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1561–1575, 2017.

[11] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 58–66, 2015.

[12] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[13] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1430–1438, 2016.

[14] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488. Springer, 2016.

[15] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5486–5494, 2017.

[18] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *ICCV*, pages 1144–1152, 2017.

[19] Bohyung Han, Jack Sim, and Hartwig Adam. Branchout: Regularization for online ensemble tracking with convolutional neural networks. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2217–2224, 2017.

[20] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[21] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[22] 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, 2014.

[23] Mohamed Hussein, Fatih Porikli, and Larry Davis. Kernel integral images: A framework for fast non-uniform filtering. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[24] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation filters with limited boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4630–4638, 2015.

[25] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Gustav Hager, Alan Lukezic, Abdelrahman Eldesokey, et al. The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1972, 2017.

[26] Matej Kristan, Aleš Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin Zajc, Tomas Vojir, Gustav Häger, Alan Lukežič, and Gustavo Fernandez. The visual object tracking vot2016 challenge results. Springer, Oct 2016.

[27] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[28] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV Workshops (2)*, pages 254–265, 2014.

[29] Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep regression tracking with shrinkage loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 353–369, 2018.

[30] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 3074–3082, 2015.

[31] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5388–5396, 2015.

[32] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

[33] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, 2017.

[34] Shi Pu, Yibing Song, Chao Ma, Honggang Zhang, and Ming-Hsuan Yang. Deep attentive tracking via reciprocative learning. In *Advances in Neural Information Processing Systems*, pages 1935–1945, 2018.

[35] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2555–2564, 2017.

[36] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8990–8999, 2018.

[37] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Learning spatial-aware regressions for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8962–8970, 2018.

[38] Ming Tang and Jiayi Feng. Multi-kernel correlation filter for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2015.

[39] Ming Tang, Bin Yu, Fan Zhang, and Jinqiao Wang. High-speed tracking with multi-kernel correlation filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4874–4883, 2018.

[40] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429, 2016.

[41] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5000–5008. IEEE, 2017.

[42] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: residual attentional siamese network for high performance online visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4854–4863, 2018.

[43] Xiao Wang, Chenglong Li, Bin Luo, and Jin Tang. Sint++: robust visual tracking via adversarial positive instance generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4864–4873, 2018.

[44] Max Welling. Kernel ridge regression.

[45] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

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

[47] Yunhua Zhang, Lijun Wang, Jinqing Qi, Dong Wang, Mengyang Feng, and Huchuan Lu. Structured siamese network for real-time visual tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 351–366, 2018.

[48] Linyu Zheng, Ming Tang, and Jinqiao Wang. Learning robust gaussian process regression for visual tracking. In *IJCAI*, pages 1219–1225, 2018.

[49] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.

[50] Zheng Zhu, Wei Wu, Wei Zou, and Junjie Yan. End-to-end flow correlation tracking with spatial-temporal attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 548–557, 2018.