Learning Background-Aware Correlation Filters for Visual Tracking

Hamed Kiani Galoogahi[1]*, Ashton Fagg[2,1], and Simon Lucey[1,2]

[1]Robotics Institute
Carnegie Mellon University, USA

[2] SAIVT Lab
Queensland University of Technology, Australia

Abstract

*Correlation Filters (CFs) have recently demonstrated excellent performance in terms of rapidly tracking objects under challenging photometric and geometric variations. The strength of the approach comes from its ability to efficiently learn - on the fly - how the object is changing over time. A fundamental drawback to CFs, however, is that the background of the target is not be modeled over time which can result in suboptimal performance. Recent tracking algorithms have suggested to resolve this drawback by either learning CFs from more discriminative deep features (*e.g. *DeepSRDCF [9] and CCOT [11]) or learning complex deep trackers (*e.g. *MDNet [28] and FCNT [33]). While such methods have been shown to work well, their use comes at a high cost: extracting deep features or applying deep tracking frameworks is very computationally expensive. This limits the real-time performance of such methods, even on high-end GPUs. This work proposes a Background-Aware CF based on hand-crafted features (HOG [6]) that can efficiently model how both the foreground and background of the object varies over time. Our approach, like conventional CFs, is extremely computationally efficient- and extensive experiments over multiple tracking benchmarks demonstrate the superior accuracy and real-time performance of our method compared to the state-of-the-art trackers including those based on a deep learning paradigm.*

1. Introduction

Correlation Filters (CFs) have been a widely used framework for visual object tracking [23, 37, 10, 13], due to their superior computation and fair robustness to photometric and geometric variations. CF trackers can learn and detect quickly in the frequency domain [20], being the most notable example the MOSSE tracker with the tracking speed of ∼700 frames per second [3]. Furthermore, these trackers learn "on-the-fly". The approach quickly models how an object varies visually over time by updating the tracker when the next frames become available. Such per frame adaptation offers robust tracking under challenging circumstances such as motion blur, scaling and lighting variation.

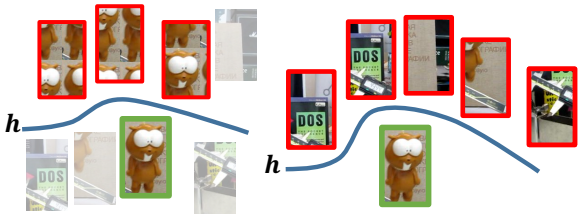


Figure 1. (a) Traditional CFs discard the background patches, and instead learn from shifted patches of the cropped target. This may result to suboptimal results. (b) The BACF, however, exploits all background patches as negative examples for learning a filter which is more discriminative to background clutter.

Learning CF trackers in the frequency domain, however, comes at the high cost of learning from circular shifted examples of the foreground target. These shifted patches are implicitly generated through the circulant property of correlation in the frequency domain and are used as negative examples for training the filter [20]. All shifted patches are plagued by circular boundary effects and are not truly representative of negative patches in real-world scenes [17].

These boundary effects have been shown to have a drastic impact on tracking performance, due to a number of factors. First, learning from limited shifted patches may lead to training an over-fitted filter which is not well-generalized to rapid visual deformation *e.g.* caused by fast motion [10]. Second, the lack of real negative training examples can drastically degrade the robustness of such trackers against cluttered background, and as a result, increase the risk of tracking drift specifically when the target and background display similar visual cues. Third, discarding background information from the learning process may reduce the tracker's ability to distinguish the target from occlusion patches. This limits the potential of such trackers to re-detect after an occlusion or out-of-plane movement [10].

Recently, two methods were proposed to address the dis-

*Author Contact: hamedkg@gmail.com

advantage of learning from shifted foreground patches [17, 10]. The method of CFs with limited boundaries (CFLB) proposed to learn CFs with less boundary effects for the tasks of facial landmark localization and object tracking. Despite its promising results, this method was limited to learning CFs from pixel intensities- which as shown in [16] are not expressive enough for detecting challenging patterns in visual contents. Similar to our work, spatially regularized CFs (SRDCF) [10] proposed to learn trackers from training examples with large spatial supports. The major disadvantage of this method is that the regularized objective is costly to optimize, even in the Fourier domain. Furthermore, in order to form the regularization weights, a set of hyper-parameters must be carefully tuned, which if not performed correctly can lead to poor tracking performance.

Contribution: We propose to learn Background-Aware Correlation Filters (BACF) for real-time object tracking. Our method is capable of learning/updating filters from real negative examples densely extracted from the background. We demonstrate that learning trackers from negative background patches, instead of shifted foreground patches, achieves superior accuracy with real-time performance. This paper offers the following contributions:

- We propose a new correlation filter for real-time visual tracking. Unlike prior CF-based trackers in which negative examples are limited to circular shifted patches, our tracker is trained from real negative training examples, densely extracted from the background.
- We propose an efficient Alternating Direction Method of Multipliers (ADMM) based approach for learning our filter on multi-channel features (*e.g.* HOG), with computational cost of $\mathcal{O}(LKT\log(T))$, where T is the size of vectorized frame, K is the number of feature channels, and L is the ADMM's iterations.

We calculate model updates with Sherman-Morrison lemma to cope with changes in target and background appearance with real-time performance. We extensively evaluate our tracker on OTB50, OTB100, Temple-Color128 and VOT2015 datasets. The result demonstrates very competitive accuracy of our method compared to the state-of-the-art CF based and deep trackers, with superior real-time tracking speed of $\sim$ 40 FPS on a CPU.

2. Prior Art

The interest in employing CFs for visual tracking was ignited by the seminal work of Bolme *et al*. [3] on the MOSSE filter with an impressive speed of $\sim$700 FPS. Thereafter, several works [7, 8, 13, 22, 1] were built upon the MOSSE approach showing notable improvement by learning CFs trackers on multi-channel features such as HOG [6]. All these approaches, however, imitated the standard formulation of CFs in the frequency domain to retain their computations efficient for real-time applications. Learning CF trackers quickly in the frequency domain, however, is highly affected by boundary effects of shifted patches [17], leading to suboptimal training [10]. Moreover, such methods solely learn trackers from object patches cropped from the whole frame, and the background visual information is discarded from the learning process. This leads to poor discrimination against cluttered background, and thereby, increases the risk of spurious detection when the target and its surrounding background share similar visual cues [10]. Several recent works addressed the constraint of learning from shifted patches by exploiting training samples whose spatial size is much larger than the trained filters [17, 10, 8, 9]. Learning from large training samples not only dramatically reduces boundary effects [17], but offers learning filters from a huge number of background patches [10]. The method of CFLB [17] was originally designed to learn from pixel intensities which was shown to be inaccurate and suffers from poor generalization on challenging patterns [16]. Our method, on the other hand, is capable of handling more discriminative and well-generalized multi-channel features such as HOG [6] and convolutional neural networks (CNNs) features [31]. The SRDCF method [10] and its variations [8, 9] require regularization weights to penalize the correlation filter coefficients during learning. These weights are highly target and video dependent, and have to be carefully fine-tuned over a set of sensitive hyper-parameters to perform well for each video. Furthermore, due to their computational expense ($\sim$4 FPS), SRDCF methods are not a suitable choice for real-time tracking.

The excellent performance of deep convolutional neural networks (CNNs) on several challenging vision tasks [19, 31, 24, 14] has encouraged more recent works to either exploit CNN deep features within CFs framework [26, 9, 11] or design deep architectures [2, 28, 34, 5, 32, 33] for robust visual tracking. This trend has its own pros and cons. Compared to hand-crafted features such as HOG, learning CF trackers using CNN features significantly improves their robustness against geometric and photometric variations [11]. This is mainly resulted from the high discrimination of such features, since CNNs are trained over large scale dataset [19]. However, extracting CNN features from each frame and training/updating CF trackers over high dimensional deep features is computationally expensive. Such an approach leads to poor real time performance ($\sim$ 0.2 FPS in the case of [9, 11]). Similarly, purely deep trackers also suffer the same drawback [2, 28, 34, 5, 32, 33], with some methods performing at only 1 FPS on a typical desktop PC.

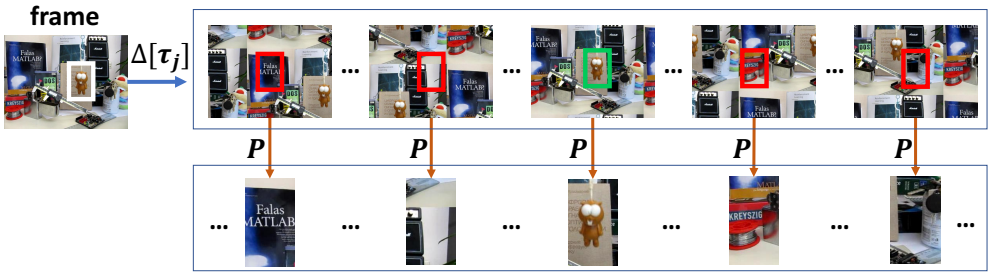


Figure 2. BACF learns from all possible positive and negative patches extracted from the entire frame. $[\Delta\tau_j]$ generates all circular shifts of the frame over all $j = [0, ..., T-1]$ steps. T is the length of vectorized frame. $\mathbf{P}$ is the cropping operator (a binary matrix) which crops the central patch of each shifted image. The size of the cropped patches is same as the size of the target/filter (D), where $T \gg D$. All cropped patches are utilized to train a CF tracker. In practice, we *do not* apply circular shift and cropping operators. Instead, we perform these operations efficiently by augmenting our objective in the Fourier domain. The red and green boxes indicate the negative (background) and positive (target) training patches. Please refer to Section 4 for more details.

3. Correlation Filters

Learning multi-channel CFs in the spatial domain is formulated by minimizing the following objective [16]:

$$E(\mathbf{h}) = \frac{1}{2}||\mathbf{y} - \sum_{k=1}^{K} \mathbf{h}_k \star \mathbf{x}_k||_2^2 + \frac{\lambda}{2}\sum_{k=1}^{K}||\mathbf{h}_k||_2^2 \quad (1)$$

where $\mathbf{x}_k \in \mathbb{R}^D$ and $\mathbf{h}_k \in \mathbb{R}^D$ refers to the kth channel of the vectorized image and filter respectively, and K is the number of feature channels. $\mathbf{y} \in \mathbb{R}^D$ is the desired correlation response, λ is a regularization, and $\star$ is the spatial correlation operator. Eq. 1 can be identically expressed as a ridge regression objective in the spatial domain:

$$E(\mathbf{h}) = \frac{1}{2}\sum_{j=1}^{D}||\mathbf{y}(j) - \sum_{k=1}^{K}\mathbf{h}_k^\top \mathbf{x}_k[\Delta\tau_j]||_2^2 + \frac{\lambda}{2}\sum_{k=1}^{K}||\mathbf{h}_k||_2^2 \quad (2)$$

where $\mathbf{y}(j)$ is the j-th element of $\mathbf{y}$. $[\Delta\tau_j]$ is the circular shift operator, and $\mathbf{x}_k[\Delta\tau_j]$ applies a j-step discrete circular shift to the signal $\mathbf{x}_k$. For a full treatment of multi-channel correlation filters, please see [16].

The main drawback of Eq. 2 is learning a correlation filter/detector from $D-1$ circular shifted foreground patches which are generated through the $[\Delta\tau_j]$ operator. This trains a filter which perfectly discriminates the foreground target from its shifted examples. As mentioned earlier, this, however, increases the risk of over-fitting and limits the potential of the filter to classify the target from real non-target patches (Fig. 1 (a)). For generic object detection task (such as pedestrian detection in [16]), this drawback can be significantly diminished by exploiting a huge amount of positive (pedestrian) and negative (non-pedestrian) patches to train a well-generalized filter/detector. This, however, is not practical for the task of visual tracking. The target is the only positive sample available at the training time and gathering positive and negative examples from a pre-collected training set for each individual target is infeasible. Fortunately, the target comes with a large surrounding background which can be used as negative samples at the training stage. We propose the method of background-aware correlation filters to directly learn more robust and well-generalized CF tracker from background patches (Fig. 1 (b)).

4. Background-Aware Correlation Filters

We propose to learn multi-channel background-aware correlation filters by minimizing the following objective:

$$E(\mathbf{h}) = \frac{1}{2}\sum_{j=1}^{T}||\mathbf{y}(j) - \sum_{k=1}^{K}\mathbf{h}_k^\top \mathbf{P}\mathbf{x}_k[\Delta\tau_j]||_2^2 + \frac{\lambda}{2}\sum_{k=1}^{K}||\mathbf{h}_k||_2^2 \quad (3)$$

where $\mathbf{P}$ is a $D \times T$ binary matrix which crops the mid D elements of signal $\mathbf{x}_k$. In this formulation, $\mathbf{x}_k \in \mathbb{R}^T$, $\mathbf{y} \in \mathbb{R}^T$ and $\mathbf{h} \in \mathbb{R}^D$, where $T \gg D$.

For tracking task, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{h}$ are respectively a training sample with large spatial support, $\mathbf{y}$ is the correlation output with a peak centered upon the target of interest, and $\mathbf{h}$ is the correlation filter whose spatial size is much smaller than training samples. Applying the circular shift operator on the training sample followed by the cropping operator, $\mathbf{P}\mathbf{x}_k[\Delta\tau_j]$, returns all possible patches with the size of D from the entire frame, Fig. 2. The cropped patch corresponding to the peak of the correlation output displays the target (positive example), and those corresponding to

the zero values of the correlation output display the background content (negative examples). The computational cost of Eq. 3 is approximately the same as Eq. 2, $\mathcal{O}(D^3K^3)$, since $\mathbf{P}$ can be precomputed, and $\mathbf{P}\mathbf{x}_k$ (cropping) can be efficiently performed via a lookup table.

Correlation filters are typically learned in the frequency domain, for computational efficiency [20]. Similarly, Eq. 3 can be expressed in the frequency domain as:

$$\begin{aligned} E(\mathbf{h}, \hat{\mathbf{g}}) &= \frac{1}{2}||\hat{\mathbf{y}} - \hat{\mathbf{X}}\hat{\mathbf{g}}||_2^2 + \frac{\lambda}{2}||\mathbf{h}||_2^2 \\ \text{s.t.} \quad \hat{\mathbf{g}} &= \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h} \end{aligned} \tag{4}$$

where, $\hat{\mathbf{g}}$ is an auxiliary variable and the matrix $\hat{\mathbf{X}}$ is defined as $\hat{\mathbf{X}} = [\text{diag}(\hat{\mathbf{x}}_1)^\top, \ldots, \text{diag}(\hat{\mathbf{x}}_K)^\top]$ of size $T \times KT$. $\mathbf{h} = [\mathbf{h}_1^\top, \ldots, \mathbf{h}_K^\top]^\top$ and $\hat{\mathbf{g}} = [\hat{\mathbf{g}}_1^\top, \ldots, \hat{\mathbf{g}}_K^\top]^\top$ respectively show the $KD \times 1$ and $KT \times 1$ over-complete representations of $\mathbf{h}$ and $\hat{\mathbf{g}}$ by concatenating their K vectorized channels. $\mathbf{I}_K$ is a $K \times K$ identity matrix, and $\otimes$ indicates the Kronecker product. $\hat{\mathbf{A}}$ denotes the Discrete Fourier Transform (DFT) of a signal, such that $\hat{\mathbf{a}} = \sqrt{T}\mathbf{F}\mathbf{a}$, where $\mathbf{F}$ is the orthonormal $T \times T$ matrix of complex basis vectors for mapping to the Fourier domain for any T dimensional vectorized signal. The transpose operator $^\top$ on a complex vector or matrix computes the conjugate transpose.

4.1. Augmented Lagrangian

To solve Eq. 4, we employ an Augmented Lagrangian Method (ALM) [4]:

$$\begin{aligned} \mathcal{L}(\hat{\mathbf{g}}, \mathbf{h}, \hat{\boldsymbol{\zeta}}) &= \frac{1}{2}||\hat{\mathbf{y}} - \hat{\mathbf{X}}\hat{\mathbf{g}}||_2^2 + \frac{\lambda}{2}||\mathbf{h}||_2^2 \\ &+ \hat{\boldsymbol{\zeta}}^\top(\hat{\mathbf{g}} - \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}) \\ &+ \frac{\mu}{2}||\hat{\mathbf{g}} - \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}||_2^2 \end{aligned} \tag{5}$$

where μ is the penalty factor and $\hat{\boldsymbol{\zeta}} = [\hat{\boldsymbol{\zeta}}_1^\top, \ldots, \hat{\boldsymbol{\zeta}}_K^\top]^\top$ is the $KT \times 1$ Lagrangian vector in the Fourier domain. Equation 5 can be solved iteratively using the ADMM [4] technique. Each of the subproblems, $\hat{\mathbf{g}}^*$ and $\mathbf{h}^*$, have closed form solutions.

Subproblem $\mathbf{h}^*$:

$$\begin{aligned} \mathbf{h}^* &= \arg\min_{\mathbf{h}} \Big\{ \frac{\lambda}{2}||\mathbf{h}||_2^2 + \hat{\boldsymbol{\zeta}}^\top(\hat{\mathbf{g}} - \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}) \\ &+ \frac{\mu}{2}||\hat{\mathbf{g}} - \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}||_2^2 \Big\} \\ &= (\mu + \frac{\lambda}{\sqrt{T}})^{-1}(\mu\mathbf{g} + \boldsymbol{\zeta}) \end{aligned} \tag{6}$$

where $\mathbf{g} = \frac{1}{\sqrt{T}}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\hat{\mathbf{g}}$ and $\boldsymbol{\zeta} = \frac{1}{\sqrt{T}}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\hat{\boldsymbol{\zeta}}$. The Kronecker product with the identity matrix can be broken into K independent Inverse Fast Fourier Transform (IFFT) computations of $\mathbf{g}_k = \frac{1}{\sqrt{T}}\mathbf{F}\mathbf{P}^\top\hat{\mathbf{g}}_k$ and $\boldsymbol{\zeta}_k = \frac{1}{\sqrt{T}}\mathbf{F}\mathbf{P}^\top\hat{\boldsymbol{\zeta}}_k$. In practice, both $\mathbf{g}_k$ and $\boldsymbol{\zeta}_k$ can be estimated efficiently by applying an IFFT on each $\hat{\mathbf{g}}_k$ and $\hat{\boldsymbol{\zeta}}_k$ and then applying the lookup table formed from the masking matrix $\mathbf{P}$. The over-complete vectors $\mathbf{g}$ and $\boldsymbol{\zeta}$ can be obtained by concatenating $\{\mathbf{g}_k\}_{k=1}^K$ and $\{\boldsymbol{\zeta}_k\}_{k=1}^K$, respectively. The computation of Eq. 6 is bounded by $\mathcal{O}(KT\log(T))$, where K is the number of channels, and $T\log(T)$ is the cost of computing the IFFT of a signal with the length of T.

Subproblem $\hat{\mathbf{g}}^*$:

$$\begin{aligned} \hat{\mathbf{g}}^* &= \arg\min_{\hat{\mathbf{g}}} \Big\{ \frac{1}{2}||\hat{\mathbf{y}} - \hat{\mathbf{X}}\hat{\mathbf{g}}||_2^2 \\ &+ \hat{\boldsymbol{\zeta}}^\top(\hat{\mathbf{g}} - \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}) \\ &+ \frac{\mu}{2}||\hat{\mathbf{g}} - \sqrt{T}(\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}||_2^2 \Big\} \end{aligned} \tag{7}$$

Solving Eq. 7 directly is $\mathcal{O}(T^3K^3)$. This computation is intractable for real-time tracking, since we need to solve for $\hat{\mathbf{g}}^*$ at every ADMM iteration. Fortunately, $\hat{\mathbf{X}}$ is sparse banded, and thus, each element of $\hat{\mathbf{y}}$ ($\hat{\mathbf{y}}(t),\ t = 1, ..., T$) is dependent only on K values of $\hat{\mathbf{x}}(t) = [\hat{\mathbf{x}}_1(t), ..., \hat{\mathbf{x}}_K(t)]^\top$ and $\hat{\mathbf{g}}(t) = [\text{conj}(\hat{\mathbf{g}}_1(t)), ..., \text{conj}(\hat{\mathbf{g}}_K(t))]^\top$ [16]. The operator conj(.) applies the complex conjugate to a complex vector/number. Therefore, solving Eq. 7 for $\hat{\mathbf{g}}^*$ can be identically expressed as T smaller, independent objectives, solving for $\hat{\mathbf{g}}(t)^*$, over $t = [1, ..., T]$:

$$\begin{aligned} \hat{\mathbf{g}}(t)^* &= \arg\min_{\hat{\mathbf{g}}(t)} \Big\{ \frac{1}{2}||\hat{\mathbf{y}}(t) - \hat{\mathbf{x}}(t)^\top\hat{\mathbf{g}}(t)||_2^2 \\ &+ \hat{\boldsymbol{\zeta}}(t)^\top(\hat{\mathbf{g}}(t) - \hat{\mathbf{h}}(t)) \\ &+ \frac{\mu}{2}||\hat{\mathbf{g}}(t) - \hat{\mathbf{h}}(t)||_2^2 \Big\} \end{aligned} \tag{8}$$

where $\hat{\mathbf{h}}(t) = [\hat{\mathbf{h}}_1(t), ..., \hat{\mathbf{h}}_K(t)]$ and $\hat{\mathbf{h}}_k = \sqrt{D}\mathbf{F}\mathbf{P}^\top\mathbf{h}_k$. In practice, each $\hat{\mathbf{h}}_k$ can be estimated efficiently by applying a FFT to each $\mathbf{h}_k$ padded with zeros. The solution for each $\hat{\mathbf{g}}(t)^*$ is obtained by:

$$\begin{aligned} \hat{\mathbf{g}}(t)^* = &\left(\hat{\mathbf{x}}(t)\hat{\mathbf{x}}(t)^\top + T\mu\mathbf{I}_K\right)^{-1} \\ &\left(\hat{\mathbf{y}}(t)\hat{\mathbf{x}}(t) - T\hat{\boldsymbol{\zeta}}(t) + T\mu\hat{\mathbf{h}}(t)\right) \end{aligned} \tag{9}$$

Eq. 9 has the complexity of $\mathcal{O}(TK^3)$, since we still need to solve T independent $K \times K$ linear systems. Even though this computation is substantially smaller than directly solving ($\mathcal{O}(T^3K^3)$), it is still intractable for real-time tracking.

Real-Time Extension: We propose to compute $\left(\hat{\mathbf{x}}(t)\hat{\mathbf{x}}(t)^\top + T\mu\mathbf{I}_K\right)^{-1}$ rapidly using the Sherman-Morrison formula [30], stating that $(\mathbf{u}\mathbf{v}^\top + \mathbf{A})^{-1} =$

$\mathbf{A}^{-1} - (\mathbf{v}^\top \mathbf{A}^{-1}\mathbf{u})^{-1}\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^\top\mathbf{A}^{-1}$, where in our case, $\mathbf{A} = T\mu\mathbf{I}_K$ and $\mathbf{u} = \mathbf{v} = \hat{\mathbf{x}}(t)$. Hence, Eq. 9 can be rewritten as:

$$\begin{aligned}\hat{\mathbf{g}}(t)^* &= \frac{1}{\mu}\left(T\hat{\mathbf{y}}(t)\hat{\mathbf{x}}(t) - \hat{\boldsymbol{\zeta}}(t) + \mu\hat{\mathbf{h}}(t)\right) \\ &\quad -\frac{\hat{\mathbf{x}}(t)}{\mu b}\left(T\hat{\mathbf{y}}(t)\hat{s}_{\mathbf{x}}(t) - \hat{s}_{\boldsymbol{\zeta}}(t) + \mu\hat{s}_{\mathbf{h}}(t)\right)\end{aligned} \tag{10}$$

where, $\hat{s}_{\mathbf{x}}(t) = \hat{\mathbf{x}}(t)^\top\hat{\mathbf{x}}$, $\hat{s}_{\boldsymbol{\zeta}}(t) = \hat{\mathbf{x}}(t)^\top\hat{\boldsymbol{\zeta}}$, $\hat{s}_{\mathbf{h}}(t) = \hat{\mathbf{x}}(t)^\top\hat{\mathbf{h}}$ and $b = \hat{s}_{\mathbf{x}}(t) + T\mu$ are scalar. The cost of computing $\hat{\mathbf{g}}$ using Eq. 10 is $\mathcal{O}(TK)$, whichi is much smaller than the computation of Eq. 9 ($\mathcal{O}(TK^3)$).

Lagrangian Update: We update the Lagrangians as

$$\hat{\boldsymbol{\zeta}}^{(i+1)} \leftarrow \hat{\boldsymbol{\zeta}}^{(i)} + \mu(\hat{\mathbf{g}}^{(i+1)} - \hat{\mathbf{h}}^{(i+1)}) \tag{11}$$

where $\hat{\mathbf{g}}^{(i+1)}$ and $\hat{\mathbf{h}}^{(i+1)}$ are the current solutions to the above subproblems at iteration $i+1$ within the iterative ADMM, and $\hat{\mathbf{h}}^{(i+1)} = (\mathbf{F}\mathbf{P}^\top \otimes \mathbf{I}_K)\mathbf{h}^{(i+1)}$. A common scheme for selecting μ is $\mu^{(i+1)} = \min(\mu_{\max}, \beta\mu^{(i)})$ [4].

Online Update: Similar to other CF trackers [13, 3, 1], we utilize an online adaptation strategy to improve our robustness to pose, scale and illumination changes. The online adaptation at frame f is formulated as $\hat{\mathbf{x}}_{model}^{(f)} = (1-\eta)\quad \hat{\mathbf{x}}_{model}^{(f-1)} + \eta\quad \hat{\mathbf{x}}^{(f)}$, where η is the online adaptation rate. Based on this strategy, we use $\hat{\mathbf{x}}_{model}^{(f)}$ instead of $\hat{\mathbf{x}}^{(f)}$ in Eq. 10 to compute $\hat{\mathbf{g}}(t)^*$, $\hat{s}_{\mathbf{x}}(t)$, $\hat{s}_{\boldsymbol{\zeta}}(t)$ and $\hat{s}_{\mathbf{h}}$.

Detection: The spatial location of the target in frame f is detected by applying the filter $\hat{\mathbf{g}}^{(f-1)}$ that has been updated in the previous frame. Following [10, 1], the filter is applied on multiple resolutions of the searching area to estimate scale changes. The searching area has the same spatial size of the filter $\hat{\mathbf{g}}$. This returns S correlation outputs, where S is the number of scales. We employ the interpolation strategy in [10, 11] to maximize detection scores per each correlation output. The scale with the maximum correlation output is used to update the object location and scale.

5. Experiments

We extensively evaluate our tracker on four standard datasets, including OTB50 [36], OTB100 [37], TempleColor128 (TC128) [23], and VOT2015 [18], comparing with 24 state-of-the-art methods, such as TLD [15], Struck [12], CFLB [17], KCF [13], DSST [7], SAMF [22], MEEM [38], DAT [29], LCT [27], HCF [26], Staple [1], SRDCF [10], SRDCFdecon [8], DeepSRDCF [9], CCOT [11], S3Tracker [21], SC-EBT [35], LDP [25], SiamFC [2], MDNet [28], STCT [34], YCNN [5], SINT [32] and FCNT [33].

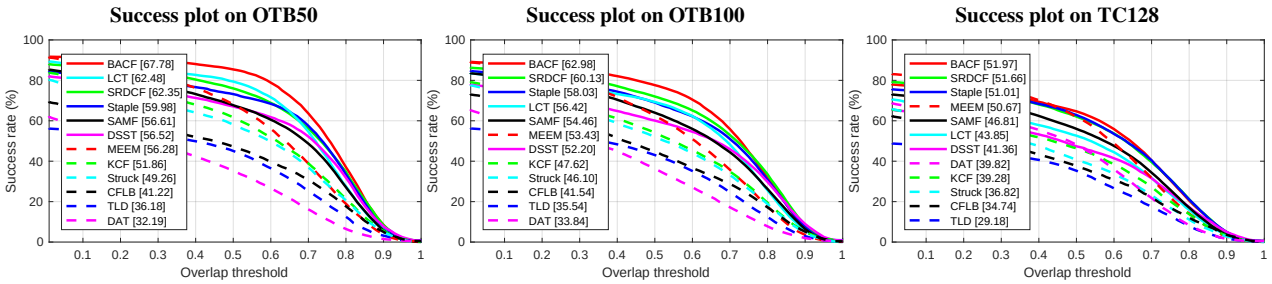
Evaluation Methodology: We use the success metric [36] to evaluate all trackers on OTB50, OTB100 and TC128. Success measures the intersection over union (IoU) of predicted and ground truth bounding boxes. The success plot shows the percentage of bounding boxes whose IoU score is larger than a given threshold. We use the Area Under the Curve (AUC) of success plots to rank the trackers. We also compare all the trackers by the success rate at the conventional thresholds of 0.50 (IoU $>$ 0.50) [36]. For the VOT15 dataset, tracking performance is evaluated in terms of accuracy (overlap with the ground-truth) and robustness (failure rate) [18]. In VOT2015, a tracker is restarted in the case of a failure, where there is no overlap between the detected bounding box and ground truth. For a full treatment of these metrics, readers are encouraged to read [18, 36].

Comparison Scenarios: We evaluate the BACF tracker over four experiments. The first experiment is conducted to show the superiority of our method to the state-of-the-art trackers with hand-crafted features (HOG). At the second experiment, we compare BACF with CF trackers with deep features to demonstrate compared to such methods BACF offers very competitive accuracy with almost two orders of magnitude (170 times) improvement in tracking speed. The third experiment compares our method with the state of the art deep trackers, and the last experiment compares our tracker with leading methods of the VOT2015 challenge.

Implementation Details: Similar to recent CF trackers [10, 13, 1] we employ 31-channel HOG features [6] using 4×4 cell size multiplied by a Hann window [3]. The regularization factor, λ, is set to 0.001, and the number of scales (S) is set to 5 with an scale-step of 1.01. A 2D Gaussian function with bandwidth of $\sqrt{wh}/16$ is used to define the correlation output for an object of size $[h, w]$. For the ADMM optimization, we set the number of iterations and the penalty factor, μ, to 2 and 1, respectively. The penalty factor at iteration $i+1$ is updated by $\mu^{(i+1)} = \min(\mu_{\max}, \beta\mu^{(i)})$, where $\beta = 10$ and $\mu_{\max} = 10^3$. We tested different configurations of μ and β, and we observed that choosing large values of β and μ over very few iterations helps the ADMM to converge much faster than smaller β and μ over more ADMM iterations. This substantially saves learning complexity with almost the same tracking accuracy. The learning (adaptation) rate of BACF $\eta = 0.0125$ for all experiments. We tested our MATLAB implementation on a machine equipped with an Intel Core i7 running at 2.40 GHz.

5.1. Comparison with HOG-based Trackers

Fig. 3 and Table 1 compare the BACF method with the state-of-the-art HOG-based trackers on the OTB50, OTB100 and TC128 datasets, where our method achieved the highest accuracy over all three datasets. More particularly, BACF achieved the best AUC (67.78) on OTB50 fol-



(a) (b) (c)

Figure 3. Success plots comparing BACF with the state-of-the-art HOG based trackers on (a) OTB50, (b) OTB100, and (c) TC128. We also include CFLB as the latest pixel intensities-based CF tracker. AUCs are reported in brackets.

Table 1. Success rates (% at IoU > 0.50) of BACF versus HOG-based trackers. The first, second and third best methods are shown in color.

	BACF	SRDCF	Staple	LCT	SAMF	MEEM	DSST	KCF	Struck	CFLB	TLD	DAT
OTB50	85.4	76.0	73.2	79.4	67.7	68.2	67.1	61.8	58.2	47.3	45.1	35.2
OTB100	77.6	72.0	69.1	69.3	64.0	62.6	60.1	54.2	52.2	44.7	43.1	36.3
TC128	65.2	62.1	62.9	52.6	56.0	62.0	47.4	46.4	40.7	37.7	35.3	48.1
Avg. succ. rate	76.0	70.0	68.4	67.1	62.5	64.2	58.2	54.1	50.3	43.2	41.1	39.8
Avg. FPS	35.3	3.8	48.3	18.5	11.4	11.1	17.7	173.4	9.2	87.1	22.1	60.3

lowed by LCT (62.48) and SRDCF (62.35). On OTB100, BACF (62.98) outperformed SRDCF (60.13) and Staple (58.03). BACF (51.97) is also the winner of the comparison on TC128, which is closely followed by SRDCF (51.66) and Staple (51.01). This result demonstrates the importance of utilizing background patches to learn more robust CF trackers from hand-crafted features. This evaluation also shows that BACF's strategy is more efficient than that of SRDCF to learn robust CF trackers from background. This is mainly because- unlike BACF- SRDCF has two crucial parameters (the minimum value of each weight and the impact of regularizer) to compute a regularization weight for each pixel. Since these two parameters are fixed for all videos, frames and pixels, there is no guarantee of delivering optimal results for all scenarios. Table 1 reports the average success rates of all trackers (IoU > 0.50) as well as their tracking speed (FPS) on CPUs. The best tracking speed belongs to KCF (173.4 FPS) followed by CFLB (87.1 FPS), DAT (60.3 FPS) and Staple (48.3 FPS). Higher speed of such trackers, however, came at the cost of much lower accuracy compared to BACF. Our method obtained the real-time speed of 35.3 FPS which is almost 10 times faster than SRDCF (the second best tracker).

Attribute Based Evaluation: Fig. 4 illustrates the attribute based evaluation of all HOG-based trackers on OTB100. All sequences in OTB100 are manually annotated by 11 different visual attributes such as occlusion, deformation and motion blur. We only reported the results of 6 attributes and full evaluation can be found in the supplemental material. The results show our superior tracking performance on all attributes. This empirically demonstrates how learning CFs from a huge set of background patches improves the stability of such trackers against challenging photometric and geometric variations. Trackers such as SRDCF, Staple and MEEM showed to be less robust to out of view, occlusion, deformation, respectively.

Robustness to Initialization: Following [10, 9, 1], we evaluated our tracker towards different spatial and temporal initializations [37] using two robustness metrics: spatial robustness (SRE) and temporal robustness (TRE). SRE measures the sensitivity of a tracker against noisy initialization (small perturbations from the ground truth). TRE measures the sensitivity of a tracker when initialized at different frames of the sequence. Fig. 5 shows the TRE and SRE success plots of BACF compared with the other HOG based trackers (SRE and TRE of DAT and Struck are not available). Our method achieved the best TRE and SRE AUCs followed by SRDCF and Staple. This evaluation shows that compared to other HOG based methods our method is more robust to different spatial and temporal initializations.

5.2. Comparing with Deep Feature-based Trackers

Table 7 compares BACF with the state of the art CF trackers with deep features, showing that BACF achieved the best success rate on OTB50, and the second best accuracy on OTB100 and TC128 after CCOT. Overall, BACF (76.0) outperformed HCF (65.0) and DeepSRDCDF (72.9) and obtained very comparable average success rate to CCOT (77.8). In terms of tracking speed, however, BACF dramatically outperformed the other trackers, with almost

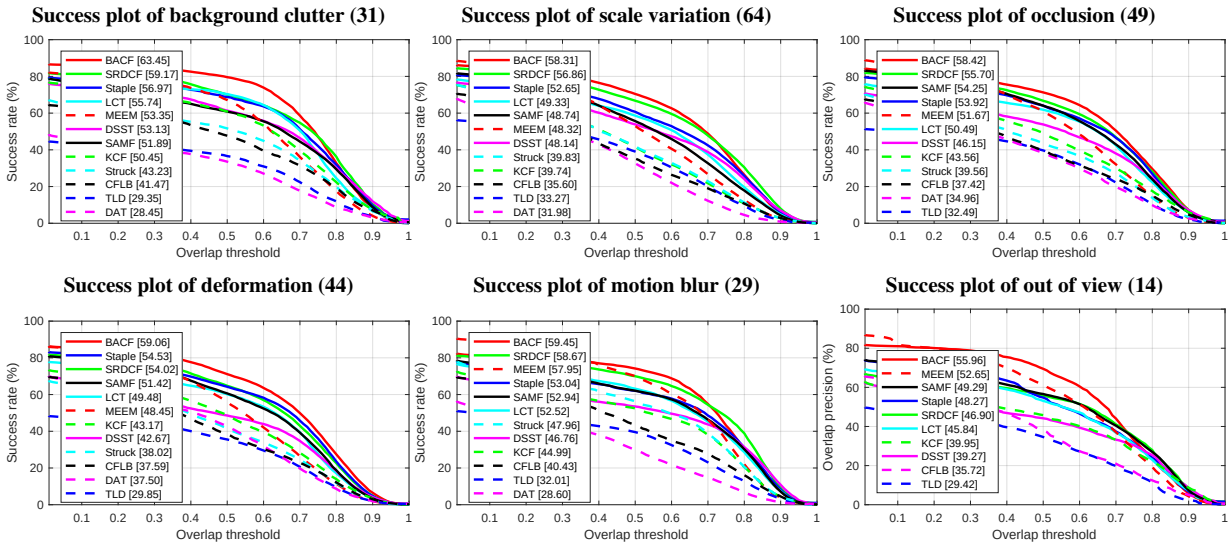


Figure 4. Attribute based evaluation. Success plots compare BACF with state-of-the-art HOG based trackers on OTB100. BACF outperformed all the trackers over all attributes. AUCs are reported in brackets. The number of videos for each attribute is shown in parenthesis.

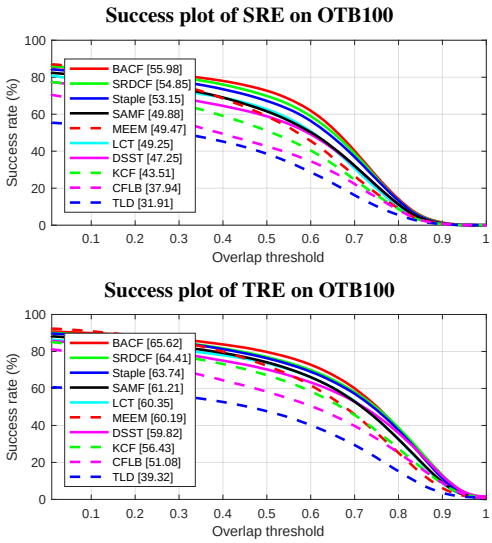


Figure 5. SRE TRE evaluation: Success plots comparing BACF with state-of-the-art HOG based trackers on OTB100.

170 times faster tracking speed. Despite the efficient optimization of CFs in the Fourier domain, CF trackers with deep features suffer from intractable complexity which is mainly caused by 1) computationally expensive deep feature extraction on CPUs, and 2) computing the FFT/IFFT of hundreds of deep feature channels at each frame.

To emphasise the superior tracking speed of BACF and its competitive accuracy to CCOT, we directly compared these two methods on OTB100 sequences in terms of the number of videos these trackers show superior accuracy, and relative tracking speed for each sequence. Results in Fig. 6 show that for 37 videos (of 101 videos) BACF outperformed CCOT, and for 41 videos CCOT achieved superior accuracy. Both CCOT and BACF showed the same accuracy on 23 sequences. This comparison highlights the competitive accuracy of these methods. In terms of relative speed, BACF showed at least 100 times faster speed on all videos. For some sequences with smaller targets, such as Freeman4, FaceOcc2 and Twinnings, BACF is almost 400 times faster than CCOT.

Table 2. Success rates (% at IoU $>$ 0.50) of BACF compared to CF trackers with deep features. The first, second and third highest rates are highlighted in color.

	BACF	HCF	DeepSRDCF	CCOT
OTB50	85.4	72.1	77.4	81.6
OTB100	77.6	64.8	76.4	81.5
TC128	65.2	58.1	64.9	70.5
Avg. succ. rate	76.0	65.0	72.9	77.8
Avg. FPS	35.3	0.8	0.4	0.2

5.3. Evaluation on VOT2015

Table 3 shows the comparison of our method with the top 5 participants in the VOT2016 challenge[1], CCOT, Staple and DSST on 60 challenging videos of VOT15. Our method achieved the best accuracy (0.56) by improving 5% of the accuracy obtained by SRDCF and DeepSRDCF. The highest robustness (0.82) belongs to CCOT, followed by DeepSRDCF (1.05) and SRDCF (1.24). Our tracker significantly

[1] http://www.votchallenge.net/vot2016/

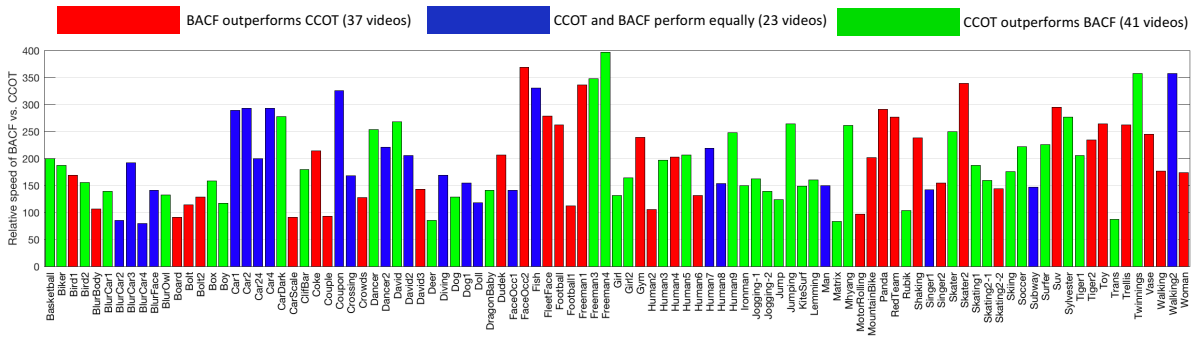


Figure 6. Comparing BACF with CCOT in terms of speed up on OTB100. We also show performance details over which tracker performs best on a given sequence. Sequences coded red are where BACF outperforms CCOT. Sequences coded green are when CCOT outperforms BACF, and blue indicates equal performance. Across the board, we note that BACF provides an appreciable speedup (on average around 200x) compared to CCOT. CCOT, however, only outperforms BACF in terms of tracking accuracy on a small portion of the videos. For most videos, BACF performs as well or better than CCOT.

Table 3. Evaluation on VOT2015 by the means of robustness and accuracy.

	Ours	S3Tracker	Struck	SC-EBT	LDP	DSST	DAT	Staple	SRDCF	DeepSRDCF	CCOT
Acc.	0.59	0.52	0.47	0.55	0.51	0.49	0.44	0.53	0.56	0.56	0.54
Rob.	1.56	1.77	1.26	1.86	1.84	2.53	2.06	1.35	1.24	1.05	0.82

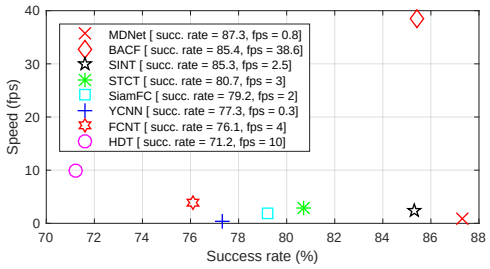


Figure 7. Comparing our method with deep trackers in terms of tracking speed (fps) and success rate at IoU > 0.50.

improved the accuracy and robustness of the top participants of VOT2016 (S3Tracker, SC-EBT, LDP and DSST).

5.4. Comparing with Deep Trackers

We also compared our tracker against recent deep trackers, including SiamFC [2], MDNet [28], STCT [34], YCNN [5], SINT [32] and FCNT [33]. Results in Fig. 7, show our superior real-time performance. Surprisingly, our accuracy (85.4) is very competitive with MDNet (87.3), and our method outperformed SINT (85.3), SiamFC (79.2), STCT (80.7), YCNN (77.3) and FCNT (76.1).

Qualitative Results: Fig. 8 shows some qualitative results.

6. Conclusion

In this work, we proposed background aware correlation filters for the task of visual tracking. Compared to current CF trackers which are trained by shifted patches, our

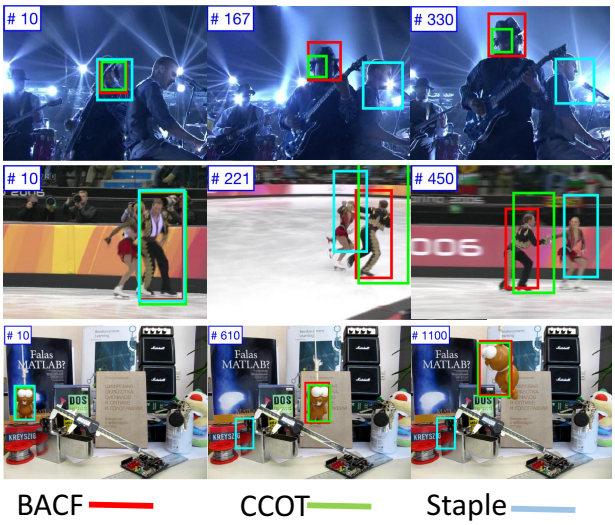


Figure 8. Qualitative comparison of our approach with state-of-the-art trackers on the Shaking, Skating and Lemming videos. Our approach provides consistent results in challenging scenarios, such as illumination change, fast motion and background clutter.

method exploits real background patches together with the target patch to learn the tracker. Moreover, we utilized an online adaptation strategy to update the tracker model respect to the new appearance of the target and background over time. Learning from real patches over online adaptation significantly improved the robustness of our method against challenging deformation, scaling, and background clutter. We demonstrated the competitive accuracy and su-

perior tracking speed of our method compared to recent CF-based and deep trackers over an extensive evaluation.

References

[1] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, June 2016. 2, 5, 6

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865. Springer, 2016. 2, 5, 8

[3] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550. IEEE, 2010. 1, 2, 5

[4] S. Boyd. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2010. 4, 5

[5] K. Chen and W. Tao. Once for all: a two-flow convolutional neural network for visual tracking. *arXiv preprint arXiv:1604.07507*, 2016. 2, 5, 8

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005. 1, 2, 5

[7] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*. BMVA Press, 2014. 2, 5

[8] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. CVPR, 2016. 2, 5

[9] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 58–66, 2015. 1, 2, 5, 6

[10] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, pages 4310–4318, 2015. 1, 2, 5, 6

[11] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, pages 472–488. Springer, 2016. 1, 2, 5

[12] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270. IEEE, 2011. 5

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 37(3):583–596, 2015. 1, 2, 5

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 2

[15] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 34(7):1409–1422, 2012. 5

[16] H. Kiani Galoogahi, T. Sim, and S. Lucey. Multi-channel correlation filters. In *ICCV*, pages 3072–3079, 2013. 2, 3, 4

[17] H. Kiani Galoogahi, T. Sim, and S. Lucey. Correlation filters with limited boundaries. In *CVPR*, pages 4630–4638, 2015. 1, 2, 5

[18] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–23, 2015. 5

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[20] B. V. Kumar. *Correlation pattern recognition*. Cambridge University Press, 2005. 1, 4

[21] J.-Y. Lee and W. Yu. Visual tracking by partition-based histogram backprojection and maximum support criteria. In *Robotics and Biomimetics (ROBIO)*, pages 2860–2865. IEEE, 2011. 5

[22] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV*, pages 254–265. Springer, 2014. 2, 5

[23] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: algorithms and benchmark. *TIP*, 24(12):5630–5644, 2015. 1, 5

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2

[25] A. Lukežič, L. Čehovin, and M. Kristan. Deformable parts correlation filters for robust visual tracking. *arXiv preprint arXiv:1605.03720*, 2016. 5

[26] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, pages 3074–3082, 2015. 2, 5

[27] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, pages 5388–5396, 2015. 5

[28] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. *arXiv preprint arXiv:1510.07945*, 2015. 1, 2, 5, 8

[29] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *CVPR*, pages 2113–2120, 2015. 5

[30] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950. 5

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[32] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. *arXiv preprint arXiv:1605.05863*, 2016. 2, 5, 8

[33] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, pages 3119–3127, 2015. 1, 2, 5, 8

[34] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. CVPR, 2016. 2, 5, 8

[35] N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *ICML*, pages 1107–1115, 2014. 5

[36] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418, 2013. 5

[37] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *PAMI*, 37(9):1834–1848, 2015. 1, 5, 6

[38] J. Zhang, S. Ma, and S. Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *ECCV*, pages 188–203. Springer, 2014. 5