ECO: Efficient Convolution Operators for Tracking

Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, Michael Felsberg
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Sweden
{martin.danelljan, goutam.bhat, fahad.khan, michael.felsberg}@liu.se

Abstract

In recent years, Discriminative Correlation Filter (DCF) based methods have significantly advanced the state-of-the-art in tracking. However, in the pursuit of ever increasing tracking performance, their characteristic speed and real-time capability have gradually faded. Further, the increasingly complex models, with massive number of trainable parameters, have introduced the risk of severe over-fitting. In this work, we tackle the key causes behind the problems of computational complexity and *over-fitting, with the aim of simultaneously improving* both *speed and performance.*

We revisit the core DCF formulation and introduce: (i) a factorized convolution operator, which drastically reduces the number of parameters in the model; (ii) a compact generative model of the training sample distribution, that significantly reduces memory and time complexity, while providing better diversity of samples; (iii) a conservative model update strategy with improved robustness and reduced complexity. We perform comprehensive experiments on four benchmarks: VOT2016, UAV123, OTB-2015, and TempleColor. When using expensive deep features, our tracker provides a 20-fold speedup and achieves a 13.3% *relative gain in Expected Average Overlap compared to the top ranked method [9] in the VOT2016 challenge. Moreover, our fast variant, using hand-crafted features, operates at 60 Hz on a single CPU, while obtaining* 64.8% *AUC on OTB-2015.*

1. Introduction

Generic visual tracking is one of the fundamental problems in computer vision. It is the task of estimating the trajectory of a target in an image sequence, given only its initial state. Online visual tracking plays a crucial role in numerous real-time vision applications, such as smart surveillance systems, autonomous driving, UAV monitoring, intelligent traffic control, and human-computer-interfaces. Due to the online nature of tracking, an ideal tracker should be accurate and robust under the hard computational constraints of real-time vision systems.

In recent years, Discriminative Correlation Filter (DCF) based approaches have shown continuous performance improvements in terms of accuracy and robustness on tracking benchmarks [21, 35]. The recent advancement in DCF based tracking performance is driven by the use of multi-dimensional features [12, 10], robust scale estimation [5], non-linear kernels [18], long-term memory components [26], sophisticated learning models [8, 3] and reducing boundary effects [7, 13]. However, these improvements in accuracy come at the price of significant reductions in tracking speed. For instance, the pioneering MOSSE tracker by Bolme *et al.* [4] is about 1000× faster than the recent top-ranked DCF tracker [9] in the VOT2016 challenge [21], but obtains only half the accuracy.

Several milestones in the advancement of the DCF paradigm are due to powerful learning formulations [18, 12, 7]. Most importantly, the DCF formulation has been extended to mult-dimensional feature maps, enabling discriminative image representations such as HOG [18, 5] and deep features [25, 6]. Recently, Danelljan *et al.* [9] introduced a continuous-domain formulation of the DCF, called C-COT, that enables integration of multi-resolution deep

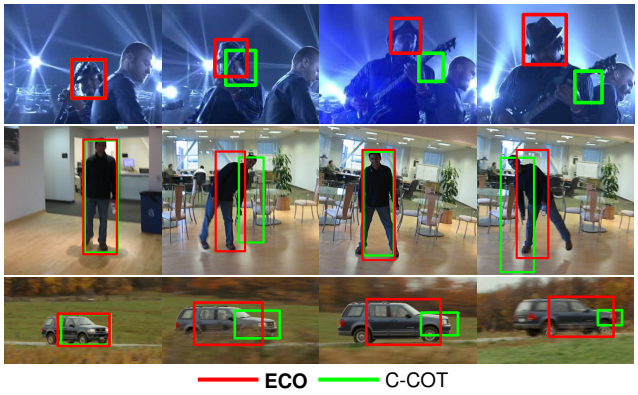


Figure 1. A comparison of our approach ECO with the baseline C-COT [9] on three example sequences. In all three cases, C-COT suffers from severe over-fitting to particular regions of the target. This causes poor target estimation in cases of partial occlusion (the guitar in top row), deformations (middle row) and out-of-plane rotations (bottom row). ECO successfully tackles the causes of over-fitting, leading to better generalization of the target appearance.

feature maps, leading to top performance in the VOT2016 challenge [21]. On the other hand, this gain is obtained by sacrificing the real-time capabilities that has been the hallmark of early DCF-based trackers [4, 17, 10].

We identify three key factors that contribute to *both* increased computational complexity and over-fitting in state-of-the-art DCF trackers.

Model size: The straight-forward integration of multi-dimensional feature maps into DCF-based trackers leads to a radical increase of the number of parameters in the appearance model, often beyond the dimensionality of the input. As an example, C-COT [9] continuously updates about 800,000 parameters during the online learning of the model. Due to the inherent scarcity of training data in tracking, such a high-dimensional parameter space is prone to over-fitting. Further, the high dimensionality causes an increase in the computational complexity, leading to slow tracking speed.

Training set size: State-of-the-art DCF trackers, including C-COT, require a large training sample set to be stored due to their reliance on iterative optimization algorithms. In practice however, the memory size is limited, particularly when using high-dimensional features. A typical strategy for maintaining a feasible memory consumption is to discard the oldest samples. This may however cause over-fitting to recent appearance changes, leading to model drift (see figure 1). Moreover, a large training set increases the computational burden.

Model update: Most DCF-based trackers apply a continuous learning strategy, where the model is updated rigorously in every frame. On the contrary, recent works have shown impressive performance without any model update, using Siamese networks [2]. Motivated by these findings, we argue that the continuous model update in state-of-the-art DCF is excessive and sensitive to sudden changes caused by, *e.g.*, occlusions, deformations, and out-of-plane rotations (see figure 1). This excessive update strategy causes both lower frame-rates and degradation of robustness due to over-fitting to the recent frames.

1.1. Contributions

We propose a novel formulation that addresses the previously listed issues of state-of-the-art DCF trackers. As our first contribution, we introduce a factorized convolution operator that dramatically reduces the number of parameters in the DCF model. Our second contribution is a compact generative model of the training sample space that effectively reduces the number of samples in the learning, while maintaining their diversity. As our final contribution, we introduce an efficient model update strategy, that simultaneously improves tracking speed and robustness.

Comprehensive experiments clearly demonstrate that our approach concurrently improves both tracking performance and speed, thereby setting a new state-of-the-art on four benchmarks: VOT2016, UAV123, OTB-2015, and TempleColor. Our approach significantly reduces the number of model parameters by 80%, training samples by 90% and optimization iterations by 80% in the learning, compared to the baseline. On VOT2016, our approach outperforms the top ranked tracker, C-COT [9], in the challenge, while achieving a significantly higher frame-rate. Furthermore, we propose a fast variant of our tracker that maintains competitive performance with a speed of 60 frames per second (FPS) on a single CPU, thereby being especially suitable for computationally restricted robotics platforms.

2. Baseline Approach: C-COT

In this work, we collectively address the problems of computational complexity and over-fitting in state-of-the-art DCF trackers. We adopt the recently introduced Continuous Convolution Operator Tracker (C-COT) [9] as our baseline. The C-COT obtained the top rank in the recent VOT2016 challenge [21], and has also demonstrated superior results on other tracking benchmarks [35, 24]. Unlike the standard DCF formulation, Danelljan *et al*. [9] pose the problem of learning the filters in the continuous spatial domain. The generalized formulation in C-COT yields two advantages that are relevant to our work.

The first advantage of C-COT is the natural integration of multi-resolution feature maps, achieved by performing convolutions in the continuous domain. This provides the flexibility of choosing the cell size (*i.e.* resolution) of each visual feature independently, without the need for explicit re-sampling. The second advantage is that the predicted detection scores of the target are directly obtained as a continuous function, enabling accurate sub-grid localization.

Here, we briefly describe the C-COT formulation, adopting the same notation as in [9] for convenience. The C-COT discriminatively learns a convolution filter based on a collection of M training samples $\{x_j\}_1^M \subset \mathcal{X}$. Unlike the standard DCF, each feature layer $x_j^d \in \mathbb{R}^{N_d}$ has an independent resolution N_d.[1] The feature map is transfered to the continuous spatial domain $t \in [0, T)$ by introducing an interpolation model, given by the operator J_d,

$$J_d\{x^d\}(t) = \sum_{n=0}^{N_d-1} x^d[n] b_d \left(t - \frac{T}{N_d} n \right). \qquad (1)$$

Here, b_d is an interpolation kernel with period $T > 0$. The result $J_d\{x^d\}$ is thus an interpolated feature layer, viewed as a continuous T-periodic function. We use $J\{x\}$ to denote the entire interpolated feature map, where $J\{x\}(t) \in \mathbb{R}^D$.

In the C-COT formulation, a continuous T-periodic multi-channel convolution filter $f = (f^1 \dots f^D)$ is trained

[1] For clarity, we present the one-dimensional domain formulation. The generalization to higher dimensions, including images, is detailed in [9].

to predict the detection scores $S_f\{x\}(t)$ of the target as,

$$S_f\{x\} = f * J\{x\} = \sum_{d=1}^{D} f^d * J_d\{x^d\}\,. \tag{2}$$

The scores are defined in the corresponding image region $t \in [0, T)$ of the feature map $x \in \mathcal{X}$. In (2), the convolution of single-channel T-periodic functions is defined as $f * g(t) = \frac{1}{T}\int_0^T f(t-\tau)g(\tau)\,\mathrm{d}\tau$. The multi-channel convolution $f * J\{x\}$ is obtained by summing the result of all channels, as defined in (2). The filters are learned by minimizing the following objective,

$$E(f) = \sum_{j=1}^{M} \alpha_j \left\| S_f\{x_j\} - y_j \right\|_{L^2}^2 + \sum_{d=1}^{D} \left\| w f^d \right\|_{L^2}^2 . \tag{3}$$

The labeled detection scores $y_j(t)$ of sample x_j is set to a periodically repeated Gaussian function. The data term consists of the weighted classification error, given by the L^2-norm $\|g\|_{L^2}^2 = \frac{1}{T}\int_0^T |g(t)|^2\,\mathrm{d}t$, where $\alpha_j \geq 0$ is the weight of sample x_j. The regularization integrates a spatial penalty $w(t)$ to mitigate the drawbacks of the periodic assumption, while enabling an extended spatial support [7].

As in previous DCF methods, a more tractable optimization problem is obtained by changing to the Fourier basis. Parseval's formula implies the equivalent loss,

$$E(f) = \sum_{j=1}^{M} \alpha_j \left\| \widehat{S_f\{x_j\}} - \hat{y}_j \right\|_{\ell^2}^2 + \sum_{d=1}^{D} \left\| \hat{w} * \hat{f}^d \right\|_{\ell^2}^2 . \tag{4}$$

Here, the hat $\hat{g}$ of a T-periodic function g denotes the Fourier series coefficients $\hat{g}[k] = \frac{1}{T}\int_0^T g(t) e^{-i\frac{2\pi}{T}kt}\,\mathrm{d}t$ and the ℓ^2-norm is defined by $\|\hat{g}\|_{\ell^2}^2 = \sum_{-\infty}^{\infty} |\hat{g}[k]|^2$. The Fourier coefficients of the detection scores (2) are given by the formula $\widehat{S_f\{x\}} = \sum_{d=1}^{D} \hat{f}^d X^d \hat{b}_d$, where X^d is the Discrete Fourier Transform (DFT) of x^d.

In practice, the filters f^d are assumed to have finitely many non-zero Fourier coefficients $\{\hat{f}^d[k]\}_{-K_d}^{K_d}$, where $K_d = \left\lfloor \frac{N_d}{2} \right\rfloor$. Eq. (4) then becomes a quadratic problem, optimized by solving the normal equations,

$$\left(A^{\mathrm{H}} \Gamma A + W^{\mathrm{H}} W\right) \hat{\mathbf{f}} = A^{\mathrm{H}} \Gamma \hat{\mathbf{y}}\,. \tag{5}$$

Here, $\hat{\mathbf{f}}$ and $\hat{\mathbf{y}}$ are vectorizations of the Fourier coefficients in f^d and y_j, respectively. The matrix A exhibits a sparse structure, with diagonal blocks containing elements of the form $X_j^d[k]\hat{b}_d[k]$. Further, Γ is a diagonal matrix of the weights α_j and W is a convolution matrix with the kernel $\hat{w}[k]$. The C-COT [9] employs the Conjugate Gradient (CG) method [30] to iteratively solve (5), since it was shown to effectively utilize the sparsity structure of (5).

3. Our Approach

As discussed earlier, over-fitting and computational bottlenecks in the DCF learning stem from common factors. We therefore proceed with a collective treatment of these issues, aiming at both improved performance *and* speed.

Robust learning: As mentioned earlier, the large number of optimized parameters in (3) may cause over-fitting due to limited training data. We alleviate this issue by introducing a factorized convolution formulation in section 3.1, which reduces the number of parameters in the model. We show that this strategy radically reduces the number of model parameters by 80% in the case of deep features, while increasing tracking performance. Moreover, we propose a compact generative model of the sample distribution in section 3.2, that boosts diversity and avoids the previously discussed problems related to storing a large sample set. Finally, we show in section 3.3 that a less frequent update strategy reduces over-fitting, leading to more robust tracking.

Computational complexity: The learning step is the computational bottleneck in optimization-based DCF trackers, such as C-COT. The computational complexity of the appearance model optimization in C-COT is obtained by analyzing the Conjugate Gradient algorithm applied to (5). The complexity can be expressed as $\mathcal{O}(N_{\mathrm{CG}} D M \bar{K})$,[2] where N_{CG} is the number of CG iterations and $\bar{K} = \frac{1}{D}\sum_d K_d$ is the average number of Fourier coefficients per filter channel. Motivated by this complexity analysis of the learning, we propose methods for reducing D, M and N_{CG} in sections 3.1, 3.2, and 3.3 respectively.

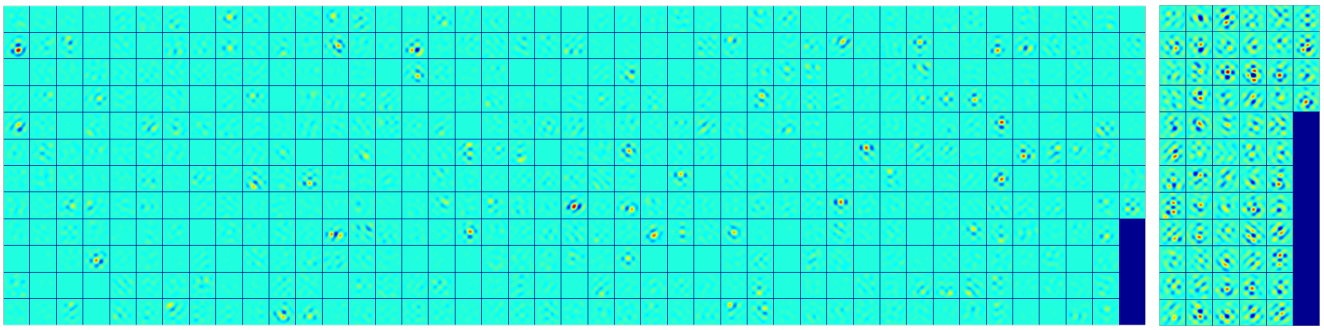
3.1. Factorized Convolution Operator

We first introduce a factorized convolution approach, with the aim of reducing the number of parameters in the model. We observed that many of the filters f^d learned in C-COT contain negligible energy. This is particularly apparent for high-dimensional deep features, as visualized in figure 2. Such filters hardly contribute to target localization, but still affect the training time. Instead of learning one separate filter for each feature channel d, we use a smaller set of basis filters $f^1, \ldots, f^C$, where $C < D$. The filter for feature layer d is then constructed as a linear combination $\sum_{c=1}^{C} p_{d,c} f^c$ of the filters f^c using a set of learned coefficients $p_{d,c}$. The coefficients can be compactly represented as a $D \times C$ matrix $P = (p_{d,c})$. The new multi-channel filter can then be written as the matrix-vector product Pf. We obtain the factorized convolution operator,

$$S_{Pf}\{x\} = Pf * J\{x\} = \sum_{c,d} p_{d,c} f^c * J_d\{x^d\} = f * P^{\mathrm{T}} J\{x\}. \tag{6}$$

The last equality follows from the linearity of convolution. The factorized convolution (6) can thus alternatively

[2] See the supplementary material for a derivation.



(a) C-COT (b) Ours

Figure 2. Visualization of the learned filters corresponding to the last convolutional layer in the deep network. We display all the 512 filters f^d learned by the baseline C-COT (a) and the reduced set of 64 filters f^c obtained by our factorized formulation (b). The vast majority of the baseline filters contain negligible energy, indicating irrelevant information in the corresponding feature layers. Our factorized convolution formulation learns a compact set of discriminative basis filters with significant energy, achieving a radical reduction of parameters.

be viewed as a two-step operation where the feature vector $J\{x\}(t)$ at each location t is first multiplied with the matrix P^{T}. The resulting C-dimensional feature map is then convolved with the filter f. The matrix P^{T} thus resembles a linear dimensionality reduction operator, as used in *e.g.* [10]. The key difference is that we learn the filter f and matrix P *jointly*, in a discriminative fashion, by minimizing the classification error (3) of the factorized operator (6).

For simplicity, we consider learning the factorized operator (6) from single training sample x. To simplify notation, we use $\hat{z}^d[k] = X^d[k]\hat{b}_d[k]$ to denote the Fourier coefficients of the interpolated feature map $z = J\{x\}$. The corresponding loss in the Fourier domain (4) is derived as,

$$E(f, P) = \left\| \hat{z}^{\mathrm{T}} P \hat{f} - \hat{y} \right\|_{\ell^2}^2 + \sum_{c=1}^{C} \left\| \hat{w} * \hat{f}^c \right\|_{\ell^2}^2 + \lambda \|P\|_F^2. \quad (7)$$

Here we have added the Frobenius norm of P as a regularization, controlled by the weight parameter λ.

Unlike the original formulation (4), our new loss (7) is a non-linear least squares problem. Due to the bi-linearity of $\hat{z}^{\mathrm{T}} P \hat{f}$, the loss (7) is similar to a matrix factorization problem [19]. Popular optimization strategies for these applications, including Alternating Least Squares, are however not feasible due to the parameter size and online nature of our problem. Instead, we employ Gauss-Newton [30] and use the Conjugate Gradient method to optimize the quadratic subproblems. The Gauss-Newton method is derived by linearizing the residuals in (7) using a first order Taylor series expansion. Here, this corresponds to approximating the bilinear term $\hat{z}^{\mathrm{T}} P \hat{f}$ around the current estimate $(\hat{f}_i, P_i)$ as,

$$\begin{aligned} \hat{z}^{\mathrm{T}}(P_i + \Delta P)(\hat{f}_i + \Delta \hat{f}) &\approx \hat{z}^{\mathrm{T}} P_i \hat{f}_{i,\Delta} + \hat{z}^{\mathrm{T}} \Delta P \hat{f}_i \quad (8) \\ &= \hat{z}^{\mathrm{T}} P_i \hat{f}_{i,\Delta} + (\hat{f}_i \otimes \hat{z})^{\mathrm{T}} \operatorname{vec}(\Delta P). \end{aligned}$$

Here, we set $\hat{f}_{i,\Delta} = \hat{f}_i + \Delta \hat{f}$. In the last equality, the Kronecker product $\otimes$ is used to obtain a vectorization of the matrix step ΔP.

The Gauss-Newton subproblem at iteration i is derived by substituting the first-order approximation (8) into (7),

$$\begin{aligned} \tilde{E}(\hat{f}_{i,\Delta}, \Delta P) = &\left\| \hat{z}^{\mathrm{T}} P_i \hat{f}_{i,\Delta} + (\hat{f}_i \otimes \hat{z})^{\mathrm{T}} \operatorname{vec}(\Delta P) - \hat{y} \right\|_{\ell^2}^2 \\ &+ \sum_{c=1}^{C} \left\| \hat{w} * \hat{f}^c_{i,\Delta} \right\|_{\ell^2}^2 + \lambda \|P_i + \Delta P\|_F^2. \quad (9) \end{aligned}$$

Since the filter f is constrained to have finitely many non-zero Fourier coefficients, eq. (9) is a linear least squares problem. The corresponding normal equations have a partly similar structure to (5), with additional components corresponding to the matrix increment ΔP variable.[3] We employ the Conjugate Gradient method to optimize each Gauss-Newton subproblem to obtain the new filter $\hat{f}^*_{i,\Delta}$ and matrix increment ΔP^*. The filter and matrix estimates are then updated as $\hat{f}_{i+1} = \hat{f}^*_{i,\Delta}$ and $P_{i+1} = P_i + \Delta P^*$.

The main objective of our factorized convolution operation is to reduce the computational and memory complexity of the tracker. Due to the adaptability of the filter, the matrix P can be learned just from the first frame. This has two important implications. Firstly, only the projected feature map $P^{\mathrm{T}} J\{x_j\}$ requires storage, leading to significant memory savings. Secondly, the filter can be updated in subsequent frames using the projected feature maps $P^{\mathrm{T}} J\{x_j\}$ as input to the method described in section 2. This reduces the linear complexity in the feature dimensionality D to the filter dimensionality C, *i.e.* $\mathcal{O}(N_{\mathrm{CG}} C M \bar{K})$.

3.2. Generative Sample Space Model

Here, we propose a compact generative model of the sample set that averts the earlier discussed issues of storing a large set of recent training samples. Most DCF trackers,

[3] See supplementary material for the derivation of the normal equations.

Figure 3. Visualization of the training set representation in the baseline C-COT (bottom row) and our method (top row). In C-COT, the training set consists of a sequence of consecutive samples. This introduces large redundancies due to slow change in appearance, while previous aspects of the appearance are forgotten. This can cause over-fitting to recent samples. Instead, we model the training data as a mixture of Gaussian components, where each component represent a different aspect of the appearance. Our approach yields a compact yet diverse representation of the data, thereby reducing the risk of over-fitting.

such as SRDCF [7] and C-COT [9], add one training sample x_j in each frame j. The weights are typically set to decay exponentially $\alpha_j \sim (1-\gamma)^{M-j}$, controlled by the learning rate γ. If the number of samples has reached a maximum limit M_{max}, the sample with the smallest weight α_j is replaced. This strategy however requires a large sample limit M_{max} to obtain a representative sample set.

We observe that collecting a new sample in each frame leads to large redundancies in the sample set, as visualized in figure 3. The standard sampling strategy (bottom row) populates the whole training set with similar samples x_j, despite containing almost the same information. Instead, we propose to use a probabilistic generative model of the sample set that achieves a compact description of the samples by eliminating redundancy and enhancing variety (top).

Our approach is based on the joint probability distribution $p(x, y)$ of the sample feature maps x and corresponding desired outputs scores y. Given $p(x, y)$, the intuitive objective is to find the filter that minimizes the expected correlation error. This is obtained by replacing (3) with

$$E(f) = \mathbb{E}\left\{ \|S_f\{x\} - y\|_{L^2}^2 \right\} + \sum_{d=1}^{D} \left\|w f^d\right\|_{L^2}^2 . \quad (10)$$

Here, the expectation $\mathbb{E}$ is evaluated over the joint sample distribution $p(x, y)$. Note that the original loss (3) is obtained as a special case by estimating the sample distribution as $p(x, y) = \sum_{j=1}^{M} \alpha_j \delta_{x_j, y_j}(x, y)$, where δ_{x_j, y_j} denotes the Dirac impulse at the training sample (x_j, y_j).[4] Instead, we propose to estimate a compact model of the sample distribution $p(x, y)$ that leads to a more efficient approximation of the expected loss (10).

We observe that the shape of the desired correlation output y for a sample x is predetermined, here as a Gaussian function. The label functions y_j in (3) only differ by a translation that aligns the peak with the target center. We can equivalently shift the feature map x to perform this alignment[5] and thus assume that the target is centered in the image region and that all $y = y_0$ are identical. Hence, the sample distribution can be factorized as $p(x, y) = p(x)\delta_{y_0}(y)$ and we only need to estimate $p(x)$. For this purpose we employ a Gaussian Mixture Model (GMM) such that $p(x) = \sum_{l=1}^{L} \pi_l \mathcal{N}(x; \mu_l, I)$. Here, L is the number of Gaussian components $\mathcal{N}(x; \mu_l, I)$, π_l is the prior weight of component l, and $\mu_l \in \mathcal{X}$ is its mean. The covariance matrix is set as I to avoid costly inference in the high-dimensional sample space.

To update the GMM, we use a simplified version of the online algorithm by Declercq and Piater [11]. Given a new sample x_j, we first initialize a new component m with $\pi_m = \gamma$ and $\mu_m = x_j$ (*concatenate* in [11]). If the number of components exceeds the limit L, we *simplify* the GMM. We discard a component if its weight π_l is below a threshold. Otherwise, we merge the two closest components k and l into a common component n [11],

$$\pi_n = \pi_k + \pi_l \quad , \quad \mu_n = \frac{\pi_k \mu_k + \pi_l \mu_l}{\pi_k + \pi_l} . \quad (11)$$

The required distance comparisons $\|\mu_k - \mu_l\|$ are efficiently computed in the Fourier domain using Parseval's formula. Finally, the expected loss (10) is approximated as,

$$E(f) = \sum_{l=1}^{L} \pi_l \left\|S_f\{\mu_l\} - y_0\right\|_{L^2}^2 + \sum_{d=1}^{D} \left\|w f^d\right\|_{L^2}^2 . \quad (12)$$

[4] We can without loss of generality assume the weights α_j sum to one.

[5] Implemented as pre-processing step in the Fourier domain.

Note that the Gaussian means μ_l and the prior weights π_l directly replace x_j and α_j, respectively, in (3). So, the same training strategy as described in section 2 can be applied.

The key difference in complexity compared to (3) is that the number of samples has decreased from M to L. In our experiments, we show that the number of components L can be set to $M/8$, while obtaining an improved tracking performance. Our sample distribution model $p(x, y)$ is combined with the factorized convolution from section 3.1 by replacing the sample x with the projected sample $P^{\mathrm{T}}Jx$. The projection does not affect our formulation since the matrix P is constant after the first frame.

3.3. Model Update Strategy

The standard approach in DCF based tracking is to update the model in each frame [4, 18, 7]. In C-COT, this implies optimizing (3) after each new sample is added, by iteratively solving the normal equations (5). Iterative optimization based DCF methods exploit that the loss function changes gradually between frames. The current estimate of the filter therefore provides a good initialization of the iterative search. Still, updating the filter in each frame have a severe impact on the computational load.

Instead of updating the model in a continuous fashion every frame, we use a sparser updating scheme. Intuitively, an optimization process should only be started once sufficient change in the objective has occurred. However, finding such conditions is non-trivial and may lead to unnecessarily complex heuristics. Moreover, optimality conditions based on the gradient of the loss (3), given by the residual of (5), are expensive to evaluate in practice. We therefore avoid explicitly detecting changes in the objective and simply update the filter by starting the optimization process in every N_{S}th frame. The parameter N_{S} determines how often the filter is updated, where $N_{\mathrm{S}} = 1$ corresponds to optimizing the filter in every frame, as in standard DCF methods. In every N_{S}th frame, we perform a fixed number of N_{CG} Conjugate Gradient iterations to refine the model. As a result, the average number of CG iterations per frame is reduced to $N_{\mathrm{CG}}/N_{\mathrm{S}}$, which has a substantial effect on the overall computational complexity of the learning. Note that N_{S} does not affect the updating of the sample space model, introduced in section 3.2, which is updated every frame.

To our initial surprise, we observed that a moderately infrequent update of the model ($N_{\mathrm{S}} \approx 5$) generally led to improved tracking results. We mainly attribute this effect to reduced over-fitting to the recent training samples. By postponing the model update a few frames, the loss is updated by adding a new mini-batch to the training samples, instead of only a single one. This might contribute to stabilizing the learning, especially in scenarios where a new sample is affected by sudden changes, such as out-of-plane rotations, deformations, clutter, and occlusions (see figure 1).

	Conv-1	Conv-5	HOG	CN
Feature dimension, D	96	512	31	11
Filter dimension, C	16	64	10	3

Table 1. The settings of the proposed factorized convolution approach, as employed in our experiments. For each feature, we show the dimensionality D and the number of filters C.

While increasing N_{S} leads to reduced computations, it may also reduce the convergence speed of the optimization, resulting in a less discriminative model. A naive compensation by increasing the number of CG iterations N_{CG} would counteract the achieved computational gains. Instead, we aim to achieve a faster convergence by better adapting the CG algorithm to online tracking, where the loss changes dynamically. This is obtained by substituting the standard Fletcher-Reeves formula to the Polak-Ribière formula [32] for finding the momentum factor, since it has shown improved convergence rates for inexact and flexible preconditioning [15], which have similarities to our scenario.

4. Experiments

We validate our proposed formulation by performing comprehensive experiments on four benchmarks: VOT2016 [21], UAV123 [27], OTB-2015 [35], and TempleColor [24].

4.1. Implementation Details

Our tracker is implemented in Matlab.[6] We apply the same feature representation as C-COT, namely a combination of the first (Conv-1) and last (Conv-5) convolutional layer in the VGG-m network, along with HOG and ColorNames (CN). For the factorized convolution presented in section 3.1, we learn one coefficient matrix P for each feature type. The settings for each feature is summarized in table 1. The regularization parameter λ in (7) is set to $2 \cdot 10^{-7}$. The loss (7) is optimized in the first frame using 10 Gauss-Newton iterations and 20 CG iterations for the subproblems (9). In the first iteration $i = 0$, the filter $\hat{f}_0$ is initialized to zero. To preserve the deterministic property of the tracker, we initialize the coefficient matrix P_0 by PCA, though we found random initialization to be equally robust.

For the sample space model, presented in section 3.2, we set the learning rate to $\gamma = 0.01$. The number of components are set to $L = 50$, which represents an 8-fold reduction compared to the number of samples ($M = 400$) used in C-COT. We update the filter in every $N_{\mathrm{S}} = 6$ frame (section 3.3). We use the same number of $N_{\mathrm{CG}} = 5$ Conjugate Gradient iterations as in C-COT. Note that *all* parameters settings are kept fixed for all videos in a dataset.

[6] Hardware specifications are included in the supplementary material.

	Baseline C-COT (Sec. 2)	$\Longrightarrow$	Factorized Convolution (Sec. 3.1)	$\Longrightarrow$	Sample Space Model (Sec. 3.2)	$\Longrightarrow$	Model Update (Sec. 3.3)
EAO	0.331		0.335		0.351		**0.375**
FPS	0.3		1.1		2.6		6.0
Compl. change	-		$D \rightarrow C$		$M \rightarrow L$		$N_{\text{CG}} \rightarrow \frac{N_{\text{CG}}}{N_{\text{S}}}$
Compl. red.	-		$6\times$		$8\times$		$6\times$

Table 2. Analysis of our approach on the VOT2016. The impact of progressively integrating one contribution at the time, from left to right, is displayed. We show the performance in Expected Average Overlap (EAO) and speed in FPS (benchmarked on a single CPU). We also summarize the reduction in learning complexity $\mathcal{O}(N_{\text{CG}} D M \bar{K})$ obtained in each step, both symbolically and in absolute numbers (bottom row) using our settings. Our contributions systematically improve both performance and speed.

	DNT	Staple+	SRBT	EBT	DDC	Staple	MLDF	SSAT	TCNN	C-COT	**ECO**
EAO	0.278	0.286	0.290	0.291	0.293	0.295	0.311	0.321	0.325	*0.331*	**0.375**
Failure rate	1.18	1.32	1.25	0.90	1.23	1.35	*0.83*	1.04	0.96	0.85	**0.73**
Accuracy	0.50	*0.55*	0.50	0.44	0.53	0.54	0.48	**0.57**	0.54	0.52	0.53
EFO	1.127	**44.765**	3.688	3.011	0.198	*11.144*	1.483	0.475	1.049	0.507	4.530

Table 3. State-of-the-art in terms of expected average overlap (EAO), robustness (failure rate), accuracy, and speed (in EFO units) on the VOT2016 dataset. Only the top-10 best compared trackers are shown. Our approach achieves superior results in terms of performance and speed compared to top ranked methods in the challenge: C-COT, TCNN, and SSAT.

4.2. Baseline Comparison

Here, we analyze our approach on the VOT2016 benchmark by demonstrating the impact of progressively integrating our contributions. The VOT2016 dataset consists of 60 videos compiled from a set of more than 300 videos. The performance is evaluated both in terms of accuracy (average overlap during successful tracking) and robustness (failure rate). The overall performance is evaluated using Expected Average Overlap (EAO) which accounts for both accuracy and robustness. We refer to [22] for details.

Table 2 shows an analysis of our contributions. The integration of our factorized convolution into the baseline leads to a slight performance improvement with significant reduction in complexity ($6\times$). The sample space model further improves the performance by a relative gain of 4.8% in EAO, while reducing the learning complexity by a factor of 8. Additionally incorporating our proposed model update elevates us to an EAO score of 0.375, leading to a final relative gain of 13.3% compared to the baseline. In table 2 we also show the impact on the tracker speed achieved by our contributions. For a fair comparison, we report the FPS measured on a single CPU for all entries in the table, without accounting for feature extraction time. Each of our contributions systematically improves the speed of the tracker, combining to a 20-fold final gain compared to the baseline. When including all steps (also feature extraction), the GPU version of our tracker operates at 8 FPS.

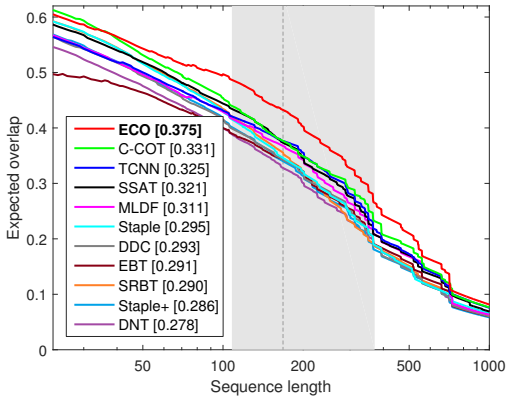


Figure 4. Expected Average Overlap (EAO) curve on VOT2016. We only display the top 10 ranked trackers in the challenge for clarity. The EAO measure for each tracker is displayed in the legend. It represents the average EAO over typical sequence lengths (grey region) in the dataset (see [22]). Our approach achieves outstanding performance with a relative gain of 13.3% in the EAO measure compared to the top performer in the challenge.

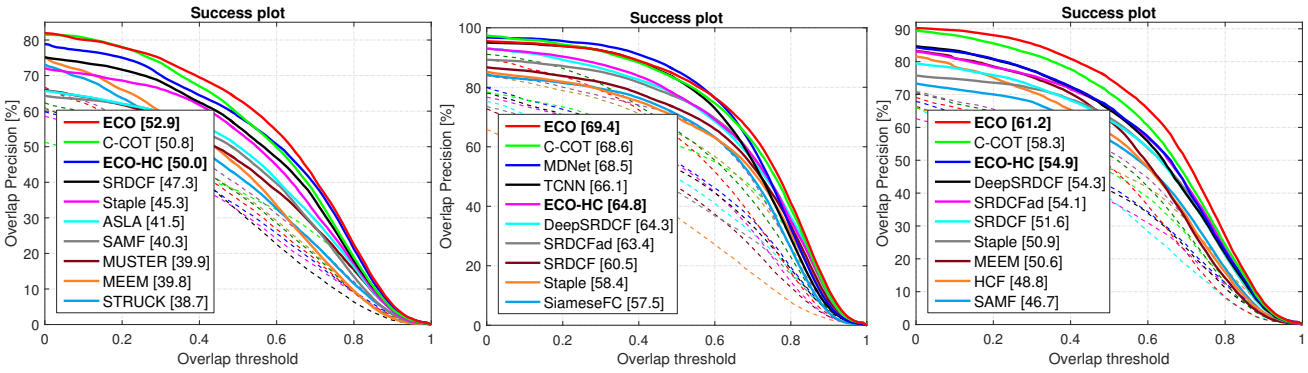
We found the settings in table 1 to be insensitive to minor changes. Substantial gain in speed can be obtained by reducing the number of filters C further, at the cost of slight reduction in performance. Moreover, we observed that our sample model provides consistently better results compared to the training sample set management employed in C-COT when using the same number of components and samples ($L = M$). This is particularly evident for a smaller number of components/samples $L, M \leq 100$: When reducing the number of samples from $M = 400$ to $M = 50$ in the standard approach, the EAO decreases from 0.335 to 0.326 (-2.7%). Instead, when using our approach with $L = 50$ components, the EAO increases by $+4.8\%$ to 0.351. In case of the model update, we observed an upward trend in performance when increasing N_{S} from 1 to 6. When increasing N_{S} further, a gradual downward trend was observed. We therefore use $N_{\text{S}} = 6$ throughout our experiments.

4.3. State-of-the-art Comparison

Here, we compare our approach with state-of-the-art trackers on four challenging tracking benchmarks.[7]

VOT2016 Dataset: In table 3 we compare our approach, in terms of expected average overlap (EAO), robustness, accuracy and speed (in EFO units), with the top-10 ranked trackers in the VOT2016 challenge. The top-ranked performer in VOT2016 challenge, C-COT, provides an EAO score of 0.331%. Our approach achieves a relative gain of 13.3% in EAO compared to C-COT. Among the compared trackers, MLDF obtains substantial robustness with a failure rate of 0.83. Our approach significantly improves the robustness with a failure rate of 0.73. Further, our tracker

[7] Detailed results are provided in the supplementary material.



(a) UAV123 (b) OTB-2015 (c) Temple-Color

Figure 5. Success plots on the UAV-123 (a), OTB-2015 (b) and TempleColor (c) datasets. Only the top 10 trackers are shown in the legend for clarity. The AUC score of each tracker is shown in the legend. Our approach significantly improves the state-of-the-art on all datasets.

achieves competitive accuracy. We also report the total speed in terms of EFO, which normalizes the speed with respect to hardware performance. Note that EFO also takes feature extraction time into account, a major additive complexity that is independent of our DCF improvements. In our comparison, Staple [1] and its variant Staple+ achieve the best speed. Among the top three trackers in the challenge, TCNN [28] obtains the best speed, with an EFO of 1.049. Our approach achieves an almost 5-fold speedup in EFO and a relative performance improvement of 14.2% in EAO compared to TCNN. Figure 4 shows the EAO curves.

UAV123 Dataset: Aerial tracking using unmanned aerial vehicle (UAVs) has received much attention recently with many vision applications, including wild-life monitoring, search and rescue, navigation, and crowd surveillance. In these applications, persistent UAV navigation is required, for which real-time tracking output is crucial. In such cases, the desired tracker should be accurate and robust, while operating in real-time under limited hardware capabilities, *e.g.*, CPUs or mobile GPU platforms. We therefore introduce a real-time variant of our method (ECO-HC), based on hand-crafted features (Color Names and HOG), operating at 60 FPS on a single i7 CPU (including feature extraction).

We evaluate our trackers on the recently introduced aerial video benchmark, UAV123 [27], for low altitude UAV target tracking. The dataset consists of 123 aerial videos with more than 110K frames. The trackers are evaluated using success plot [34], calculated as percentage of frames with an intersection-over-union (IOU) overlap exceeding a threshold. Trackers are ranked using the area-under-the-curve (AUC) score. Figure 5a shows the success plot over all the 123 videos in the dataset. We compare with all tracking results reported in [27] and further add Staple [1], due to its high frame-rate, and C-COT [9]. Among the top 5 compared trackers, only Staple runs at real-time with an AUC score of 45.3%. Our ECO-HC tracker also operates in real-time (60 FPS) with an AUC score of 50.0%, significantly outperforming Staple by 4.7%. C-COT obtains an AUC score of 50.8%. Our ECO outperforms C-COT, achieving an AUC score of 52.9%, using same features.

OTB2015 Dataset: We compare our tracker with 20 state-of-the-art methods: TLD [20], Struck [16], CFLB [13], ACT [10], TGPR [14], KCF [18], DSST [5], SAMF [23], MEEM [36], DAT [31], LCT [26], HCF [25], SRDCF [7], SRDCFad [8], DeepSRDCF [6], Staple [1], MDNet [29], SiameseFC [2], TCNN [28] and C-COT [9].

Figure 5b shows the success plot over all the 100 videos in the OTB-2015 dataset [35]. Among the compared trackers using hand-crafted features, SRDCFad provides the best results with an AUC score of 63.4%. Our proposed method, ECO-HC, also employing hand-crafted features outperforms SRDCFad with an AUC score of 64.8%, while running on a CPU with a speed of 60 FPS. Among the compared deep feature trackers, C-COT, MDNet and TCNN provide the best results with AUC scores of 68.6%, 68.5% and 66.1% respectively. Our approach ECO, provides the best performance with an AUC score of 69.4%.

TempleColor Dataset: In figure 5c we present results on the TempleColor dataset [24] containing 128 videos. Our method again achieves a substantial improvement over C-COT, with a gain of 2.9% in AUC.

5. Conclusions

We revisit the core DCF formulation to counter the issues of over-fitting and computational complexity. We introduce a factorized convolution operator to reduce the number of parameters in the model. We also propose a compact generative model of the training sample distribution to drastically reduce memory and time complexity of the learning, while enhancing sample diversity. Lastly, we suggest a simple yet effective model update strategy that reduces over-fitting to recent samples. Experiments on four datasets demonstrate state-of-the-art performance with improved frame rate.

References

[1] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016. 8

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

[3] A. Bibi, M. Mueller, and B. Ghanem. Target response adaptation for correlation filter tracking. In *ECCV*, 2016. 1

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 1, 2, 6

[5] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 1, 8

[6] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshop*, 2015. 1, 8

[7] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. 1, 3, 5, 6, 8

[8] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *CVPR*, 2016. 1, 8

[9] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 1, 2, 3, 5, 8, 10, 11

[10] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, 2014. 1, 2, 4, 8

[11] A. Declercq and J. H. Piater. Online learning of gaussian mixture models - a two-level approach. In *3rd International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 605–611, 2008. 5

[12] H. K. Galoogahi, T. Sim, and S. Lucey. Multi-channel correlation filters. In *ICCV*, 2013. 1, 10

[13] H. K. Galoogahi, T. Sim, and S. Lucey. Correlation filters with limited boundaries. In *CVPR*, 2015. 1, 8

[14] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian process regression. In *ECCV*, 2014. 8

[15] G. H. Golub and Q. Ye. Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM J. Scientific Computing*, 21(4):1305–1320, 1999. 6

[16] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 8

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012. 2

[18] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015. 1, 6, 8

[19] J. Hyeong Hong and A. Fitzgibbon. Secrets of matrix factorization: Approximations, numerics, manifold optimization and random restarts. In *ICCV*, 2015. 4

[20] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, 2010. 8

[21] M. Kristan, A. Leonardis, J. Matas, R. Felsberg, Pflugfelder, M., L. Čehovin, G. Vojír, T.and Hger, and et al.˙ The visual object tracking vot2016 challenge results. In *ECCV workshop*, 2016. 1, 2, 6, 11

[22] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojír, G. Nebehay, R. Pflugfelder, and G. Hger. The visual object tracking vot2015 challenge results. In *ICCV workshop*, 2015. 7

[23] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV Workshop*, 2014. 8

[24] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: Algorithms and benchmark. *TIP*, 24(12):5630–5644, 2015. 2, 6, 8

[25] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 1, 8

[26] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, 2015. 1, 8

[27] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016. 6, 8

[28] H. Nam, M. Baek, and B. Han. Modeling and propagating cnns in a tree structure for visual tracking. *CoRR*, abs/1608.07242, 2016. 8

[29] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 8

[30] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006. 3, 4, 10

[31] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *CVPR*, 2015. 8

[32] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994. 6, 10

[33] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. 11

[34] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 8

[35] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 1, 2, 6, 8, 11, 12

[36] J. Zhang, S. Ma, and S. Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014. 8

Supplementary Material

This supplementary material contains additional details and derivations related to the our approach presented in section 3. It also includes hardware specifications and additional experimental results on the VOT2016 and OTB-2015 datasets.

Complexity Analysis of the Learning

Here, we derive the computational complexity of the learning step in the baseline C-COT [9]. The learning itself is completely dominated by the problem of solving the normal equations (5). This linear system is iteratively solved using the Conjugate Gradient (CG) method [30, 32]. The dominating computation in CG is the evaluation of the left-hand side of (5), which is performed once per CG iteration. This computation is performed as

$$A^{\mathrm{H}}(\Gamma(A\hat{\mathbf{f}})) + W^{\mathrm{H}}(W\hat{\mathbf{f}}), \tag{13}$$

where the parentheses are used to indicate the order in which the operations are performed. Since the conjugate symmetry in the filter $\hat{\mathbf{f}}$ is preserved by the operations in (13), only half of the spectrum needs to be processed. We can therefore regard $\hat{\mathbf{f}}$ as a complex vector of $\sum_d K_d = D\bar{K}$ elements, where K_d is the bandwidth of channel d in the filter (see section 2), $\bar{K} = \frac{1}{D}\sum_d K_d$ is the average number of Fourier coefficients per channel and D is the number of feature channels d.

The matrix A contains a diagonal block $A_{j,d}$ of size $K \times K_d$ for each sample $j \in \{1, \dots, M\}$ and channel $d \in \{1, \dots, D\}$. Here, we have defined $K = \max_d K_d$. The diagonal of $A_{j,d}$ consists of the elements $\{X_j^d[k]\hat{b}_d[k]\}_{k=0}^{K_d}$. As previously shown for the discrete DCF case [12], the matrix A can be permuted to a block diagonal matrix $\tilde{A} = \oplus_{k=0}^{K} \tilde{A}_k$, where $\tilde{A}_k$ contains the elements $(\tilde{A}_k)_{j,d} = X_j^d[k]\hat{b}_d[k]$. The operations $\hat{\mathbf{f}} \mapsto A\hat{\mathbf{f}}$ and $\hat{\mathbf{v}} \mapsto A^{\mathrm{H}}\hat{\mathbf{v}}$ can thus be implemented as block-wise dense matrix-vector multiplications, with a total of $\mathcal{O}(DM\bar{K})$ operations. Moreover, Γ is a diagonal matrix containing the weights α_j, giving $\mathcal{O}(M\bar{K})$ operations.

In the second term of (13), arising from the spatial regularization in the loss (3), W and W^{H} are convolution matrices with the kernel $\hat{w}[k]$. These operations have a complexity of $\mathcal{O}(D\bar{K}K_w)$, where K_w are the number of non-zero coefficients in $\hat{w}$ (*i.e.* the size of the kernel). In practice however, the kernel $\hat{w}[k]$ is small (typically 5×5), having a lesser impact on the overall complexity. To simplify the complexity expression, we therefore disregard this term. By taking the number of CG iterations N_{CG} into account, we thus obtain the final expression $\mathcal{O}(N_{\mathrm{CG}}DM\bar{K})$ for the complexity of the learning step.

The preprocessing steps needed for the CG optimization only have a marginal impact on the overall learning time. The most significant of these being the Fast Fourier Transform (FFT) of the feature map, having a $\mathcal{O}(\sum_d N_d \log N_d)$ complexity, where N_d is the resolution of feature channel d. But since the FFT computations correspond to roughly 1% of the total time in C-COT, we exclude this part.

Factorized Convolution Operator Optimization

Here, we provide more details regarding the optimization procedure for learning the factorized convolution operator (section 3.1). We consider the case of learning the factorized operator $S_{Pf}\{x\}$ in (6) based on a single sample (x, y),

$$E(f, P) = \|S_{f,P}\{x\} - y\|_{L^2}^2 + \sum_{c=1}^{C} \|wf^c\|_{L^2}^2 + \lambda\|P\|_F^2 \,. \tag{14}$$

The loss is obtained by employing the factorized operator $S_{f,P}\{x\}$ in the data term of the original loss (3) and adding a regularization on the Frobenius norm $\|P\|_F^2$ of P.

By applying the Parseval's formula to the first two terms of (14) and utilizing the linearity and convolution properties of the Fourier series coefficients, we obtain the equivalent loss (7), where we have defined the interpolated feature map as $z = J\{x\}$ to simplify notation. Note that the matrix-vector products in (7) are performed point-wise,

$$(\hat{z}^{\mathrm{T}}P\hat{f})[k] = \sum_{d=1}^{D}\sum_{c=1}^{C} \hat{z}^d[k]p_{d,c}\hat{f}^c[k] \,, \quad k \in \mathbb{Z} \,. \tag{15}$$

We use the Gauss-Newton method [30] to optimize the non-linear least squares problem (7). In each iteration i, the residual in the data-term is linearized by performing a first order Taylor expansion (8) at the current estimate $(\hat{f}_i, P_i)$. This gives the following quadratic sub-problem (9). To derive a simple formula for the normal equations of (9), we first introduce some notation. Let $\hat{\mathbf{f}}$ be the vectorization of $\hat{f}_{i,\Delta}$, analogously to (5), and define $\mathbf{\Delta p} = \mathrm{vec}(\Delta P)$. Further, let $\mathbf{p}_c$ denote the cth column in P_i and set $\mathbf{p} = \mathrm{vec}(P_i)$. We then define the matrices,

$$A_P = \begin{bmatrix} \mathbf{0}_{K-K_1\times 2K_1+1} & \cdots & \mathbf{0}_{K-K_C\times 2K_C+1} \\ \mathrm{diag}\begin{pmatrix} \hat{z}[-K_1]^{\mathrm{T}}\mathbf{p}_1 \\ \vdots \\ \hat{z}[K_1]^{\mathrm{T}}\mathbf{p}_1 \end{pmatrix} & \cdots & \mathrm{diag}\begin{pmatrix} \hat{z}[-K_C]^{\mathrm{T}}\mathbf{p}_C \\ \vdots \\ \hat{z}[K_C]^{\mathrm{T}}\mathbf{p}_C \end{pmatrix} \\ \mathbf{0}_{K-K_1\times 2K_1+1} & \cdots & \mathbf{0}_{K-K_C\times 2K_C+1} \end{bmatrix}$$

$$B_f = \begin{pmatrix} (\hat{f}_i \otimes \hat{z})[-K]^{\mathrm{T}} \\ \vdots \\ (\hat{f}_i \otimes \hat{z})[K]^{\mathrm{T}} \end{pmatrix} . \tag{16}$$

Here, A_P has a structure very similar to the matrix A in (5), but contains only a single training sample. Note that

the diagonal blocks in A_P are padded with zero matrices $\mathbf{0}_{M\times N}$ along the columns to achieve the same number of $2K+1$ rows.

The Gauss-Newton sub-problem (9) can then be expressed as,

$$\tilde{E}(\hat{\mathbf{f}},\mathbf{\Delta p}) = \left\| A_P\hat{\mathbf{f}} + B_f\mathbf{\Delta p} - \hat{\mathbf{y}} \right\|_2^2 + \left\| W\hat{\mathbf{f}} \right\|_2^2 + \lambda\|\mathbf{p}+\mathbf{\Delta p}\|_2^2 . \tag{17}$$

Here, the convolution matrix W and the vectorization $\hat{\mathbf{y}}$ are defined as in (5). The normal equations of (17) are obtained by setting the gradient to zero,

$$\begin{bmatrix} A_P^{\mathrm{H}}A_P + W^{\mathrm{H}}W & A_P^{\mathrm{H}}B_f \\ B_f^{\mathrm{H}}A_P & B_f^{\mathrm{H}}B_f + \lambda I \end{bmatrix} \begin{bmatrix} \hat{\mathbf{f}} \\ \mathbf{\Delta p} \end{bmatrix} = \begin{bmatrix} A_P^{\mathrm{H}}\hat{\mathbf{y}} \\ B_f^{\mathrm{H}}\hat{\mathbf{y}} - \lambda\mathbf{p} \end{bmatrix}. \tag{18}$$

We employ the Conjugate Gradient method to iteratively solve the sub-problem (18).

Hardware Specifications

Our tracker is implemented in Matlab and uses Matconvnet [33] for deep feature extraction. The frame-rate measurements of our CPU implementation were performed on a desktop computer with a 4-core Intel Core i7-6700 CPU at 3.4 GHz. The frame-rate measurements of our GPU implementation were performed on a Tesla K40 GPU.

Additional Results on VOT2016

Here, we provide further experimental evaluation on the VOT2016 dataset [21] with 60 videos. The videos and the evaluation toolkit can be obtained from http://www.votchallenge.net/vot2016/.

In the VOT2016 dataset, each frame is labeled with five different attributes: camera motion, illumination change, occlusion, size change and motion change. Figure 6 visualizes the EAO of each attribute individually. Our approach achieves the best results on three attributes and improves over the baseline C-COT [9] on all five attributes.

Additional Results on OTB-2015

Here, we report additional results on the OTB-2015 dataset [35] with 100 videos. The ground truth annotations and videos are available at https://sites.google.com/site/benchmarkpami/.

In the OTB-2015 dataset, each video is annotated with 11 different attributes: scale variation, background clutter, out-of-plane rotation, in-plane rotation, illumination variation, motion blur, fast motion, deformation, occlusion, out of view and low resolution. The success plots of all attributes are shown in figure 7. Our ECO tracker achieves the best performance on 7 out of 11 attributes. Further, our method improves over the baseline C-COT [9] on 9 out of 11 attributes. For a fair comparison, we employ the same combination of deep and hand-crafted features in the baseline C-COT and as in our ECO tracker. Note that this set of features leads to an improved performance in C-COT compared to the original results reported in [9], where only deep convolutional features are used.

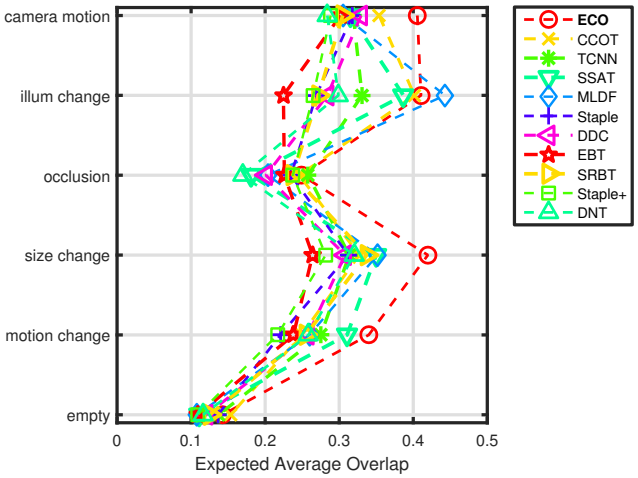


Figure 6. Expected Average Overlap (EAO) scores for each attribute on the VOT2016 dataset. Here, *empty* denotes frames with no labeled attribute.

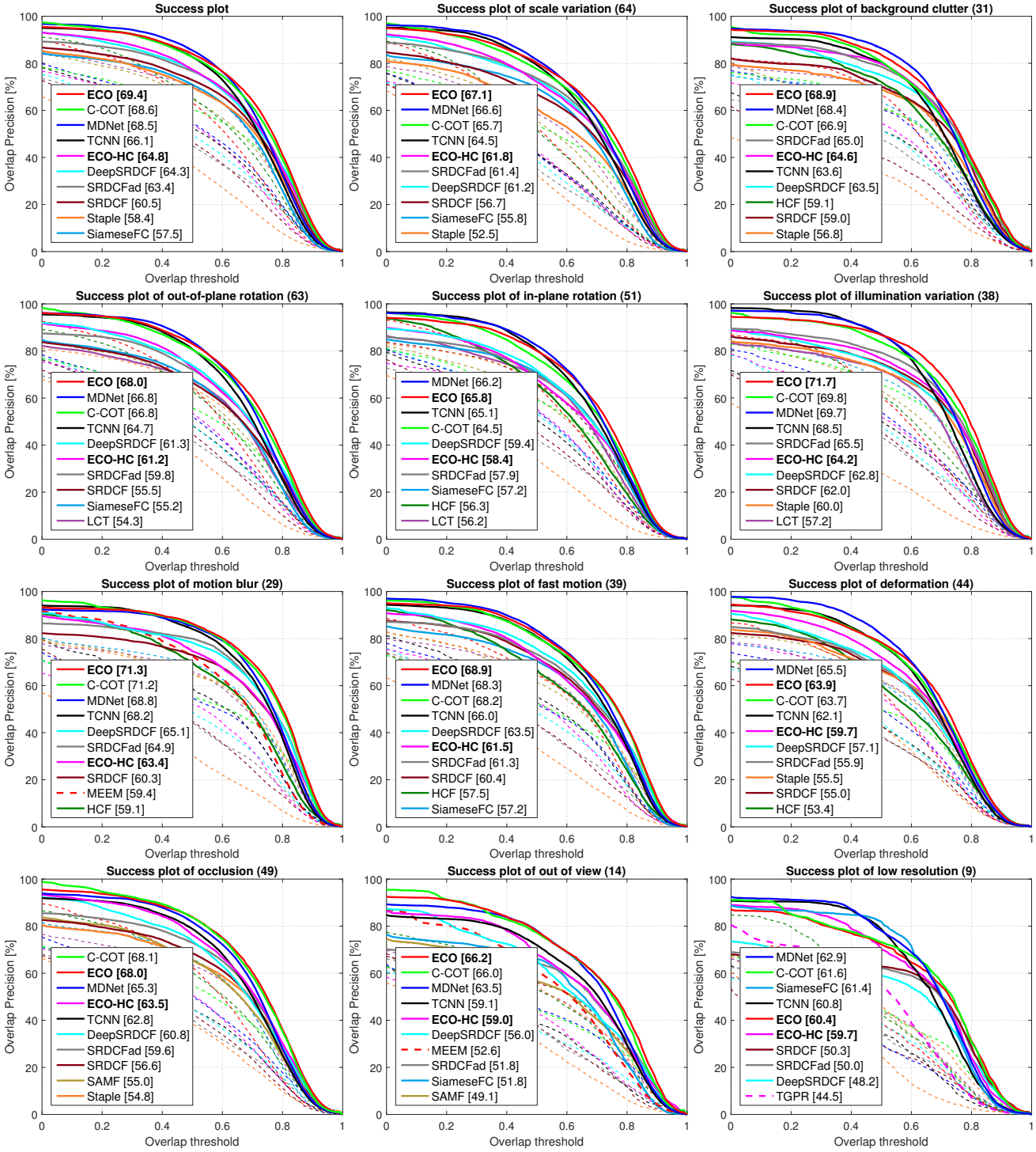


Figure 7. Success plots on the OTB-2015 dataset [35]. The total success plot (top-left) is displayed along with the plots for all 11 attributes. The title text indicate the name of the attribute and the number of videos associated with it. The area-under-the-curve scores for the top 10 trackers are shown in the legend.