On the Optimization of Advanced DCF-Trackers

Joakim Johnander[1,2], Goutam Bhat[1], Martin Danelljan[1], Fahad Shahbaz Khan[1,3], and Michael Felsberg[1]

[1] CVL, Department of Electrical Engineering, Linköping University, Sweden
[2] Zenuity, Sweden
[3] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE

Abstract. Trackers based on discriminative correlation filters (DCF) have recently seen widespread success and in this work we dive into their numerical core. DCF-based trackers interleave learning of the target detector and target state inference based on this detector. Whereas the original formulation includes a closed-form solution for the filter learning, recently introduced improvements to the framework no longer have known closed-form solutions. Instead a large-scale linear least squares problem must be solved each time the detector is updated. We analyze the procedure used to optimize the detector and let the popular scheme introduced with ECO serve as a baseline. The ECO implementation is revisited in detail and several mechanisms are provided with alternatives. With comprehensive experiments we show which configurations are superior in terms of tracking capabilities *and* optimization performance.

1 Introduction

Visual tracking is the computer vision problem of estimating a target trajectory in a video, given only its initial state. This is a challenge occurring in a wide range of vision problems seen for instance in autonomous cars, UAVs, and surveillance. For such applications, there is often a real-time constraint coupled with a desire to track objects undergoing challenging appearance changes. In recent years trackers based on Discriminative Correlation Filters (DCF) have shown promising tracking performance while often attaining good speed [16][17]. These trackers have repeatedly improved the state-of-the-art, recently due to the use of powerful features [9, 20] and more sophisticated models [8, 7, 12, 19].

The aim in DCFs is to learn a filter that, when applied to an input sample, produces a sharp and distinct peak at the target location. This is formulated as an objective over the filter coefficients. Via application of Parseval's formula, this problem is transformed into a linear least squares loss over the Fourier coefficients. In the general case, there is no solution in closed form, and the loss is minimized with an iterative solver. A standard method for such situations is the method of conjugate gradients (CG). A major advantage is that CG is able to handle the large size of the problem. Each iteration is linear in the number of parameters and it is known to converge to approximate solutions in a small number of iterations.

The effectiveness of the DCF-framework coupled with CG is well-supported empirically. In the most recent visual object tracking challenge [17], four of the top five trackers employed CG within the DCF-framework. Recent approaches investigate: more powerful features and how they can be learnt; changes to the model inference such as how scales are best handled; and additional components or alterations to the loss. In this work we instead shift our gaze toward the optimization procedure itself. The visual tracking scenario contains repeated filter optimizations of a loss that changes slowly over time, a very particular situation unlike what is usually studied in optimization literature. This leads not only to considerations regarding the objective, but also how previous optimizations are best exploited to warm-start future optimizations. As the target may undergo appearance changes, an additional concern is overfitting. A better optimization method may yield improved performance in some cases, but may lead to severe overfitting in others.

Our contributions: In this work we present a thorough description of the implementation and optimization procedure employed in many state-of-the-art visual tracking methods. Several new variants of this method are presented, with different approaches to select the search-direction, step-length, and perform warm-starts. We present comprehensive experiments and provide an analysis of the different methods both from a tracking performance perspective, and from an optimization perspective.

2 Related Work

In visual tracking, DCF-based methods have shown promising performance across several benchmarks. In essence the DCF is a linear regressor, which is trained in a supervised fashion to predict the classification score of the target object. There are two unique characteristics of the DCF paradigm that are credited for its success and popularity. First, the DCF implicitly performs a dense sampling of training patches by modeling detection as a convolution operation. This is particularly important for tracking, where labeled training data is scarce. Secondly, the convolution operation is approximated by a circular convolution, which enables efficient training and detection to be performed in the Fourier domain and exploitation of the $\mathcal{O}(n \log n)$ FFT algorithm. Furthermore, the negative effects of the circular (i.e. periodic) assumption can be effectively mitigated using windowing [2] and spatial regularization [7].

Since the DCF loss is convex, the minimizer can be expressed in closed form as the solution of the set of linear normal equations. There are however two special cases which admit particularly simple closed form expression. These only require element-wise operations, enabling simple and efficient $\mathcal{O}(n)$ implementations. The two cases are: (1) multiple training samples with a single feature dimension [2], and (2) a single training sample with multiple feature dimensions [6]. In practice, however, it is essential to use both multiple samples and multi-dimensional features to obtain a discriminative appearance model. As no closed form solution is known for the general case, early works update the filter sequen-

tially using simple update rules (KCF [15], ACT [10], DSST [6]). However, these update schemes rely on harsh assumptions leading to suboptimal filters with significantly reduced discriminability. Further, these methods cannot address the periodic artifacts using spatial regularization [7] or constraints [12].

Several recent works employ iterative optimization strategies in order to minimize the full DCF objective in an online manner for tracking [7, 8, 12]. These methods enjoy two key advantages. First, they benefit from asymptotic convergence to the optimal filter, leading to a more discriminative model. Second, alternate regularization approaches and filter constraints can be integrated, which is important for addressing the periodic artifacts. While previous optimization based approaches [7, 8, 12] suffered from significantly increased computational complexity, recent work [5, 19] have demonstrated that state-of-the-art results can be obtained at impressive frame rates.

3 DCF-Formulation Revisited

The DCF-paradigm solves the visual tracking task by constructing a correlation or convolution filter that, when applied to an image, discriminates the target from the background. The filter is learnt in the first frame where a label of the target location is available, and is usually updated in subsequent frames treating earlier frames with corresponding predictions as training data. For the training, the formulation of a suitable loss is essential. In this section we formulate the loss employed by several state-of-the-art trackers in the filter construction [8, 5, 14, 13]. We proceed to discuss appropriate methods of minimizing this loss.

3.1 Formulation of the Loss

The aim is to find the filter f that is the best fit to a set of sample-label pairs $\{(x^1, y^1), (x^2, y^2), \dots, (x^C, y^C)\}$. Each sample $x^c \in \mathbb{R}^{D \times T_1 \times T_2}$ is a multidimensional feature map, with D feature channels and spatial extent $T_1 \times T_2$. The corresponding label $y^c \in \mathbb{R}^{T_1 \times T_2}$ attains a high value at the target location and low values otherwise. This is achieved by a Gaussian function with low variance. The DCF-framework finds the filter $f \in \mathbb{R}^{D \times T_1 \times T_2}$ by minimizing the loss

$$\epsilon(f) = \sum_{c=1}^{C} \mu^c \left\| \sum_{d=1}^{D} x_d^c * f_d - y^c \right\|_2^2 + \lambda \sum_{d=1}^{D} \|f_d\|_2^2 \ , \tag{1}$$

where each sample is weighted by μ^c. The convolution $(*)$ is applied per dimension, where x_d^c and f_d denote the d'th feature channel of the sample and filter respectively. The second term of (1) regularizes the filter weights with some parameter λ. In DCFs, convolution is a key concept as it performs a dense sampling of negative training data while being efficient to calculate. The Fourier basis diagonalizes the convolution operation into multiplication, while the transform itself is quick to calculate with the Fast Fourier Transform (FFT). There is a caveat, however, in that the convolution is cyclic, introducing boundary effects

which hampers performance and may allow the filter to learn the background, rather than the target appearance. Two ways have been proposed to deal with this. Galoogahi et al. propose to constrain filter coefficients far away from the center to zero [12]. They solve the constrained minimization problem with the Alternating Direction Method of Multipliers (ADMM) [3]. A disadvantage of this approach is that it requires repeated transitions between the spatial and Fourier domain, adding a substantial computational cost. An alternative proposed by Danelljan et al. [7] is to replace the filter weight regularization with a term that depends on the filter coefficients' distance to the center

$$\epsilon(f) = \sum_{c=1}^{C} \mu^c \left\| \sum_{d=1}^{D} x_d^c * f_d - y^c \right\|_2^2 + \sum_{d=1}^{D} \|w f_d\|_2^2 \ . \tag{2}$$

The second term applies a cost $w_{i,j}$ to each filter coefficient, depending on its position. The regularization function w typically attains small values in the filter center, and increases as the position moves towards the borders. Experimental results show that the spatial regularization indeed leads to a filter that further emphasizes the target while obtaining improved tracking performance [7, 19].

Previously, the samples x^c comprised handcrafted features such as Histogram of Oriented Gradients (HOG) or Color Names (CN). However, with the advent of deep learning, there emerged a desire to employ features extracted from deep convolutional neural networks (CNN). The features extracted from the deeper layers of a CNN exhibit robustness to severe appearance changes which would otherwise lead to tracking failure. They are however of low resolution, and in order to obtain accurate detections we have to rely on low level information when it is available. We would therefore like to utilize shallow features, such as HOG, CN, or features extracted from the earlier layers of a CNN. In order to combine features of different resolutions, Danelljan et al. [8] proposed to view the components of (2) as continuous functions. That is, the filter and the labels f_d, y^c are functions of two variables (t_1, t_2). The extracted feature maps x^c are interpolated to the continuous domain with some interpolation kernel b. The continuous interpretation is possible as the DCF-formulation works with a Fourier basis. A continuous function can be written as its Fourier series which may be truncated for a finite representation. The loss is transformed into the Fourier domain via Parseval's formula,

$$\epsilon(f) = \sum_{c=1}^{C} \mu^c \left\| \sum_{d=1}^{D} \mathrm{DFT}\{x_d^c\} \hat{b} \hat{f}_d - \hat{y}^c \right\|_2^2 + \sum_{d=1}^{D} \|\hat{w} * \hat{f}_d\|_2^2, \tag{3}$$

where $\hat{\cdot}$ denotes the Fourier coefficients and DFT the Discrete Fourier Transform.

3.2 DCF-Loss Vectorization

In order to solve (3) a finite representation of the included terms is required. The continuous functions are represented with infinite sequences of Fourier coefficients, which may be truncated. We use the first K coefficients from each

feature channel, a total of $N^2 = (2K+1)^2$ coefficients per channel. We note that including the same number of coefficients in each dimension is inefficient, as fewer coefficients are required with decreasing feature channel resolution. This will however not be a problem as we may constrain those coefficients to zero in the formulation and employ an optimization scheme which exploits sparsity. We will now rewrite the loss into matrix-vector-form in order to apply standard optimization techniques. The vectorized filter is rewritten as

$$\hat{\mathbf{f}} = \begin{bmatrix} \hat{\mathbf{f}}_1 \\ \hat{\mathbf{f}}_2 \\ \vdots \\ \hat{\mathbf{f}}_D \end{bmatrix}, \text{ where } \hat{\mathbf{f}}_d = \begin{bmatrix} \hat{f}_d[-K,-K] \\ \vdots \\ \hat{f}_d[-K,K] \\ \vdots \\ \hat{f}_d[K,K] \end{bmatrix}, \tag{4}$$

which is a DN^2 sized vector. The feature map components are contained in a $CN^2 \times DN^2$ sized matrix

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,D} \\ A_{2,1} & A_{2,2} & \dots & A_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ A_{C,1} & A_{C,2} & \dots & A_{C,D} \end{bmatrix}, \tag{5}$$

where

$$A_{c,d} = \operatorname{diag} \begin{bmatrix} \mathrm{DFT}\{x_d^c\}[-K,-K] \cdot \hat{b}[-K,-K] \\ \vdots \\ \mathrm{DFT}\{x_d^c\}[-K,K] \cdot \hat{b}[-K,K] \\ \vdots \\ \mathrm{DFT}\{x_d^c\}[K,K] \cdot \hat{b}[K,K] \end{bmatrix}. \tag{6}$$

Here, diag is the transformation from a vector to a corresponding diagonal matrix. The labels are stored in a size CN^2 vector

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{\mathbf{y}}^1 \\ \hat{\mathbf{y}}^2 \\ \vdots \\ \hat{\mathbf{y}}^C \end{bmatrix}, \text{ where } \hat{\mathbf{y}}^c = \begin{bmatrix} \hat{y}^c[-K,-K] \\ \vdots \\ \hat{y}^c[-K,K] \\ \vdots \\ \hat{y}^c[K,K] \end{bmatrix}. \tag{7}$$

The sample weights are stored in the diagonal matrix

$$\mu = \begin{bmatrix} \mu^1 I_{N^2} & & & \\ & \mu^2 I_{N^2} & & \\ & & \ddots & \\ & & & \mu^C I_{N^2} \end{bmatrix}, \tag{8}$$

where I_{N^2} is the identity matrix of size $N^2 \times N^2$. Lastly, the spatial regularization is rewritten as

$$\begin{bmatrix} \hat{w} * \hat{f}_1 \\ \vdots \\ \hat{w} * \hat{f}_D \end{bmatrix} = W\hat{\mathbf{f}} \ , \tag{9}$$

where W is the block-diagonal matrix where each block is a convolution matrix containing the elements of $\hat{w}$. With these definitions, the objective (3) is expressed as

$$\begin{aligned} \epsilon(f) &= \left\| \sqrt{\Gamma} A\hat{\mathbf{f}} - \hat{\mathbf{y}} \right\|_2^2 + \|W\hat{\mathbf{f}}\|_2^2 = \\ &= (A\hat{\mathbf{f}} - \hat{\mathbf{y}})^H \Gamma (A\hat{\mathbf{f}} - \hat{\mathbf{y}}) + \hat{\mathbf{f}}^H W^H W \hat{\mathbf{f}} = \\ &= \hat{\mathbf{f}}^H (A^H \Gamma A + W^H W)\hat{\mathbf{f}} - 2\hat{\mathbf{y}}^H \Gamma A \hat{\mathbf{f}} + \hat{\mathbf{y}}^H \Gamma \hat{\mathbf{y}} \ , \end{aligned} \tag{10}$$

where $\cdot^H$ denotes conjugate transpose. The difference between (3) and (10) is the truncation of the Fourier coefficients, and it is assumed that the difference is small. In order to minimize the loss, we note that the problem is unconstrained and convex. We can therefore set the derivative to zero and solve the arising normal equations. The derivative is found as

$$\frac{\partial \epsilon(f)}{\partial \hat{\mathbf{f}}} = 2\hat{\mathbf{f}}^H (A^H \Gamma A + W^H W) - 2\hat{\mathbf{y}}^H \Gamma A \ , \tag{11}$$

and setting the derivative to zero leads to

$$(A^H \Gamma A + W^H W)\hat{\mathbf{f}} = A^H \Gamma \hat{\mathbf{y}} \ . \tag{12}$$

It is important to note here that the left-hand side contains two terms with exploitable properties. Namely, (i) the matrix-vector product $A^H \Gamma A \hat{\mathbf{f}}$ is very sparse if it is calculated in the order $A^H(\Gamma(A\hat{\mathbf{f}}))$; and (ii) the product $W\hat{\mathbf{f}}$ corresponds to a convolution that, if the kernel is small, is efficient to calculate.

3.3 The Conjugate Gradient Method

The size of the equation system (12) depends on the number of training instances C and the feature dimensionality D. If the high-dimensional deep features are employed, D can be very large. Furthermore, C should be sufficiently large in order for the model to generalize the target appearance. The number of equations and the number of variables may each be in the order 10^5, and we would therefore like to employ a first order optimization method that can exploit sparsity. Danelljan et al. [8] proposed to employ the method of Conjugate Gradients (CG),

which fulfills these priorities. It is a first order line-search method, which applies a Gram-Schmidt procedure to the steepest descent direction. This makes sure that the step directions are orthogonal with respect to a special inner product. For this method, linear convergence has been proven and typically an acceptable solution is reached in very few steps. CG furthermore does not need to store or form any additional matrices like second order methods would. We will briefly describe the method in order to later describe design choices. Details and proofs regarding the convergence of CG are available in [22].

The idea is to apply CG to the normal equations (12). To simplify notation, we apply the notation found in [22, 21]. Consider a system of n equations and n variables

$$A\mathbf{x} = \mathbf{b} \ . \tag{13}$$

It should be noted that CG enjoys nice convergence properties if and only if the left-hand side is positive definite. This is the case for normal equations such as (12). The CG method solves the system by iteratively performing the update

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \ , \tag{14}$$

that is, taking a step along direction $\mathbf{p}_k$ of some step-length α_k. The step directions are found as

$$\mathbf{p}_k = \mathbf{r}_k - \sum_{i<k} \frac{\mathbf{p}_i^H A \mathbf{r}_k}{\mathbf{p}_i^H A \mathbf{p}_i} \mathbf{p}_i \ , \tag{15}$$

where $\mathbf{p}_i$ are previous search directions, and $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$ is the residual in the k'th iteration. This step is the component of the negative gradient that is orthogonal to all earlier steps, with respect to the inner product $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T A \mathbf{v}$. This is called *conjugacy* and lets CG avoid the inefficient zig-zag pattern that often emerges with steepest descent. The step lengths α_k are found as

$$\alpha_k = \frac{\mathbf{p}_k^H \mathbf{b}}{\mathbf{p}_k^H A \mathbf{p}_k} \ . \tag{16}$$

The step directions $\mathbf{p}$ form a basis and it is possible to show that our choice of α leads to the minimizer in n steps. The convergence is typically much faster however, and depends directly on the distribution of the eigenvalues of A. It is therefore common to apply a preconditioner M to the system of equations, which clusters the eigenvalues. In this case, we instead solve the system

$$E^{-1} A (E^{-1})^H \mathbf{x}' = E^{-1} \mathbf{b} \ , \tag{17}$$

where $\mathbf{x}' = E^T \mathbf{x}$ and $EE^T = M$. The solution will remain the same, but the convergence is faster if $E^{-1}A(E^{-1})^T$ has a more favorable distribution of eigenvalues than A. A common choice for the preconditioner that is efficient to calculate is the diagonal preconditioner M that keeps the diagonal elements of A, but is

zero everywhere else. It is not obvious what preconditioner should be used, as a more sophisticated preconditioner may yield faster convergence but will instead be more expensive to calculate.

As storing all the search directions $\mathbf{p}_k$ would lead to quadratic memory consumption, an equivalent recursive implementation is utilized instead [22]. The step length and step direction are found as

$$\alpha_k = \frac{\mathbf{r}_k^H \mathbf{z}_k}{\mathbf{p}_k^H A \mathbf{p}_k}, \quad \mathbf{p}_k = \mathbf{z}_k + \beta_{k-1} \mathbf{p}_{k-1} \ , \tag{18}$$

with

$$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1} A \mathbf{p}_{k-1} \ , \tag{19a}$$

$$\mathbf{z}_k = M^{-1} \mathbf{r}_k \ , \tag{19b}$$

$$\beta_{k-1} = \frac{\mathbf{z}_k^H \mathbf{r}_k}{\mathbf{z}_{k-1}^H \mathbf{r}_{k-1}} \ . \tag{19c}$$

The initial values of $\mathbf{r}$, $\mathbf{z}$, and $\mathbf{p}$ are

$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0 \ , \tag{20a}$$

$$\mathbf{z}_0 = M^{-1} \mathbf{r}_0 \ , \tag{20b}$$

$$\mathbf{p}_0 = \mathbf{z}_0 \ . \tag{20c}$$

Next we look at how CG is applied to the normal equations in the DCF-framework.

3.4 Applying CG to the DCF-problem

The conjugate gradient formulation is efficient as it only performs vector-vector products, except for two cases. In the updates of α in (18) and $\mathbf{r}$ in (19a), the matrix-vector products needs to be computed. As previously mentioned, the sample matrix A is sparse, and the application of W can be done via a small-kernel convolution as long as it is possible to represent the spatial regularization function w with few Fourier coefficients.

The matrix-vector product performed by the CG method when applied to the normal equations (12) is

$$\mathbf{q} = (A^H \Gamma A + W^H W)\mathbf{p} \ . \tag{21}$$

From (4) and (5) we obtain [8]

$$A^H \Gamma A \mathbf{p} = A^H \begin{bmatrix} \mu^1 \left(\sum_{d=1}^{D} A_{1,d} \mathbf{p} \right) \\ \vdots \\ \mu^C \left(\sum_{d=1}^{D} A_{C,d} \mathbf{p} \right) \end{bmatrix} = \begin{bmatrix} \left(\sum_{c=1}^{C} \bar{A}_{c,1} \mu^c \left(\sum_{d=1}^{D} A_{c,d} \mathbf{p} \right) \right) \\ \vdots \\ \left(\sum_{c=1}^{C} \bar{A}_{c,D} \mu^c \left(\sum_{d=1}^{D} A_{c,d} \mathbf{p} \right) \right) \end{bmatrix} . \tag{22}$$

That is, the calculation relies only on matrix-vector products with the diagonal blocks $A_{c,d}$ and is implemented as a vector-vector product. We also avoid storing the entire matrix A. For the second part of the matrix-vector product, we note that there is no need for the spatial regularization function w to contain high frequences. A smooth function is sufficient, and we therefore select a regularizer that is well represented by a few low frequency components. We then calculate

$$W^H W\hat{\mathbf{f}} = \begin{bmatrix} \hat{w} * \hat{w} * \hat{f}_1 \\ \vdots \\ \hat{w} * \hat{w} * \hat{f}_D \end{bmatrix}, \tag{23}$$

which is efficient as the filter $\hat{w}$ contains few coefficients. The conjugate transpose of the leftmost W was discarded here, as the functions w are usually symmetric and real. Hence, the conjugate gradient method can exploit the sparsity and performs only vector-vector products.

3.5 Subsequent Optimizations

In visual tracking we have a single labeled sample from which a model is learned to track an object in subsequent frames. For that case, we apply the CG-procedure and in a few iterations obtain an acceptable solution and information required for the warm-starts. In order to improve the model, samples extracted from subsequent frames and their labels are treated as additional training data. The tracker ECO for instance, updates its model every five frames. As the loss (2) does not change dramatically between subsequent optimizations, we warm-start CG. As an initial guess, the previous filter vector $\hat{\mathbf{f}}$ is used. An additional component of ECO is that the final search direction of the previous optimization is used to select the initial components. The variables $\mathbf{p}_{\text{old}}, \mathbf{z}_{\text{old}}, \mathbf{r}_{\text{old}}$ are used to modify the initialization (20) into

$$\mathbf{p}_0 = \mathbf{z}_0 + \beta_{-1}\mathbf{p}_{\text{old}} \ , \tag{24}$$

where the denominator of the formula for β is calculated as

$$\mathbf{z}_{k-1}^H \mathbf{r}_{k-1} = (1-\lambda)^{-\gamma}\mathbf{z}_{\text{old}}^H \mathbf{r}_{\text{old}} \ . \tag{25}$$

The learning rate λ is the same learning rate as that used to recursively weight samples, and γ is a hyperparameter that describes the decay of the previous search direction.

CG has remarkable convergence properties when minimizing a linear least squares loss and employing a constant preconditioner. The presented procedure introduces a somewhat different situation. It may be viewed as minimizing a loss that changes slightly over time. The preconditioners will not be constant, and the assumptions on which CG relies are violated. The literature proposes a way to deal with loss nonlinearities and non-constant preconditioners, which may be beneficial for our case. The strategy is to replace the β formula (19c), referred to as the *Fletcher-Reeves* formula, with the *Polak-Ribiere* formula

$$\beta_{k-1} = \frac{\mathbf{z}_k^H(\mathbf{r}_k - \mathbf{r}_{k-1})}{\mathbf{z}_{k-1}\mathbf{r}_{k-1}} \quad . \tag{26}$$

As the case presented in the literature differs somewhat from the DCF-framework case, it is not obvious whichever is superior. For completeness, we also include the *gradient descent* method. This is actually easily incorporated into the CG framework by selecting $\beta_k = 0$ as $\mathbf{z}_k$ is the negative gradient of the preconditioned loss. In this case, the way α_k is selected within CG is the optimal step length along the negative gradient. A fourth alternative is to employ the *Barzilai-Borwein* method [1], which has shown favorable performance in practice. We integrate it in the CG-framework by noting that the method takes steps in the direction of the negative gradient, where the step length is based on the change of the parameters and the gradients

$$\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_{k-1}, \quad \mathbf{y}_k = \mathbf{z}_k - \mathbf{z}_{k-1} \quad . \tag{27}$$

The step-length is selected by interleaving the updates

$$\alpha_k = \frac{\mathbf{s}_k^H \mathbf{s}_k}{\mathbf{s}_k^H \mathbf{y}_k} \quad , \quad \alpha_k = \frac{\mathbf{s}_k^H \mathbf{y}_k}{\mathbf{y}_k^H \mathbf{y}_k} \quad . \tag{28}$$

The motivation behind this choice is that it approximates the secant equation of the quasi-Newton methods. Next we describe our experiments, where we compare the four different updates and the impact of the conjugate direction warm-start.

4 Experiments

We analyze the optimization scheme and several of its components based on tracking performance and the behaviour of the loss . First we introduce the eight tracker configurations employed in our experiments including their respective rationale.

4.1 Optimizer Configurations

In order to minimize the loss resulting from the continuous, spatially regularized DCF-formulation, ECO relies on the conjugate gradient method with a special heuristic for warm-starting filter optimizations subsequent the initial one. We investigate the impact of this heuristic by comparing it with the two extremes: (i) warm-starting subsequent optimizations only with the previous solution and not with the previous search direction, that is, setting the forgetting rate $\gamma = \infty$; and (ii) fully keeping the initial step conjugate to the final search direction of the previous optimization, setting $\gamma = 0$. We further investigate the mechanism to ensure conjugacy: (i) Fletcher-Reeves formula; (ii) Polak-Ribiere formula; (iii) the removal of this mechanism, equivalent to gradient descent; and (iv) the replacement of this mechanism by another way of selecting the step-length, namely

Table 1: The strategies to calculate α and β are shown for four configurations.

	Fletcher-Reeves	**Polak-Ribiere**	**Gradient Descent**	**Barzilai-Borwein**
α	Eq. (18)	Eq. (18)	Eq. (18)	Eq. (28)
β	Eq. (19c)	Eq. (26)	0	0

the Barzilai-Borwein method. They are described in Table 1. The importance of the preconditioner is asserted via its removal. Finally, we try to hamper the baseline CG-method by always multiplying its calculated step-length with a factor 0.8. The intuition is that a lower loss not necessarily provides improved tracking capabilities, and such a heuristic could have a regularizing effect.

4.2 Evalulation Methodology

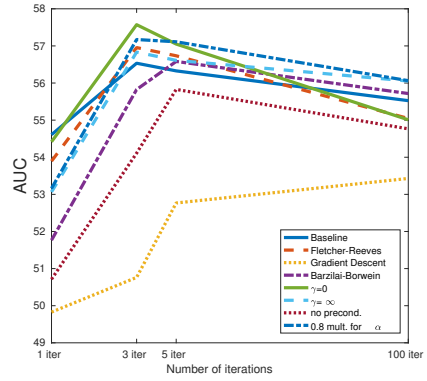
We run experiments on two tracking benchmarks. First we utilize the public VOT2018-benchmark, where performance is measured in terms of Accuracy and Robustness. Robustness measures the tracking failure rate and can be interpreted as the probability of succesful tracking for S frames, where S is a constant. Accuracy is the average overlap during successfully tracked frames. A more thorough description and analysis of the measures is given in [4]. There is typically a trade-off between accuracy and robustness, and one may be improved at cost of the other. Additionally we run experiments on a larger dataset formed by pooling the sequences of OTB-2015 [23], TempleColor [18], and NFS [11]. Overlapping sequences are removed, leaving a total of 286 sequences. Performance is measured in terms of area-under-the-curve (AUC) of the success plots.

We run each tracking algorithm at four different iteration configurations: (i) the baseline setting where the initial filter optimization runs for 150 iterations, and subsequent optimizations for 5 iterations; (ii) a fast setting where we run 90 and 3 iterations respectively; (iii) a setting with 30 initial iterations and only a single iteration in subsequent runs; (iv) and an overfitting setting with 150 iterations are used initially, and 100 iterations in subsequent runs.

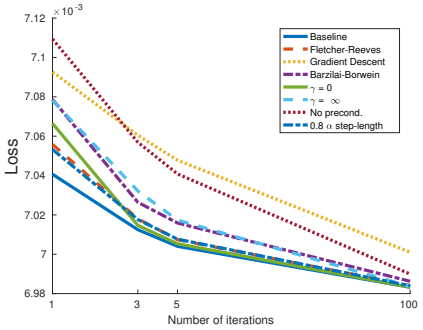
The trackers are analyzed based on their performance on the two benchmarks. We would furthermore like to gain some insight on how well the different methods minimize the loss and the relationship between this and tracking performance. Therefore we run an additional experiment on the pooled dataset where the loss is considered. The value of the loss after each optimization is stored as $L_{i,j}$ where i enumerates the N sequences and j the M_i frames in sequence i. A single performance number of the loss is obtained as

$$\bar{L} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{M_i} \sum_{j=1}^{M_i} L_{i,j} \quad . \tag{29}$$

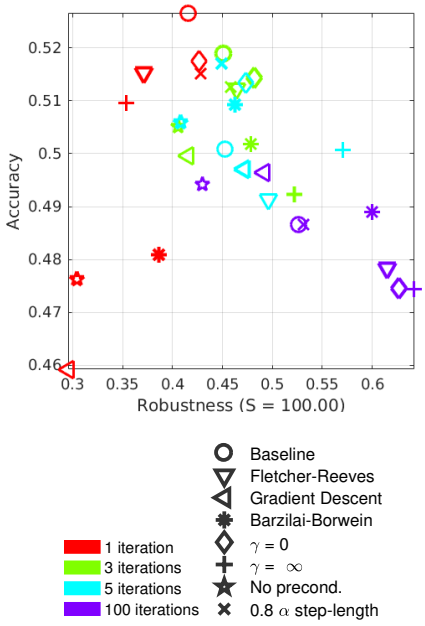
Our initial experiments revealed that the value of the loss is heavily correlated with the tracking performance, possibly due to contamination of the training set after a tracking failure. As our intention is to study the optimization performance, we provide all trackers with the same training set: the ground truth.



Tracking performance on pooled dataset



Mean average loss on pooled dataset



Tracking performance on VOT2018

Fig. 2: The results of the eight configurations are shown. The tracking performance (a) and the mean average loss (b) on the pooled dataset are shown as a function of the number of optimizer iterations. This dataset is the union of the OTB-2015, TempleColor, and NFS datasets. The performance on the public VOT2018 benchmark is shown in terms of accuracy and robustness (c).

4.3 Results

Figures 1a and 1c show the results on the VOT2018 benchmark and the pooled dataset, respectively. The average loss on the pooled dataset is shown in fig. 1b. We first consider the effect of the conjugacy warm-start. Removing the conjugacy warm-start ($\gamma = \infty$) provides reduced performance on VOT2018 in the single iteration setting, both in terms of accuracy and robustness. For 3 or more iterations it is the most robust amongst all settings, but its accuracy is still hampered. On the pooled dataset it performs slightly better than the baseline. Instead setting the first search direction in each optimization to be fully conjugate to the previous ($\gamma = 0$), leads to slightly improved robustness in all cases compared to the baseline. The accuracy is decreased in the single iteration setting, but the gap closes for 3 iterations and the accuracy is actually improved for 5 iterations.

This setting is Pareto optimal for 1, 3, and 5 iterations. On the pooled dataset, it outperforms the baseline for 3 and 5 iterations. Its average loss is very close to the baseline loss for 3, 5, and 100 iterations, whereas removing the conjugacy warm-start leads to significantly higher loss for 1, 3, and 5 iterations.

The alternative conjugacy mechanism, Fletcher-Reeves, provides inferior performance compared to the baseline for the single-iteration setting on VOT2018. However, its robustness increases with the number of iterations and surpasses the baseline for 3 iterations. On the pooled dataset, its performance is worse for 1 and 100 iterations and slightly improved for 3 and 5 iterations. The average loss is higher than that of the baseline in the 1 iteration setting, but the gap decreases with the number of iterations. Gradient descent, which lacks the conjugacy mechanism, leads to slow convergence, something reflected in the high average loss value. Performance is greatly diminished on the pooled dataset, and on VOT2018 for 1 and 3 iterations. For 5 iterations it achieves slightly improved robustness, and for 100 iterations improved accuracy at the cost of robustness.

The preconditioning is imperative in order to obtain a low loss. On the pooled dataset its removal results in lower performance for any number of iterations. On VOT2018, this leads to worse performance for 1 and 3 iterations, and improved accuracy at the cost of robustness. Hampering CG by reducing the step-lengths to 0.8α leads to reduced accuracy for 1 iteration. For 3 iterations this gap is diminished, and for 5 iterations this results in higher accuracy. For 100 iterations, they perform roughly the same. On the pooled dataset the hampering leads to reduced performance for 1 iteration, but improved performance for 3, 5, and 100 iterations. The average loss is close to that of the Fletcher-Reeves configuration.

The alternative optimization method, Barzilai-Borwein, leads to a higher loss than that of CG, close to that of removing the conjugacy warm-start from CG. On the pooled dataset, their performance is similar for 5 iterations, but Barzilao-Borwein is outperformed for 1, 3, and 100 iterations. On the VOT2018 dataset, Barzilai-Borwein leads to reduced performance compared to full conjugacy warm-start for 1, 3, and 5 iterations. For 100 iterations however, it provides significantly improved accuracy at the cost of robustness.

4.4 Analysis

The experiments suggest that the partial conjugacy warm-start heuristic in the baseline does not in general improve performance. Instead, making the initial search direction fully conjugate to the last search direction of the previous optimization improves performance for most cases, while resulting in a similar loss value. Not warm-starting the conjugacy seems to lead to slower convergence, but improved robustness if sufficiently many iterations are run. This remains a competetive alternative. The performance of the Fletcher-Reeves formula appears sensitive to the number of iterations used. Removing the conjugacy mechanism strongly deteriorates performance in most cases, as is expected. There is a surprising exception on VOT2018 for 5 iterations, where robustness slightly increased at a small cost of accuracy. Removing the preconditioner also leads to significant deterioration of performance, which is expected as normal equations

have a problemetic distribution of eigenvalues. The step-length multiplier did not hamper performance as much as expected, and actually improved performance for the 5-iteration case on both benchmarks. Tampering with the step-length leads to a method that does not converge to a solution. Possibly, the effect is attenuated with the repeated optimizations, and that running 5 iterations with the baseline leads to a case of overfitting that this heuristic is able to alleviate. For the case of visual tracking, conjugate gradient seems to outperform Barzilai-Borwein. It should be mentioned that the comparison is not entirely fair as the Barzilai-Borwein method has not been as thoroughly investigated as CG.

Overall, the full conjugacy warm-start ($\gamma = 0$) seems to provide a good trade-off between accuracy and robustness on VOT2018 while providing the best performance on the pooled dataset. In comparison, removing the conjugacy warm-start ($\gamma = \infty$) leads to a more robust but less accurate tracker. Both these settings outperform the baseline. The merit of the baseline is its performance from an optimization perspective, and if a very fast tracker is desired it is probably a good alternative.

The baseline attained the lowest loss for all iteration configurations. In the single-iteration setting it obtains the best performance on the pooled dataset while providing a very good trade-off between accuracy and robustness on VOT2018. For 3 iterations the Fletcher-Reeves, $\gamma = 0$, and 0.8α step-length settings provide the lowest loss values except for the baseline. These configurations obtains the top 3 best performance on the pooled dataset, and all provide a good accuracy and robustness on VOT2018. A lower loss does seem to provide increases to tracking performance in these cases, but as the number of iterations increase this ceases to be true. An explanation for this is overfitting, and as future work it may be beneficial to investigate strategies for stopping the optimization process when an acceptable solution has been obtained, instead of running the process for a fixed number of iterations.

5 Conclusion

In this paper we analyzed the optimization procedure of the popular ECO-tracker. The procedure was described in detail and several mechanisms were compared with their alternatives. Supported by experiments on the VOT2018-benchmark and a large pooled dataset, we showed the impact of the different configurations both in terms of tracking performance and in terms of optimization performance. We showed that a lower loss corresponds fairly well to improved tracking performance when the optimizers are run for a sufficiently low number of iterations. However, as the number of iterations increases, inferior optimizers may provide superior tracking performance.

Acknowledgments: This work was supported by Swedish Foundation for Strategic Research (SymbiCloud); Swedish Research Council (EMC 2 , starting grant 2016-05543); CENIIT grant (18.14); Swedish National Infrastructure for Computing; and Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

1. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA journal of numerical analysis **8**(1), 141–148 (1988)
2. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR (2010)
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)
4. Cehovin, L., Kristan, M., Leonardis, A.: Is my new tracker really better than yours? In: Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on. pp. 540–547. IEEE (2014)
5. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: CVPR (2017)
6. Danelljan, M., Hager, G., Khan, F.S., Felsberg, M.: Discriminative scale space tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence (2016)
7. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV (2015)
8. Danelljan, M., Robinson, A., Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV (2016)
9. Danelljan, M., Shahbaz Khan, F., Felsberg, M., van de Weijer, J.: Adaptive color attributes for real-time visual tracking. In: CVPR (2014)
10. Felsberg, M.: Enhanced distribution field tracking using channel representations. In: ICCV Workshop (2013)
11. Galoogahi, H.K., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 1134–1143. IEEE (2017)
12. Galoogahi, H.K., Sim, T., Lucey, S.: Correlation filters with limited boundaries. In: CVPR (2015)
13. Gundogdu, E., Alatan, A.A.: Good features to correlate for visual tracking. IEEE Transactions on Image Processing **27**(5), 2526–2540 (2018)
14. He, Z., Fan, Y., Zhuang, J., Dong, Y., Bai, H.: Correlation filters with weighted convolution responses. In: ICCV Workshops. pp. 1992–2000 (2017)
15. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI **37**(3), 583–596 (2015)
16. Kristan, M., Leonardis, A., Matas, J., Felsberg, Pflugfelder, R., M., Čehovin, L., Vojír, T.and Hger, G., et al. The visual object tracking vot2016 challenge results. In: ECCV workshop (2016)
17. Kristan, M., Leonardis, A., Matas, J., Felsberg, Pflugfelder, R., M., Čehovin, L., Vojír, T.and Hger, G., et al. The visual object tracking vot2017 challenge results. In: ICCV workshop (2017)
18. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: Algorithms and benchmark. TIP **24**(12), 5630–5644 (2015)
19. Lukezic, A., Vojír, T., Zajc, L.C., Matas, J., Kristan, M.: Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision **126**(7), 671–688 (2018). https://doi.org/10.1007/s11263-017-1061-3, https://doi.org/10.1007/s11263-017-1061-3
20. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV (2015)
21. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, 2nd edn. (2006)

22. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., Pittsburgh, PA, USA (1994)
23. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. TPAMI **37**(9), 1834–1848 (2015)