Adaptive Decontamination of the Training Set: A Unified Formulation for Discriminative Visual Tracking

Martin Danelljan, Gustav Häger, Fahad Khan and Michael Felsberg

Conference article

Cite this conference article as:
Danelljan, M., Häger, G., Khan, F., Felsberg, M. Adaptive Decontamination of the Training Set: A Unified Formulation for Discriminative Visual Tracking, In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Institute of Electrical and Electronics Engineers (IEEE); 2016, pp. 1430-1438.
ISBN: 978-1-4673-8851-1

DOI: https://doi.org/10.1109/CVPR.2016.159

IEEE Conference on Computer Vision and Pattern Recognition, ISRN: 1063-6919, 2016

Copyright: IEEE

The self-archived postprint version of this conference article is available at Linköping University Institutional Repository (DiVA):
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-137882

Tweet

LINKÖPING UNIVERSITY

Adaptive Decontamination of the Training Set: A Unified Formulation for Discriminative Visual Tracking

Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, Michael Felsberg
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Sweden
{martin.danelljan, gustav.hager, fahad.khan, michael.felsberg}@liu.se

Abstract

Tracking-by-detection methods have demonstrated competitive performance in recent years. In these approaches, the tracking model heavily relies on the quality of the training set. Due to the limited amount of labeled training data, additional samples need to be extracted and labeled by the tracker itself. This often leads to the inclusion of corrupted training samples, due to occlusions, misalignments and other perturbations. Existing tracking-by-detection methods either ignore this problem, or employ a separate component for managing the training set.

We propose a novel generic approach for alleviating the problem of corrupted training samples in tracking-by-detection frameworks. Our approach dynamically manages the training set by estimating the quality of the samples. Contrary to existing approaches, we propose a unified formulation by minimizing a single loss over both the target appearance model and the sample quality weights. The joint formulation enables corrupted samples to be down-weighted while increasing the impact of correct ones. Experiments are performed on three benchmarks: OTB-2015 with 100 videos, VOT-2015 with 60 videos, and Temple-Color with 128 videos. On the OTB-2015, our unified formulation significantly improves the baseline, with a gain of 3.8% in mean overlap precision. Finally, our method achieves state-of-the-art results on all three datasets.

1. Introduction

Generic visual tracking is the problem of estimating the trajectory of a target in an image sequence, given only its initial location. Tracking methods serve as important components in a variety of vision systems. The problem is particularly challenging due to the limited prior knowledge about the target. Furthermore, the tracking model must be flexible to counter rapid target appearance changes, while being robust to, *e.g.*, occlusions and background clutter.

The above mentioned problems have been addressed by methods based on the tracking-by-detection paradigm [2, 11, 13, 29], with promising results in recent years. In this paradigm, tracking methods employ machine learning techniques to train an appearance model based on samples of the target and its background. Typically, supervised learning methods such as Support Vector Machines (SVMs) or ridge regression are used to construct a discriminative classifier or regressor. The quality of the tracking model is directly dependent on the training set. Therefore, a robust approach for constructing and managing the training set is crucial for avoiding model drift and tracking failure.

Standard tracking-by-detection approaches struggle when the training set is contaminated by corrupted samples. These corrupted samples are included in the training set in several different scenarios. Firstly, when encountered with target deformation and out-of-plane rotation, inaccurate tracking predictions lead to misaligned training sam-

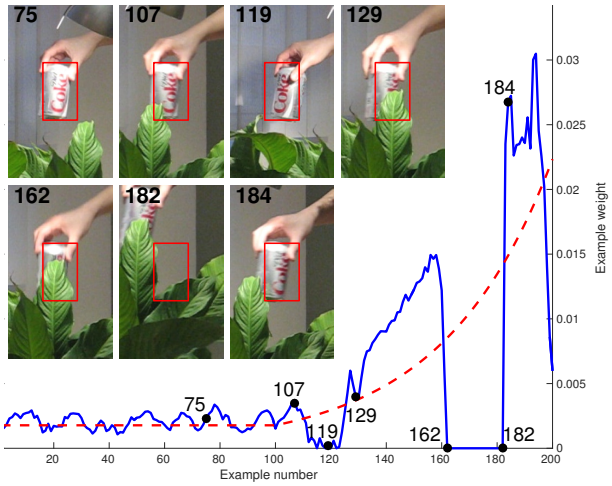


Figure 1. An illustration of our adaptive decontamination of the training set. We show the corresponding image patches and tracking predictions (red box) for selected training samples. The quality weights (blue), computed by our learning approach, determine the impact of the corresponding training samples (numbered in chronological order). The prior sample weights are plotted in red. Our approach down-weights samples that are misaligned (no. 119), partially occluded (no. 129) or fully occluded (no. 162-182).

ples (no. 119 in figure 1). Consequently, the model often drifts, eventually leading to tracking failure. Secondly, occlusions and clutter contaminate the positive training samples with background information, thereby deteriorating the discriminative power of the model (no. 162 in figure 1). In this work, we aim to enhance the robustness of standard tracking-by-detection approaches by tackling the problem of *decontaminating* the training set.

Existing discriminative trackers either ignore the problem of corrupted samples [2, 11, 26] or employ an explicit training sample management component [9, 14, 16, 25, 29]. A straightforward approach is to directly discard samples that do not meet a certain criterion [1]. Other methods use a combination of experts [16, 29], a separate tracking model [14, 22] or distance comparisons [9] for managing the training set. In this paper, we argue that the standard two-component strategy is suboptimal due to the reliance on heuristics. Instead, we revisit the standard tracking-by-detection formulation with the aim of integrating the estimation of the sample quality weights in the learning.

1.1. Contributions

We propose a novel formulation for jointly learning the tracking model and the training sample weights. Our formulation is generic, and can be integrated into common supervised learning methods. In each frame, a joint loss is minimized to update both the model parameters *and* the importance weights. Our joint learning approach down-weights corrupted samples while increasing the importance of correct ones, as visualized in figure 1. Different from previous tracking methods, our unified formulation eradicates the need of an explicit sample management component.

To validate our approach, we perform extensive experiments on three benchmarks: OTB-2015 [27] with 100 videos, VOT-2015 [17] with 60 videos, and Temple-Color [20] with 128 videos. Our unified approach demonstrates a significant gain of 3.8% in mean overlap precision on OTB-2015, compared to the baseline. Further, our tracker achieves state-of-the-art results on all three datasets.

2. Discriminative Tracking Methods

In recent years, discriminative tracking-by-detection approaches [2, 11, 13, 25, 29] have shown promising results on benchmarks, such as OTB [28] and VOT [18]. The appearance model within a tracking-by-detection framework is typically based on a discriminatively trained regressor [2, 11, 13] or classifier [25, 29]. These approaches are formulated in a supervised learning setting, where labeled training samples are collected from the sequence. Given a set of n training examples $\{(x_j, y_j)\}_{j=1}^{n}$, the aim is to find the parameters $\theta \in \Omega$ of the appearance model. Here, $x_j \in \mathcal{X}$ denotes a feature vector in the sample space $\mathcal{X}$ and $y_j \in \mathcal{Y}$ is the corresponding label in the label set $\mathcal{Y}$. Many supervised learning methods in tracking [2, 11, 13, 29] find the parameter values θ by minimizing a loss of the form,

$$J(\theta) = \sum_{k=1}^{n} L(\theta; x_j, y_j) + \lambda R(\theta). \tag{1}$$

Here, $L : \Omega \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ specifies the loss $L(\theta; x_j, y_j)$ for a training sample (x_j, y_j) as a function of the parameters θ. The impact of the regularization function $R : \Omega \rightarrow \mathbb{R}$ is controlled by the constant weight $\lambda \geq 0$.

Eq. (1) covers a variety of learning approaches, including support vector machines (SVMs) [11, 25, 29] and discriminative correlation filters (DCFs) [3, 4, 5, 13]. A common approach [16, 25, 29] is to use a two-class learning strategy to differentiate between the target $y_j = 1$ and background $y_j = -1$. Alternatively, the DCF based trackers [5, 13], utilize continuous labels $y_j \in [0, 1]$ or let y_j be the desired confidence map over an image region. Another strategy [11] is to let $\mathcal{Y}$ be the possible transformations of the target box.

2.1. Training Sample Weighting

In discriminative tracking, the model is learned using training samples collected from the video frames. Typically, the training set is updated with new samples in each frame, to account for changes in the target and background appearance. We rewrite (1) to highlight this temporal sampling used in many tracking methods. Let (x_{jk}, y_{jk}) denote the jth training sample in frame number k. Assume that n_k samples from frame $k \in \{1, \dots, t\}$ are included in the training set, where t denotes the current frame number. Typically, both positive and negative samples (x_{jk}, y_{jk}) are extracted in a frame k, based on the estimated target location. The loss (1) is then expressed in the more general form,

$$J(\theta) = \sum_{k=1}^{t} \alpha_k \sum_{j=1}^{n_k} L(\theta; x_{jk}, y_{jk}) + \lambda R(\theta). \tag{2}$$

Here, the constant weights $\alpha_k \geq 0$ control the impact of samples from frame k. By increasing α_k, a greater impact is given to samples $\{(x_{jk}, y_{jk})\}_{j=1}^{n_k}$ extracted from frame k.

There exist several strategies to control the impact of training samples in (2). In DCF-based trackers [2, 13], a learning rate parameter $\gamma \in [0, 1]$ is employed to update the weights as $\alpha_k = (1-\gamma)\alpha_{k+1}$. Such a re-weighting strategy aims to reduce the impact of older samples in the learning. Trackers based on SVMs typically prune the training set by *e.g.* rejecting samples older than a threshold [26] or removing support vectors with the least impact [11]. However, these methods do not account for the problem of corrupted samples (x_{jk}, y_{jk}) in the training set.

2.2. Corrupted Training Samples

Contrary to object detection, the problem of corrupted training samples is commonly encountered in tracking. The

problem appears since the samples are not hand-labeled, but rather labeled by the tracking algorithm itself. Several factors contribute to the unintentional inclusion of corrupted training samples in the learning. (a) Inaccurate tracking predictions, due to *e.g*. target rotations or deformations, lead to misaligned samples. This can result in model drift or tracking failure. (b) Partial or full occlusions of the target lead to positive samples being corrupted by the occluding objects. This is a common source of tracking failure, since the appearance model is contaminated due to occlusions. (c) Perturbations, such as motion blur, can lead to a distorted view of the target. These factors contribute to the inclusion of corrupted training samples in the learning, thereby deteriorating the discriminative power of the model.

State-of-the-Art: Several recent works have investigated the problem of corrupted training samples in the tracking-by-detection paradigm [1, 16, 25, 29]. Bolme *et al*. [1] propose to reject new samples based on the Peak-to-Sidelobe Ratio (PSR) criterion. PSR is computed as the ratio between the maximum confidence score and the standard deviation of the surrounding scores (outside a specified neighborhood of the peak). Zhang *et al*. [29] use an entropy-based minimization to determine the best model in an expert ensemble. The ensemble consists of the current tracking model and snapshots from earlier frames. If a disagreement occurs, the expert with the minimum entropy criterion is selected as the new tracker model. Kalal *et al*. [16] tackle the drift problem by generating positive and negative samples based on spatial and temporal constraints. Supančič and Ramanan [25] propose a strategy for updating the training set by revisiting previously rejected samples. Hong *et al*. [14] use a key-point based long-term memory component to detect occlusions and refresh the short-term memory.

Differences to Our Approach: As discussed above, existing tracking-by-detection approaches tackle the problem of corrupted samples with a dedicated separate component. This component is either based on distance comparisons [9], heuristics [1, 30], a set of experts [16, 29], a separate tracking model [14], or model fitting [25]. Our approach differs from the aforementioned methods in several aspects. To the best of our knowledge, we are the first to propose a learning formulation that jointly optimizes the model parameters and the sample weights. Instead of binary decisions [1, 16, 25, 29], our approach is based on continuous importance weights. This enables us to down-weight the impact of corrupted training samples, while up-weighting correct ones. Further, our method allows mistakes to be corrected by redetermining the sample weights at each frame.

3. Our Approach

Here, we propose our formulation for jointly learn the appearance model and the training sample weights in a tracking-by-detection framework.

3.1. Motivation

To motivate our approach, we first distinguish three desirable characteristics to be considered when designing a method for decontaminating the training set.

Continuous weights: Most existing discriminative trackers [1, 16, 25, 29] rely on binary decisions for including or removing potential training samples. This is problematic in ambiguous scenarios, such as moderate occlusions or slight misalignments (see figure 1), where the extracted samples are not entirely corrupted and still contain valuable information. Instead, continuous quality weights are expected to more accurately capture the importance of such samples.

Re-determination of Importance: A common approach is to determine the importance of a sample based on previous frames only, *e.g*. rejecting new samples based on the current appearance model [1]. Ideally, all available information should be considered when updating the importance of a specific training sample, including more recent frames. By exploiting information from *all* observed frames, the importance of older samples can be re-determined more accurately. This will enable previous mistakes to be corrected at a later stage in the tracking process.

Dynamic Sample Prior: Methods purely based on bottom-up statistics ignore prior knowledge associated with the samples. In cases of rapid target deformations and rotations, the tracker should emphasis recent samples for robustness. Dynamic prior knowledge is complementary to bottom-up information, and is expected to improve performance.

3.2. Problem Formulation

Our approach jointly estimates both the model parameters θ and the weights α_k. This is achieved by minimizing a single loss function $J(\theta, \alpha)$ to learn both the appearance model θ and the training sample weights $\alpha = (\alpha_1, \dots, \alpha_t)$. To the best of our knowledge, we are the first to cast the problem of determining the sample quality in a joint optimization framework. We introduce the joint loss $J(\theta, \alpha)$,

$$J(\theta, \alpha) = \sum_{k=1}^{t} \alpha_k \sum_{j=1}^{n_k} L(\theta; x_{jk}, y_{jk}) + \frac{1}{\mu} \sum_{k=1}^{t} \frac{\alpha_k^2}{\rho_k} + \lambda R(\theta) \tag{3a}$$

$$\text{subject to} \quad \alpha_k \geq 0\,, \; k = 1, \dots, t \tag{3b}$$

$$\sum_{k=1}^{t} \alpha_k = 1. \tag{3c}$$

Different from the standard weighted loss (2), our formulation (3a) is a function of both the model parameters θ *and* the sample weights α_k. As a result, the weights α_k are no longer pre-determined constants. The constrains (3b) and (3c) ensure that the weights α_k are non-negative and sum up to one. The second term in the joint loss (3a) is a regularization term on the sample weights α. This regularization

is controlled by the flexibility parameter $\mu > 0$ and the prior sample weights $\rho_k > 0$, satisfying $\sum_k \rho_k = 1$. The parameter μ controls the adaptiveness of the example weights α. Increasing μ leads to a higher degree of flexibility in the weights α. We analyze the effect of μ and ρ_k by considering the extreme cases of increasing ($\mu \to \infty$) and reducing ($\mu \to 0$) the flexibility parameter.

The case $\mu \to \infty$: This corresponds to removing the second term in (3a), implying no regularization on α. For a fixed model parameter θ, the loss (3) is then minimized by setting $\alpha_m = 1$ for the frame m with the smallest total loss $\sum_{j=1}^{n_m} L(\theta; x_{jm}, y_{jm})$ and setting $\alpha_k = 0$ for $k \neq m$. The model will thus overfit to samples from a single frame $k = m$, if the second term in (3a) is removed. Therefore, it is imperative to use a regularization on the weights α.

The case $\mu \to 0$: By introducing Lagrange multipliers, it can be shown that $\alpha_k \to \rho_k$ when $\mu \to 0$, for a fixed θ.[1] Thus, reducing the parameter μ also reduces the flexibility of the weights α_k about the prior weights ρ_k. The standard weighted loss (2) is therefore obtained in the limit $\mu \to 0$ by setting $\alpha_k = \rho_k$. Our approach can be seen as a generalization of (2) by introducing flexible sample weights α_k.

3.3. Optimization

Here, we propose a strategy for solving the joint learning problem (3). Our approach iteratively minimizes the loss by alternating between the model parameters θ and the example weights α. This strategy is motivated by the fact that (3) is convex in the weights α, given any fixed θ. Further, many existing supervised learning approaches, such as SVM and DCF, rely on convex optimization problems (1). It can be directly verified that (3) is biconvex if the weighted loss (2) is convex. That is, the optimization problem obtained by fixing either θ or α in (3) is convex. For biconvex problems, a standard approach is to use Alternate Convex Search (ACS) [10]. In each frame, we perform N ACS iterations to minimize our formulation (3). In each iteration, we solve the two convex subproblems obtained by fixing either α or θ in (3). We call these steps "Update θ" and "Update α".

Update θ: We first describe the subproblem of finding the optimal θ given a fixed $\alpha = \alpha^{(i-1)}$. Here, $\alpha^{(i-1)}$ denotes the estimate of the weights α in iteration $i-1$ of the optimization. In the first iteration $i = 1$, the weights $\alpha^{(0)}$ are initialized using estimates from the previous frame. The subproblem obtained by fixing the weights $\alpha = \alpha^{(i-1)}$ in (3) corresponds to optimizing the weighted loss (2) with respect to θ. This generates an updated model $\theta^{(i)}$. The optimization (2) is performed by the standard training method of the applied learning approach.

Update α: The second step of iteration i corresponds to optimizing (3) with respect to α, while keeping $\theta = \theta^{(i)}$ fixed. By defining the total loss in frame k by $L_k^{(i)} = \sum_{j=1}^{n_k} L(\theta^{(i)}; x_{jk}, y_{jk})$, the resulting subproblem is,

$$\text{minimize} \quad J_2^{(i)}(\alpha) = \sum_{k=1}^{t} L_k^{(i)} \alpha_k + \frac{1}{\mu} \sum_{k=1}^{t} \frac{\alpha_k^2}{\rho_k} \tag{4a}$$

$$\text{subject to} \quad \alpha_k \geq 0\,, \;\; k = 1, \ldots, t \tag{4b}$$

$$\sum_{k=1}^{t} \alpha_k = 1. \tag{4c}$$

The above optimization problem can be efficiently solved with convex quadratic programming methods. We use the corresponding function in Matlab's Optimization Toolbox, which internally employs the interior point method.

Algorithm 1 Our approach: tracking in frame t

Input: Current model parameters θ and weights $\{\alpha_k\}_{k=1}^{t-1}$.
Current training set $\{(x_{jk}, y_{jk})\}_{j=1,k=1}^{n_k,t-1}$.

1: Estimate the target location in frame t using θ.
2: Extract training samples $\{(x_{jt}, y_{jt})\}_{j=1}^{n_t}$ in frame t.
3: Update the prior weights $\{\rho_k\}_{k=1}^{t}$ using, *e.g.*, (5).
4: Initialize weights $\alpha_k^{(0)} = \alpha_k$ for $k < t$ and $\alpha_t^{(0)} = \rho_t$.
5: **for** $i = 1, \ldots, N$ **do**
6: **Update θ:** Find $\theta^{(i)}$ by optimizing (2) using $\alpha^{(i-1)}$.
7: **Update α:** Find $\alpha^{(i)}$ by solving (4) given $\theta^{(i)}$.
8: **end for**

3.4. Prior Weights Selection

As discussed in section 3.1, it is desirable to encode prior knowledge about the sample weights α_k in the learning. In our approach, this prior information is incorporated using the prior weights ρ_k, which serve as a regularizer for the sample weights α_k. The impact of the prior weights ρ_k are further controlled by the flexibility parameter μ. We propose a simple, yet effective strategy for setting the sample weights ρ_k based on solely temporal information. In our strategy, recent samples are given larger prior weights to account for fast appearance changes. In general, additional information about the sampling process, such as the number of training samples n_k in frame k, can be integrated into ρ_k.

We use a learning rate parameter $\eta \in [0, 1]$ to determine the prior weights for the K most recent frames, such that $\rho_k = (1-\eta)\rho_{k+1}$ for $k = t-K, \ldots, t-1$. The prior weights for all frames older than $t-K$ are set to constant, *i.e.* $\rho_k = \rho_{k+1}$ for $k < t-K$. The above recursive definition implies the formula

$$\rho_k = \begin{cases} a\,, & k = 1, \ldots, t-K-1 \\ a(1-\eta)^{t-K-k}\,, & k = t-K, \ldots, t. \end{cases} \tag{5}$$

Here, the constant $a = \left(t - K + \frac{(1-\eta)^{-K}-1}{\eta}\right)^{-1}$ is determined by the condition $\sum_k \rho_k = 1$. The prior weights ρ_k

[1] The proof is provided in the supplementary material.

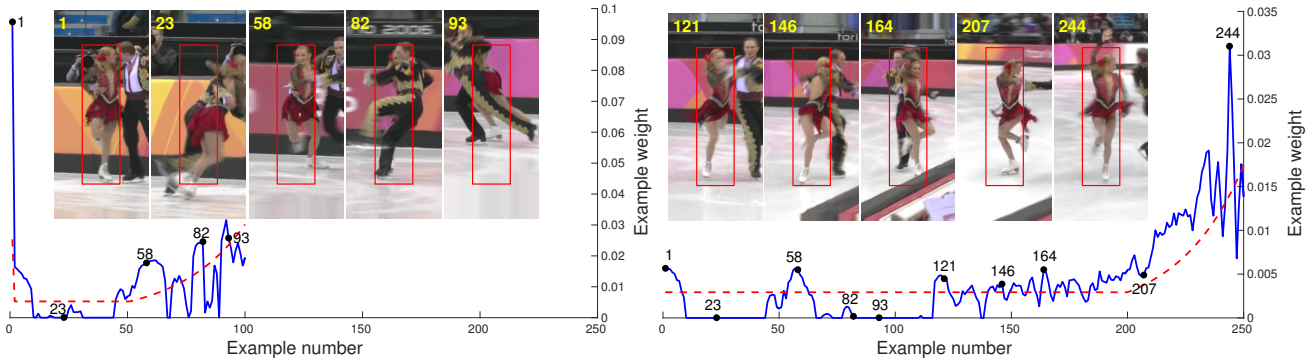


Figure 2. The training sample impact weights computed by our joint learning formulation on the *Skating* sequence. The computed weights α_k (blue curve) and corresponding prior weights ρ_k (red curve) are plotted for two different time instances during the tracking process: in frame 100 (left) and frame 250 (right). Image patches and corresponding target estimations (red box) are shown for example frames. The parameters are set as described in section 5.1. A few training examples (*e.g.* no. 82 and 93) that are corrupted by an occluding object (the male skater) are initially assigned large weights (left). By efficiently redetermining all impact weights α_k in each frame, previous mistakes are corrected. In this example, the corrupted samples (no. 82 and 93) are down-weighted at a later stage (right). On the other hand, accurate training samples (no. 1 and 58) are consistently assigned large impact weights.

in (5) put a larger emphasis on recent frames to alleviate the problem of rapid appearance changes, caused by *e.g.* target deformations and out-of-plane rotations. Instead of letting the prior weights decrease towards zero for older samples, we assign all samples older than K frames equal prior importance. This allows a significant influence of old training samples in the learning. Algorithm 1 provides an overview of our method in a generic setting.

4. The Tracking Framework

Here, we describe a tracking-by-detection framework using the unified learning formulation proposed in section 3.

4.1. Baseline Tracker

In recent years, the Discriminative Correlation Filter (DCF) based trackers have shown excellent performance on several benchmark datasets [2, 5, 13, 18]. These trackers apply Fourier transforms to efficiently train a discriminative regressor on sample feature maps extracted from the video frames. We employ the recent SRDCF [5] as our base supervised learning approach. Unlike other DCF methods, SRDCF employs a spatial regularization in the learning to alleviate the periodic effects induced by circular correlation.

The appearance model of the SRDCF tracker consists of a discriminatively trained convolution filter. In each new frame, a confidence map is first computed by applying the filter around the predicted target location. This confidence map is then maximized to estimate the target location. A single training example (x_k, y_k) is added in each frame k. The sample x_k is a d-dimensional feature map, extracted around the target, that also includes the surrounding background information. The Gaussian label function y_k contains the desired confidence map, when applying the sought convolution filter f_θ on x_k. In the SRDCF, the model parameters θ thus consist of the filter coefficients in f_θ. The standard SRDCF employs the weighted learning formulation (2), with a per-sample loss L given by,

$$L(\theta; x_k, y_k) = \left\| y_k - \sum_{l=1}^{d} f_\theta^l * x_k^l \right\|^2 . \tag{6}$$

Here, $*$ denotes circular convolution and the superscript x_k^l and f_θ^l denotes the lth channel of x_k and f_θ respectively. The loss (6) consists of the total squared error between the desired confidence map y_k and the confidence scores obtained by applying the filter f_θ to the sample x_k.

The regularization $R(\theta)$ is determined by the spatial penalty function w, consisting of a positive penalization factor at each spatial location in the filter,

$$R(\theta) = \sum_{l=1}^{d} \left\| w \cdot f_\theta^l \right\|^2 . \tag{7}$$

Here, $\cdot$ denotes pointwise multiplication. The regularization ensures a limited spatial extent of the filter by penalizing coefficients outside the target region. The filter f_θ is trained by transforming the loss (2) to a real-valued Fourier basis and solving the resulting normal equations. For more details about the SRDCF tracker, we refer to [5].

4.2. Proposed Tracker

Here, we apply our unified learning formulation to the baseline tracker. To learn the appearance model, the baseline tracker minimizes the weighted loss (2), using exponentially decaying sample weights α_k. Instead, we minimize the proposed unified formulation (3) to jointly estimate both the model θ and the sample weights α_k, in each frame.

The proposed tracker follows the outline in algorithm 1. In a new frame t, we first estimate the target location as in

the baseline tracking approach. The training set is then augmented with the new sample (x_t, y_t). The prior weights ρ_k are then updated as in (5). The importance weight of the new sample is initialized with its prior weight $\alpha_t^{(0)} = \rho_t$. The weights $\alpha_1^{(0)}, \dots, \alpha_{t-1}^{(0)}$ of earlier samples are initialized with their estimates from the previous frame and then normalized such that $\sum_k \alpha_k^{(0)} = 1$. To minimize the joint loss (3), we then perform an "Update θ" step followed by an "Update α" step in each iteration i of the optimization, as described in section 3.3. As mentioned in section 3.3, our joint learning formulation (3) is biconvex since the weighted loss (2) is convex.

Update θ: The updated filter $f_\theta^{(i)}$ is computed using the training procedure in [5], given the weights $\alpha_k^{(i-1)}$. Instead of the incremental update, we use the general formula in [5] to compute the normal equations. This enables arbitrary weights to be used. A fixed number of Gauss-Seidel iterations are then performed, with the current filter $f_\theta^{(i-1)}$ as an initial solution, to obtain the new filter $f_\theta^{(i)}$.

Update α: The new filter $f_\theta^{(i)}$ is then used to redetermine sample weights $\alpha^{(i)}$. Since each frame only contains a single sample, the total loss in frame k is given by $L_k^{(i)} = L(\theta^{(i)}; x_k, y_k)$. This is efficiently computed using the Fast Fourier Transform (FFT), by applying Parseval's formula to (6). The new weights $\alpha^{(i)}$ are then computed by solving the quadratic programming problem (4).

To achieve an upper bound on the memory consumption, we store a maximum number T of training samples. If the number of samples exceeds T, we simply remove the sample $k \leq K$ that has the smallest weight α_k. Figure 2 shows the estimated quality weights α_k for an example sequence.

5. Experiments

To evaluate our approach, we perform comprehensive experiments on three benchmark datasets: OTB-2015 [27], VOT-2015 [17] and Temple-Color [20].[2]

5.1. Parameter Settings

The prior sample weights are set as described in section 3.4, using $K = 50$ and $\eta = 0.035$. In general, the flexibility parameter μ depends on the scale of the loss function $L(\theta; x, y)$ for different discriminative methods. This dependency can however be mitigated by appropriate normalization of L with, *e.g.*, respect to the average number of samples n_k per frame. We use $\mu = 5$ in our experiments, which enables a large degree of adaptiveness in the weights (see figure 1 and 2). The maximum number of stored training samples is set to $T = 300$. In the tracking scenario, the joint loss (3) is modified marginally in each frame by adding the new training samples for frame t. Therefore, we found a single ($N = 1$) ACS iteration to be sufficient to refine the estimate of θ and α from the previous frame. This further ensures a minimal increase in computations compared to the original learning approach. Our joint learning is started at $t = 10$ frames into the sequence. This ensures a sufficient number training samples for our learning procedure.

[2]Detailed results are presented in the supplementary material.

	Baseline [5]	Baseline-Entropy	Baseline-PSR	**Ours**
Mean OP	72.9	72.2	74.4	**76.7**

Table 1. A comparison of our approach, using mean OP (%), with the baseline methods on the OTB-2015 dataset. The baseline tracker does not account for corrupted samples. We also compare our approach by incorporating the entropy and PSR strategies in the baseline tracker. The best result is displayed in red font. Our approach achieves a significant performance gain of 3.8% in mean OP, compared to the baseline tracker.

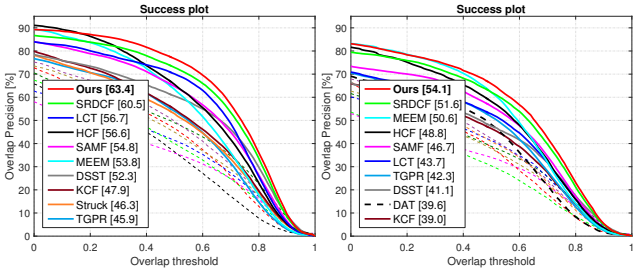
For the baseline tracker [5], we use the Matlab implementation provided by the authors. For a fair comparison, we use the same parameter settings for both our tracker and the compared baseline SRDCF. For the OTB-2015, we use HOG features for both our and the baseline tracker, as in [5]. For VOT-2015 and Temple-Color, we employ the same combination of HOG and Color Names for both trackers and use $\mu = 3$ and $T = 200$ in our method. Furthermore, we fix the parameter settings for all videos in each dataset. In our approach, solving the quadratic programming problem (4) in the "Update α" step, is highly efficient. It takes around 5 milliseconds on a standard desktop computer. The computational cost of our tracker is completely dominated by the baseline training procedure used in the "Update θ" step. We obtain a slightly reduced frame rate (around 3 frames per second) compared to the baseline tracker.

5.2. Baseline Experiments

We first compare our approach with the baseline SRDCF tracker, which does not account for corrupted training samples. We also integrate two existing training sample management strategies into the baseline tracker, for additional comparisons. The first strategy [1] is based on the Peak to Sidelobe ratio (PSR) of the tracking confidence scores. It is computed as the ratio $\frac{g_{\max} - m_r}{\sigma_r}$, where $g_{\max}$ is the maximum confidence, m_r is the mean confidence and σ_r is the standard deviation of the confidence scores outside the peak. The second strategy [29] is based on an expert ensemble of previous snapshots of the tracking model. In each frame, the confidence scores are first computed for all experts. If the target location estimates differ, an entropy based score is used to rank the experts in the ensemble. The current tracking state is then set to the expert with the highest score. This corresponds to resetting the tracker model to the best previous state. New snapshots are stored periodically, while discarding the oldest one. For a fair comparison, we optimize the parameters for the PSR and entropy based strategies.

	EDFT[7]	LSHT[12]	DFT[24]	ASLA[15]	TLD[16]	Struck[11]	CFLB[8]	ACT[6]	TGPR[9]	KCF[13]	DSST[2]	SAMF[19]	DAT[23]	MEEM[29]	LCT[22]	HCF[21]	SRDCF[5]	**Ours**
OTB-2015	41.4	40.0	35.9	49.0	46.5	52.9	44.9	49.6	54.0	54.9	60.6	64.7	36.4	63.4	70.1	65.5	*72.9*	**76.7**
Temple-Color	41.2	29.0	34.0	38.8	38.4	40.9	37.8	42.1	51.6	46.5	47.5	56.1	48.2	*62.2*	52.8	58.2	*62.2*	**65.8**

Table 2. A comparison of our approach, using mean OP (%), with state-of-the-art trackers on the OTB-2015 and Temple-Color datasets. The best two results are shown in red and blue font respectively. On the OTB-2015 dataset, the best existing tracker provides a mean OP of 72.9%. On the Temple-Color dataset, both SRDCF and MEEM obtains a mean OP score of 62.2%. Our approach obtains state-of-the-art results, outperforming the best existing trackers by 3.8% and 3.6%, on the OTB-2015 and Temple-Color datasets, respectively.



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

Figure 3. Success plots for the OTB-2015 (a) and Temple-Color (b) datasets. For clarity, we only show the top 10 trackers in the legend. On the OTB-2015 and Temple-Color, our approach achieves state-of-the-art results with a gain of 2.9% and 2.5% in AUC respectively, compared to the best previous method.

We report the results using mean overlap precision (OP). The OP is computed as the percentage of frames where the intersection-over-union (IOU) overlap with the ground-truth exceeds a threshold of 0.5. Table 1 shows the mean OP results over all the 100 videos of OTB-2015 dataset. The baseline SRDCF tracker obtains a mean OP score of 72.9%. The PSR strategy improves the results with a mean OP score of 74.4%. Our approach significantly improves the performance by providing a gain of 3.8% in mean OP, compared to the baseline tracker. The substantial improvement over the baseline demonstrates the importance of decontaminating the training sample set. It is worth to mention that our approach is generic, and can be incorporated into other discriminative tracking frameworks.

We also validate the generality of our approach by applying the proposed joint learning formulation to an SVM-based discriminative model. SVMs have been successfully applied for tracking-by-detection in recent years [11, 25, 29]. We use a binary linear SVM, where $L(\theta; x, y)$ is the standard hinge-loss. As in the SRDCF case, we use the outline described in algorithm 1 and set the prior weights ρ_k as in (5). In each frame k, we collect 1 positive and about 20 negative samples (x_{jk}, y_{jk}) from the estimated target neighborhood, using the color-based feature representation [29]. For the baseline SVM tracker, we fix the sample weights as $\alpha_k = \rho_k$. For our SVM tracker, we minimize the loss (3) as described in section 3.3. The same parameter settings is used for both the baseline and our version of the SVM-based tracker. On OTB-2015, the baseline SVM tracker provides a mean OP of 58.2%. Our SVM tracker achieves a significant gain of 3.2%, with a mean OP of 61.4%.

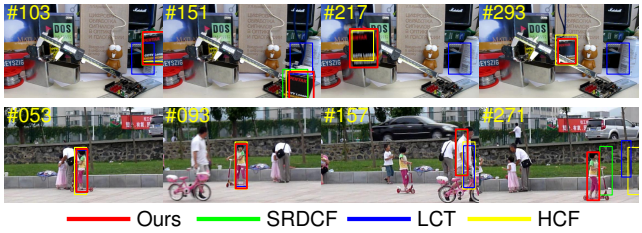


Figure 4. A qualitative comparison of our approach with state-of-the-art methods on the *Box* (top row) and *Girl* (bottom row) videos. Our approach accurately re-detects the target in the *Girl* video due to a decontaminated training set (last frame).

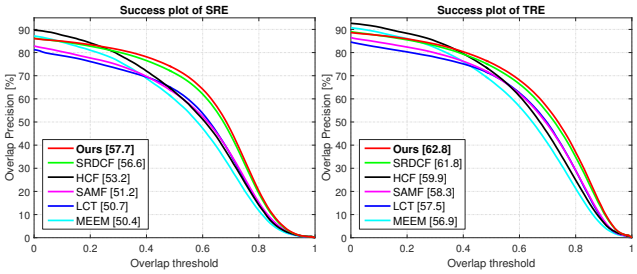


Figure 5. Robustness to initialization comparison on the OTB-2015 dataset. Success plots are shown for both spatial (SRE) and temporal (TRE) robustness. Our tracker provides consistent improvements in both cases, compared previous approaches.

5.3. OTB-2015 Dataset

We perform a comprehensive comparison with 17 recent state-of-the-art trackers: EDFT [7], LSHT [12], DFT [24], ASLA [15], TLD [16], Struck [11], CFLB [8], ACT [6], TGPR [9], KCF [13], DSST [2], SAMF [19], DAT [23], MEEM [29], LCT [22], HCF [21] and SRDCF [5].

5.3.1 State-of-the-art Comparison

A comparison with state-of-the-art trackers on the OTB-2015 is shown in Table 2 (first row). We report the mean OP over all the 100 videos. The MEEM tracker, with an entropy minimization based sample management, obtains a mean OP of 63.4%. The hierarchical convolutional features (HCF) tracker provides a mean OP of 65.5%. Our approach significantly outperforms the best compared tracker, by achieving a mean OP of 76.7%.

Figure 3 contains the success plot, showing the mean OP over the range of overlap thresholds [27], on the OTB-2015 dataset. For each tracker, area-under-the-curve (AUC) score is displayed in the legend. Among the compared tracking

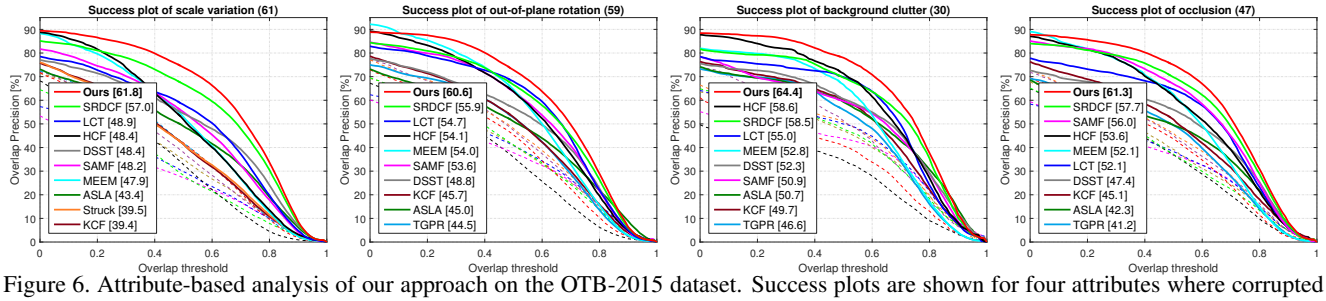


Figure 6. Attribute-based analysis of our approach on the OTB-2015 dataset. Success plots are shown for four attributes where corrupted samples are a common problem. For clarity, we only show the top 10 trackers in the legends. The title of each plot indicates the number of videos labelled with the respective attribute. Our approach provides consistent improvements compared to state-of-the-art methods.

methods, SRDCF, LCT and HCF provide the best results with AUC scores of 60.5%, 56.7% and 56.6% respectively. Our approach achieves the best results with an AUC score of 63.4%. Figure 4 shows a qualitative comparison with state-of-the-art methods on the *Box* and *Girl* videos. Our approach down-weights corrupted training samples, leading to accurate target re-detection (frame 271 in *Girl*).

5.3.2 Robustness to Initialization

To evaluate the robustness of our tracker, we follow the protocol proposed by [27]. The robustness is evaluated using two different initialization strategies: spatial robustness (SRE) and temporal robustness (TRE). The first criteria, SRE, is based on initializing the tracker at different perturbations of the initial ground-truth location. In case of TRE, the tracker is initialized at 20 different frames with the corresponding ground-truth. We present the success plots for SRE and TRE, on the OTB-2015, in Figure 5. We compare with the top 5 trackers. Our approach achieves robustness in both cases, leading to a consistent performance gain.

5.3.3 Attribute Based Analysis

In the OTB-2015, all videos are annotated with 11 different attributes. Our tracker outperforms previous approaches on all 11 attributes.[2] Figure 6 shows success plots for four attributes where corrupted samples are commonly included in the training set. In scenarios with challenging scale variations and out-of-plane rotations, inaccurate target estimations often lead to the inclusion of misaligned training samples. Our joint learning approach is capable of reducing or removing the impact of such samples, thereby lowering the risk of drift and tracking failure. In videos with significant background clutter or occlusions, positive training samples are often corrupted by background information. This a common cause for tracking failure in discriminative methods. By re-determining the sample weights in every frame using our joint formulation (3), the effect of corrupted training samples is mitigated by the learning process itself. The effectiveness of our approach is demonstrated by the superior results achieved in the aforementioned scenarios.

	Ours	SRDCF	MEEM	SAMF	ACT	DSST	KCF	CFLB	Struck	DFT	EDFT
AEO	**0.299**	*0.288*	0.221	0.202	0.186	0.172	0.171	0.152	0.141	0.140	0.139

Table 3. Comparison with state-of-the-art, based on expected average overlap (EAO), on the VOT-2015 dataset. Our approach provides improved performance compared to the best existing tracker.

5.4. VOT-2015 Dataset

In VOT-2015 [17], consisting of 60 challenging videos, trackers are evaluated in terms of expected average overlap. This measure is based on empirically estimating the average overlap (as a function of sequence length) and the typical-sequence-length distribution (cutting-off both lopes at a threshold such that the mass is 0.5). The measure itself is then obtained as the inner product of the two functions. Table 3 shows the average expected overlap (AEO) on VOT-2015 for methods with publicly available implementations.

5.5. Temple-Color Dataset

Finally, we perform experiments on the Temple-Color dataset with 128 videos. A comparison with state-of-the-art trackers is shown in Table 2 (second row). Among the compared methods, both MEEM and SRDCF obtains a mean OP of 62.2%. Our approach improves the state-of-the-art on this dataset with a mean OP of 65.8%. Figure 3 shows the success plot over all the 128 videos in the Temple-Color dataset. MEEM and SRDCF provide AUC scores of 50.6% and 51.6% respectively. Our tracker outperforms state-of-the-art approaches an AUC score of 54.1%.

6. Conclusions

We propose a unified learning formulation to counter the problem of corrupted training samples in the tracking-by-detection paradigm. Our approach efficiently down-weights the impact of corrupted training samples, while up-weighting accurate samples. The proposed approach is generic and can be integrated into other discriminative tracking frameworks. Experiments demonstrate that our approach achieves state-of-the-art tracking performance.

Acknowledgments: This work has been supported by SSF (CUAS), VR (EMC^2 and ELLIIT), the Wallenberg Autonomous Systems Program, the NSC and Nvidia.

References

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

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

[3] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Coloring channel representations for visual tracking. In *SCIA*, 2015.

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

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

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

[7] M. Felsberg. Enhanced distribution field tracking using channel representations. In *ICCV Workshop*, 2013.

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

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

[10] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Math. Meth. of OR*, 66(3):373–407, 2007.

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

[12] S. He, Q. Yang, R. Lau, J. Wang, and M.-H. Yang. Visual tracking via locality sensitive histograms. In *CVPR*, 2013.

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 2015.

[14] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *CVPR*, 2015.

[15] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012.

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

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

[18] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, and et al. The visual object tracking VOT 2014 challenge results. In *ECCV Workshop*, 2014.

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

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

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

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

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

[24] L. Sevilla-Lara and E. G. Learned-Miller. Distribution fields for tracking. In *CVPR*, 2012.

[25] J. S. Supančič and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013.

[26] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *ICCV*, 2007.

[27] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015.

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

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

[30] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012.