A Simple Adaptive Tracker with Reminiscences

Christopher Xie, Emily Fox, Zaid Harchaoui
University of Washington

Abstract—Correlation filters have provided exceptional results in the field of visual object tracking in the past few years. However, these methods typically learn a single filter to be robust to many different appearance changes, which can be challenging. We propose a simple solution to this problem by utilizing an ensemble method of base trackers trained on different temporal windows of the video history. The proposed tracker, called MTCF, exhibits the following features: i) it can be trained using gradient-based convex optimization; ii) it is robust to short-term and long-term changes in visual appearance. MTCF performs on par with or outperforms state-of-the-art trackers on the OTB and the VOT benchmark datasets. We present an extensive analysis of the performance of MTCF on these benchmark datasets.

I. INTRODUCTION

Visual tracking is a very important topic in robotic perception. Robots must be able to perceive and track manipulable objects, humans, and much more in order to understand the state of the world. In unknown environments, robots must be able to quickly learn to track potentially never-before-seen objects, which will allow them to perform fully autonomous tasks. This brings us to the problem of generic visual object tracking of a single arbitrary object. The task is to estimate the trajectory of the object throughout a video, given only a single ground truth bounding box in the first frame. The tracking algorithm must robustly estimate the trajectory of this bounding box throughout the video. Generic visual object tracking is difficult due to the changes in the object appearance such as rotation, scale variation, and deformation [1], [2].

In order to robustly track potentially never-before-seen objects, many approaches utilize the single ground truth bounding box to learn an appearance model, which they update in an online fashion as they track the object through the video. This allows the algorithm to adapt to changes in the appearance due to factors such as illumination variation, rotation, and deformation. A common family of methods that implements this approach is that of correlation filters. Popularized by the MOSSE (minimum output sum of squared errors) correlation filter [3], these methods operate by learning an object template by minimizing a least squares objective function on Fourier coefficients. At each frame, the learned template is applied to detect the object and the predicted object location is used to update the template. Because these algorithms operate in the Fourier domain, they allow for tracking at real-time speeds. Much progress has been made in advancing these correlation filters to include multiple channels [4], spatial regularization [5], and deep features [6], [7].

However, these correlation filter approaches contain a number of issues. For example, because the template is learned in the Fourier domain, the size of the object template is required to be the same as the search image, which is undesirable as the object is likely smaller. [5] proposes a method to fix this issue, but results in a complicated learning algorithm and loss in efficiency. Additionally, these approaches often aggregate the images over time [3] to learn a single template that slowly adapts to changes in appearance. Learning a single template with short memory can easily result in significant loss of performance. Such strategies are not designed to handle rapidly changing object appearances.

In this paper, we address these issues by proposing a simple correlation-filter-type visual object tracker with two components. First, we learn a single tracker over a short temporal window directly in the spatial domain. This allows the template to be appropriately sized, and the use of off-the-shelf gradient-based convex optimization. Furthermore, we propose an ensemble method that utilizes base trackers trained on different temporal windows, which we denote the Multi-Template Correlation Filter (MTCF). The proposed approach admits a pleasant simplicity and modularity while demonstrating performance that is comparable to state-of-the-art methods for visual object tracking. We demonstrate the effectiveness of MTCF by performing an extensive analysis on multiple datasets [9], [1], [2]. The code is publicly available online at https://github.com/chrisdxie/reminiscent_tracker.
II. RELATED WORK

a) Correlation Filters: Correlation Filters are a popular family of models used for visual object tracking. The MOSSE filter, introduced in [3], was one of the first works to demonstrate the efficacy of applying correlation filters to visual object tracking, running at high speeds on the order of hundreds of frames per second. These methods formulate a least squares problem in the Fourier domain to learn a filter from all circular shifts of the image. Subsequent works have extended this idea to include multiple channels [4], [10], scale estimation [11], and deep features [6], [7], [12], [13], [14]. Learning in the Fourier domain allows the resulting formulations to leverage the Convolution Theorem for efficiency; however, this requires the object template to be the same size as the search image. This causes the object template to be prone to overfitting to background noise, thus remedies have been proposed in the literature [15], [5]. We instead provide a simple formulation in the spatial domain that avoids these issues and consider appropriately sized filters while still providing fast and accurate tracking predictions.

b) Ensemble Methods in Tracking: Ensemble methods have successfully been used in tracking to handle object appearance variations. Nam et al. [16] manage an ensemble of convolutional neural networks (CNNs) in a tree structure, ranking among the top trackers in the VOT2016 competition [2]. Zhang et al. [17] keeps a history of “snapshots” of SVM-based trackers and uses an entropy minimization method to select the best tracker. Multi-template methods have been used in sparse methods as a means to model diversity in appearance [18], [19]. Similarly, Nam et al. [20] maintains a set of representative frames that inform prediction at each frame. Several methods have utilized methods (e.g. boosting methods) in order to combine weak classifiers into a strong tracker [21], [22], [23]. In contrast to these methods, MTCF explicitly maintains models of different temporal windows of the video history.

c) CNN-based Trackers: As CNNs have achieved exceptional results in the realm of image recognition [24], many tracking algorithms have adopted both the network structures and the learned feature representations from such networks. Nam et al. [25], the winning entry from the VOT2015 competition [26], proposed a multi-domain CNN where each head of the network corresponds to a different video. Siamese networks combined with correlation filter layers have also been shown to perform well in visual object tracking [27], [28], [29]. Trackers such as [30], [31], [25], [32] utilize external tracking data in an offline training stage. Many state-of-the-art approaches [12], [13] show impressive results by leveraging discriminative intermediate outputs of deep networks such as VGG [33], [34].

III. SPATIAL CORRELATION FILTERS

In this section, we describe a simple base tracker, denoted the spatial correlation filter (sCF). We discuss the difference of the proposed setup with the standard correlation filter setup.

A. Formulation

Let \( F \in \mathbb{R}^{h_f \times w_f \times d} \) be an object filter (synonymous to template) where \( d \) is the number of channels and \( h_f, w_f \) is the height and width of the filter, respectively. We learn \( F \) in an online fashion that mimics the standard correlation filter setup [3]. In particular, we solve the problem

\[
F^* = \arg\min_F \frac{1}{2} \sum_{t=1}^{N} \alpha_t \left\| Y_t - \sum_{k=1}^{d} [F]_k \ast [I_t]_k \right\|_2^2 + \frac{\lambda}{2} \|F\|_2^2
\]

where \( \ast \) denotes zero-padded convolution. Here, \( Y_t \in \mathbb{R}^{h \times w} \) is a desired response function, \( I_t \in \mathbb{R}^{h \times w \times d} \) is the \( t \)th image, \([I_t]_k\) is the \( k \)th channel of \( I_t \), \( \alpha_t \in \mathbb{R} \) is the weight for image \( t \), \( N \) is the number of images in the training set, and \( \lambda \) is the regularization parameter. The filter \( F \) is typically smaller in size than the image \( I_t \), i.e. \( h_f < h \), \( w_f < w \), which allows for the filter to be appropriately sized based on the object.

Following the correlation filter framework, \( Y_t \) is a Gaussian peaked at the center of the map, \( I_t \) is always a cropped image patch (sometimes called a search region), and \( \alpha_t \) are chosen such that it allows for the most recent images to be more heavily weighted than the past images.

This objective function is a convex function and can be efficiently solved by methods such as gradient descent. In all of our experiments, we opt to use L-BFGS with backtracking line-search [35] as it is quite efficient and does not require the user to supply parameters such as step size.

a) Online Learning and Tracking: In the paradigm of short-term single object tracking [9], [1], [2], the tracking algorithm is only provided the initial frame and the corresponding ground truth bounding box. To update the filter in an online fashion, we employ the standard online learning approach for visual object tracking: at frame 1, we start off with the single datapoint provided (i.e. \( N = 1 \)). At each subsequent frame, we predict translation by computing the response map \( \sum_{k=1}^{d} [F]_k \ast [I_t]_k \) at a search region centered at the previously predicted location, followed by selecting the location of the maximum. We then treat that prediction as ground truth and incorporate this new frame into the training set and re-solve Eq. (1). This allows for the filter \( F \) to be robust to multiple appearance variations of the object.

B. Relation to Standard Correlation Filters

Standard correlation filters [3] can be recovered by setting, \( h_f = h \), \( w_f = w \). In this setting, Eq. (1) can be efficiently solved in the Fourier domain by utilizing Parseval’s theorem, FFTs, and the Circular Convolution Theorem [3], which involves circular convolution. However, this requires the object filter to be appropriately sized; these methods effectively learn to model the background and are plagued with boundary effects [5]. Several works have proposed solutions to this that involve complicated algorithms [15], [5] at a reduced efficiency. Instead, the simplistic proposed formulation circumvents these issues by learning in the spatial domain. Although we lose efficiency, we show in Section V-C that sCF still provides fast and accurate predictions. Note that as we encounter more frames, we continually grow the
training data, thus \( N \) in Eq. (1) increases. This is in contrast to
typical correlation filters that learn from one training sample
\( I^* \) only, and aggregate observations at the \( t \)th frame in an
exponential fashion: \( I^* = (1 - \eta)I^* + \eta I_t \).

In addition to these advantages, by learning in the spatial
domain, we can consider arbitrary loss functions. As long
as the function is differentiable, we can solve adapted
versions of Eq. (1) with auto-differentiation and gradient-
based optimization methods. Thus, loss functions are more
flexible when learning in the spatial domain. For example,
using a tool such as TensorFlow [36], one could potentially
specify any differentiable function \( h(F, I_t) \) in place of
\( \sum_{k=1}^{d} [F]_k \ast [I_t]_k \) and take advantage of auto-differentiation
to calculate gradients for optimization. Other ideas can be
seamlessly integrated into the formulation.

IV. MULTI-TEMPLATE CORRELATION FILTER

While correlation filters have enjoyed strong performance
in tracking, they are often limited to learning a single rigid
filter, which is not ideal when tracking objects exhibiting
appearance variations. To remedy this, we propose a simple
algorithm denoted the Multi-Template Correlation Filter
(MTCF). MTCF maintains a collection of base trackers trained
on different temporal windows. While any tracker can be
employed as a base tracker, we use sCF from Section III for
performance and efficiency. To perform tracking, a response
map is generated by aggregating the response maps of the
individual base trackers with a weighted combination that
allows for the proposed tracker to realize new appearances
yet be robust to the old ones. Figure 1 provides a visual
description of the division of the video history.

Translation prediction for MTCF is computed by selecting
the argmax over a response map at each frame. The response
map is generated by aggregating the individual response maps
of the base trackers \( C_i \). Denote \( M_i = \sum_{k=1}^{d} [F_i]_k \ast [I_t]_k \) to be
the response map of base tracker \( C_i \), where \( F_i \) is the filter
for base tracker \( C_i \). Then the MTCF response map \( M \) is
computed as
\[
M = \sum_{i=1}^{L} w_i M_i
\]
where \( w_i \in \mathbb{R} \) is the weight of tracker \( C_i \). We would like to
weight the latest trackers more heavily as object appearances
they model are more likely to be relevant to the current frame.
Although most trackers have \( T \) images, \( C_L \) almost never has \( T \) images (see Section IV-B) and in general is not as reliable
as the other trackers. Taking this into consideration, we set
the weights to be
\[
w_i = \frac{|D_i|((1 - \gamma)^{L-i})}{\sum_{j=1}^{L} |D_j|(1 - \gamma)^{L-j}}
\]
where \( \gamma \in (0, 1) \) is the tracker decay rate that allows more
recent trackers to be more heavily weighted. However, \( \gamma \) must
be set such that the older trackers are not insignificant.

MTCF explicitly models the object’s appearance history
We experimented with such an approach and observed inferior performance. If the training data of the most recent tracker almost never has stable predictions. In this manner, we effectively build the ensemble of initializing with a search region in order to estimate scale change. Given a new image \( I_t \), we apply this prediction at multiple resolutions of \( \alpha \). However, this approach is less robust to temporary changes in appearance. For example, if the tracked object is occluded when \( C_L \) is created, \( C_L \) will model the occluding object and track it with high confidence. We experimented with such an approach and observed inferior performance.

### B. Online Learning and Tracking

The proposed algorithm for online tracking is shown in Lines 1-5 of Algorithm 1. Given a new image \( I_t \), we extract a search region \( I_t \) centered at location \( p_t-1 \in \mathbb{R}^2 \), which is the previously predicted location (in \( x, y \) coordinates). Then the response map \( M \) is computed with Eq. (2) and the new location \( p_t \) is predicted by selecting the argmax. Following \[11\], \[37\], we apply this prediction at multiple resolutions of the search region in order to estimate scale change.

Performing model updates is shown in Lines 6-13 of Algorithm 1. If the training data of the most recent tracker \( C_L \) is at capacity, we create a new tracker \( C_L+1 \). Because \( I_t \) could possibly have an occluded object or be a noisy frame, we initialize \( C_L+1 \) with the \( \tau \) most recent frames instead of initializing with \( I_t \). This leads to the overlaps between \( D_t, D_{t+1}, \forall t \) as seen in Figure 1, which results in more stable predictions. In this manner, we effectively build the ensemble of base trackers in a sequential fashion. Thus, \( C_L \) almost never has \( T \) images, which is why we include \( |D_t| \) in the weight calculation in Eq (3). If we surpass the limit of base trackers, we drop the oldest one (Line 9). Lastly, if \( C_L \) is not at capacity of training data, we simply add \( I_t \) to \( D_L \) and update \( C_L \) accordingly. Note when a tracker \( C_i \) reaches its capacity of training data, it is never updated again; thus, the appearance that \( C_i \) models persists during tracking.

### V. Experiments

In this section, we discuss implementation details, perform detailed studies of the proposed method, and compare with state-of-the-art trackers on multiple datasets. All of our experiments are run on an Intel Core i7 CPU along with an NVIDIA GeForce GTX 1080Ti GPU. Our implementation is written in Python and Tensorflow \[36\].

#### A. Implementation Details

We experiment with a combination of Histogram of Oriented Gradients (HOG) \[38\] and Color Names (CN) \[39\], and also deep convolutional features; we extract conv3-3 features from a VGG16 network \[34\] pre-trained on ImageNet \[40\], and reduce the number of features to 100 with PCA.

For the base tracker sCF, the square search region is set to be \( 5^2 \) the size of the initial target bounding box. \( Y_t \) is set to be a Gaussian density function with standard deviation \( \sqrt{h/w_t}/16 \). When initializing the tracker, we run 100 iterations of L-BFGS with backtracking line search \[35\]. At every 5th frame, we run 5 iterations of L-BFGS to update the model, setting \( \alpha_t = (1 - \eta) \gamma^{-t} \) with \( \eta = 0.013 \) and normalizing \( \alpha_t \) such that \( \sum_{t=1}^{N} \alpha_t = 1 \). Following \[11\], \[37\], we apply the filter at multiple resolutions of the search region in order to jointly predict translation and scale; we use 5 resolutions at step size 1.02.

For the proposed method MTCF, we set the maximum number of images per tracker \( T = 50 \), which is approximately 2 seconds for the datasets we experiment with. We expect this to be a reasonable amount of time for the object appearance to potentially change, and show empirically in Section V-C that this is the case. We set the maximum number of trackers \( K = 8 \), the overlap parameter \( \tau = 5 \), and the tracker decay rate \( \gamma = 0.2 \).

#### B. Datasets and Metrics

We evaluate the proposed method on multiple standard benchmarks for visual object tracking. We first investigate our results on the OTB dataset \[9\], \[1\], which contains 100 videos that are separated into the OTB-2013 dataset \[9\] which includes 51 videos, the OTB-100 dataset \[1\] which includes all 100 videos, and the OTB-50 dataset \[1\] which includes
50 of the more difficult videos for a more in-depth analysis. We evaluate on the one-pass evaluation (OPE) metric on this dataset [9], which computes intersection over union (IoU) of predicted and ground truth bounding boxes for a single run on the dataset. Success plots show the percentage of frames where the IoU is greater than a given threshold. The area under the curve (AUC) is commonly used to rank trackers. For these datasets, we also compare trackers on the success rate of frames where the IoU is larger than 0.5.

We next investigate our results on the VOT2015 [26] and VOT2016 datasets [2], which consists of 60 difficult videos. In this evaluation, trackers are restarted upon failure, where failure denotes the event that a predicted bounding box has no overlap with the ground truth. Accuracy is again measured in IoU, and the robustness metric measures the failure rate of a tracker. The authors propose the expected average overlap (EAO) metric [26] to summarize the performance of each tracker in a single number.

C. Model Analysis

In this section, we perform experiments to study the nature of MTCF. In Table I we show a comparison of sCF and MTCF on multiple feature representations on the OTB datasets. We show AUC and success rate for three feature representations: (i) HOG, (ii) HOG + CN, and (iii) deep features. We also show frames per second (FPS) of the trackers averaged over the OTB100 videos. We see that MTCF improves over sCF in all settings with a relative gain of 4.7% on average, showing the efficacy of having an ensemble trained on different temporal windows. Interestingly, the smallest gains of MTCF are on the deep convolutional feature representation where the relative gain is 1.4% on average. Note that MTCF is slightly slower than sCF as it maintains a collection of up to K base trackers and must compute translation prediction for all of them. Finally, we compare to SRDCF [5], a state-of-the-art correlation filter (using HOG features) equipped with a spatial regularization term to deal with the inappropriately sized filter. The simple base tracker sCF - HOG admits a comparable learning framework with a much simpler learning algorithm and provides similar results at twice the speed.

We test the sensitivity of MTCF to the choice of T and K. We fix TK = 400 and vary K in [2, 4, 8, 16, 20] with T ranging in [200, 100, 50, 25, 20]. We initialize each tracker with τ = T/10 images and use HOG + CN features. In Figure 3, we see a consistent trend where performance increases until K = 8, T = 50, and decreases afterwards. When K is large, the trackers are trained on smaller amounts of data, resulting in unstable trackers and degradation of performance. Note that the optimal choice of T, K depends on the distribution of object appearance variations in the dataset.

D. Comparison to state of the art

We provide thorough comparisons of the proposed tracker on deep features (MTCF-deep) and HOG + CN features (MTCF-HOG+CN). We compare our performance to many recent state-of-the-art trackers and show that MTCF either performs on par with or outperforms them.

I) OTB: We evaluate the proposed tracker MTCF on the OTB datasets [9], [1]. We compare to the trackers provided with the OTB toolkit, along with recently proposed state-of-the-art correlation filters including ECO [13], C-COT [12], MCPF [42], BACF [15], SRDCF [5], and Staple [8]. Furthermore, we include state-of-the-art deep learning based trackers including CREST [41] and SiamFC [27] in the comparison. Note that some of the correlation filter trackers (e.g. ECO, C-COT, MCPF) operate on features extracted from pre-trained deep networks (e.g VGG [34]) on ImageNet [40].

Figure 4 shows OPE AUC results of the trackers on all three OTB datasets. For succinctness, the performance of only the top 10 trackers is shown. Among these top 10 trackers, MTCF performs quite competitively, only being beaten by ECO. Note that while ECO is a state-of-the-art correlation filter, it only learns a single filter which puts it at risk of issues mentioned in Section IV (see Section V-E for an example on the Basketball video). Both versions of MTCF outperform most of the correlation filter based methods. MTCF-deep achieves a relative gain of 4.65%, 6.78%, 12.1%, and 17.8% over MCPF, BACF, SRDCF, and Staple, respectively. While the proposed method falls behind C-COT by a couple AUC points on OTB100, we outperform it by a couple AUC points on OTB50 where the videos are among the more difficult videos for tracking [1]. We also outperform deep learning based trackers CREST and SiamFC by similar margins. Interestingly, the HOG+CN version of MTCF performs only slightly worse than the deep feature version and still outperforms trackers utilizing deep features including MCPF and CREST. MTCF-deep provides a 1.1% relative gain in accuracy over MTCF-HOG+CN on average.
Fig. 5: Qualitative results of MTCF with ECO [13], CREST [41], BACF [15], and SRDCF [5] on videos Basketball, CarScale and DragonBaby on the OTB100 dataset.

<table>
<thead>
<tr>
<th>Tracker</th>
<th>EAO</th>
<th>Acc.</th>
<th>Rob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTCF-d (Ours)</td>
<td>0.322</td>
<td>0.551</td>
<td>0.867</td>
</tr>
<tr>
<td>MTCF-HC (Ours)</td>
<td>0.292</td>
<td>0.544</td>
<td>1.233</td>
</tr>
<tr>
<td>MDNet [25]</td>
<td>0.378</td>
<td>0.599</td>
<td>0.766</td>
</tr>
<tr>
<td>DeepSRDCF [6]</td>
<td>0.318</td>
<td>0.562</td>
<td>1.000</td>
</tr>
<tr>
<td>EBT [43]</td>
<td>0.313</td>
<td>0.453</td>
<td>0.814</td>
</tr>
<tr>
<td>SRDCF [5]</td>
<td>0.288</td>
<td>0.551</td>
<td>1.183</td>
</tr>
<tr>
<td>Struck [44]</td>
<td>0.246</td>
<td>0.460</td>
<td>1.496</td>
</tr>
<tr>
<td>S3Tracker [26]</td>
<td>0.240</td>
<td>0.523</td>
<td>1.667</td>
</tr>
<tr>
<td>DAT [45]</td>
<td>0.224</td>
<td>0.480</td>
<td>1.883</td>
</tr>
<tr>
<td>MEEM [17]</td>
<td>0.221</td>
<td>0.499</td>
<td>1.783</td>
</tr>
</tbody>
</table>

(a) VOT2015

<table>
<thead>
<tr>
<th>Tracker</th>
<th>EAO</th>
<th>Acc.</th>
<th>Rob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTCF-d (Ours)</td>
<td>0.316</td>
<td>0.531</td>
<td>0.933</td>
</tr>
<tr>
<td>MTCF-HC (Ours)</td>
<td>0.279</td>
<td>0.530</td>
<td>1.250</td>
</tr>
<tr>
<td>C-COT [12]</td>
<td>0.331</td>
<td>0.526</td>
<td>0.850</td>
</tr>
<tr>
<td>TCNN [16]</td>
<td>0.325</td>
<td>0.539</td>
<td>0.959</td>
</tr>
<tr>
<td>Staple [8]</td>
<td>0.295</td>
<td>0.538</td>
<td>1.350</td>
</tr>
<tr>
<td>EBT [43]</td>
<td>0.291</td>
<td>0.441</td>
<td>0.900</td>
</tr>
<tr>
<td>MDNet [2]</td>
<td>0.257</td>
<td>0.533</td>
<td>1.204</td>
</tr>
<tr>
<td>SRDCF [5]</td>
<td>0.247</td>
<td>0.523</td>
<td>1.500</td>
</tr>
<tr>
<td>DSST [11]</td>
<td>0.181</td>
<td>0.484</td>
<td>2.517</td>
</tr>
<tr>
<td>Struck [44]</td>
<td>0.142</td>
<td>0.424</td>
<td>3.367</td>
</tr>
</tbody>
</table>

(b) VOT2016

TABLE II: Results on VOT2015 (left) and VOT2016 (right). The winning trackers as described by the VOT reports are highlighted in red. Higher is better for EAO and Accuracy while lower is better for Robustness.

2) VOT: We compare to the trackers provided with the VOT2015 results including MDNet [25], DeepSRDCF [6], EBT [43], SRDCF [5], Struck [44], DAT [45], and MEEM [17] in Table IIa. The winning tracker, MDNet, is the only tracker to outperform MTCF. It utilizes external tracking data to train its convolutional network. Despite not having this, MTCF-deep performs competitively, yielding an EAO score of 0.342, and outperforms the next best tracker, DeepSRDCF, by a relative gain of 7.5%. The HOG+CN version of MTCF also performs competitively, ranking 4th (excluding MTCF-deep) among the trackers with an EAO score of 0.292, outperforming SRDCF, Struck, S3Tracker, DAT, and MEEM.

In Table IIb, we compare the proposed method on the VOT2016 dataset to trackers including C-COT [12], TCNN [16], Staple [8], SRDCF [5], DSST [11], and Struck [44]. MTCF-deep, with an EAO score of 0.316, performs quite competitively with the winning tracker C-COT and second best tracker TCNN. In fact, MTCF-deep yields 2.7% drop in the number of failures compared to TCNN. MTCF-HOG+CN results in an EAO score of 0.279, which outperforms SRDCF, DSST, Struck, and even deep learning methods such as MDNet_N. Note that MDNet_N does not have access to external tracking data for training. MTCF-HOG+CN (and MTCF-deep) is state of the art as defined by the VOT2016 rules [2].

E. Qualitative Results

In Figure 5, we show some qualitative plots of the proposed tracker (with deep features) compared to a few other trackers on a few videos from the OTB100 dataset. As seen in Figure 2 and 5, MTCF is capable of learning the different appearance models on Basketball and reliably tracks the object through the video, where two methods (ECO and BACF) fail towards the end due to an appearance change. MTCF also shows robustness to large scale change on CarScale. On DragonBaby, it performs well visually in comparison with the other state-of-the-art trackers.

VI. Conclusion

We proposed a simple ensemble tracker that maintains multiple base trackers trained on different temporal windows, making it robust to short-term and long-term changes in visual appearance. Our base trackers use a flexible correlation filter formulation in the spatial domain that circumvents known issues addressed in the literature. Extensive experiments on multiple datasets demonstrate that our tracker performs competitively with state-of-the-art methods.

Acknowledgements

This work was funded in part by an NDSEG fellowship, ONR Grant N00014-15-1-2380, NSF Award CCF-1740551, and the program “Learning in Machines and Brains” of CIFAR.