Hedged Deep Tracking

Yuankai Qi[†] Shengping Zhang[†] Lei Qin[♭] Hongxun Yao[†] Qingming Huang[†,♮] Jongwoo Lim[◇] Ming-Hsuan Yang[§]
[†]Harbin Institute of Technology [♭]Institute of Computing Technology, Chinese Academy of Sciences
[♮]University of Chinese Academy of Sciences [◇]Hanyang University [§]University of California at Merced
qykshr@gmail.com s.zhang@hit.edu.cn qinlei@ict.ac.cn h.yao@hit.edu.cn
qmhuang@jdl.ac.cn jlim@hanyang.ac.kr mhyang@ucmerced.edu

Abstract

In recent years, several methods have been developed to utilize hierarchical features learned from a deep convolutional neural network (CNN) for visual tracking. However, as features from a certain CNN layer characterize an object of interest from only one aspect or one level, the performance of such trackers trained with features from one layer (usually the second to last layer) can be further improved. In this paper, we propose a novel CNN based tracking framework, which takes full advantage of features from different CNN layers and uses an adaptive Hedge method to hedge several CNN based trackers into a single stronger one. Extensive experiments on a benchmark dataset of 100 challenging image sequences demonstrate the effectiveness of the proposed algorithm compared to several state-of-the-art trackers.

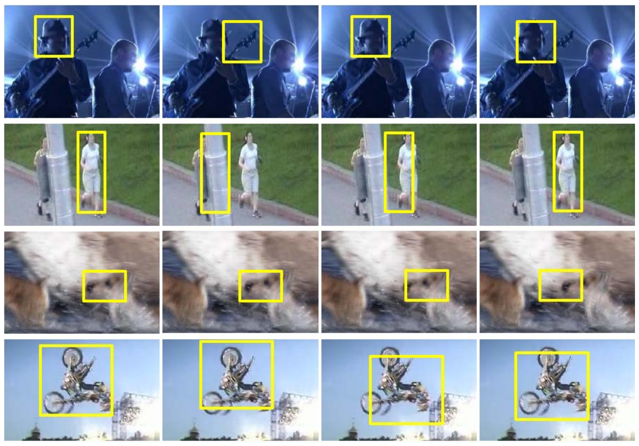


Figure 1. Tracking results of using CNN features from different convolutional layers on a representative frame of four sequences with diverse challenges. The best tracking results are obtained using layers 12, 16, 10, and 10 on four sequences, respectively.

1. Introduction

Visual tracking has become a topic of increasing interest over the past couple of decades due to its importance in numerous applications, such as intelligent video surveillance, vehicle navigation, and human-computer interaction. Despite significant efforts put into developing algorithms [20, 22, 14, 36, 9, 38, 32, 37, 39, 35] and benchmark evaluations [34, 27] for visual tracking, it is still a challenging task due to complicated interfering factors like heavy illumination changes, shape deformation, partial and full occlusion, large scale variations, in-plane and out-of-plane rotations, and fast motion, to name a few.

Most existing tracking approaches focus on either designing effective decision models [13, 12, 16, 40] or extracting robust features [6, 24, 41, 1]. Recently, inspired by the success of deep convolutional neural networks (CNNs) in object recognition and detection [26, 15, 21, 11], several CNN based trackers [30, 18, 9, 25] have been developed. Empirical studies using a large object tracking benchmark show that the performance of CNN based trackers surpasses that of hand-crafted features such as HOG [6], SIFT [24], and color histogram [28, 1].

Despite achieving state-of-the-art performance, existing CNN based trackers still have some limitations. Most of these methods represent target objects only using features from very last layers (*e.g.*, fully-connected layers) of CNNs, which capture rich category-level semantic information, and therefore are useful for object classification. However, features from last layers are not optimal for visual tracking as they do not capture spatial details of the tracked target. These details, captured by first layers, are crucial to visual tracking, as they allow for accurate localization of targets, as shown in the last two rows of Figure 1. On the other hand, as features from first layers are more generic than discriminative as ones from last layers, methods based on features from first layers are likely to fail in challenging scenarios, as shown in the first two rows of Figure 1. To achieve better tracking performances, it is imperative to combine features from different layers to best represent and separate foreground objects from the background clutters.

In this paper, we propose a novel CNN based tracking algorithm, which first builds weak trackers from convolutional layers by applying correlation filters on the layer output, and then hedges all weak trackers into a single stronger one using an online decision-theoretical Hedge algorithm. Specifically, we treat each weak tracker as an expert and compute weights for all experts as their decision confidences. The tracking result in the current frame is the weighted decisions of all experts, which combines advantages of all the considered CNN layers. Since the tracked target moves a small offset between consecutive frames and undergoes appearance variance gradually, an expert that performs well in previous frames has a higher probability to perform well in the current frame. By factoring in historical performance of experts to make decisions, we propose an improved Hedge algorithm to update the weights of all experts, which is more suitable for real-world tracking tasks.

The contributions of this paper are summarized below:

- We propose a novel tracking algorithm that combines weak CNN based trackers from various convolutional layers into a single stronger tracker.
- We develop an improved Hedge algorithm for visual tracking by considering historical performance of weak trackers.
- We carry out extensive experiments on a large-scale benchmark dataset [34] with 100 challenging sequences to demonstrate the effectiveness of the proposed algorithm in comparisons to the state-of-the-art trackers.

2. Related Work

We give a brief review of tracking methods closely related to this work. Comprehensive reviews on visual tracking approaches can be found in [23, 27].

Correlation filters based trackers. Correlation filters are introduced into visual tracking for its computational efficiency [4, 16, 17]. These methods approximate the dense sampling scheme by generating a circulant matrix, of which each row denotes a vectorized sample. As such, its regression model can be computed in the Fourier domain, which brings a large speed improvement in both training and testing stages. Bolme *et al.* [4] develop the Minimum Output Sum of Squared Error (MOSSE) method to learn the filters, and use intensity features for object representation. In [16], Henriques *et al.* propose a tracking method based on correlation filters by introducing kernel methods and employing ridge regression. Subsequently a method that extends the input features from a single channel to multiple channels (*e.g.*, HOG) is presented [17]. Danelljan *et al.* [7] propose an algorithm that searches over scale space for correlation filters to handle large variation in object size. However, all the above mentioned works use only one correlation filter, which limits the power of trackers based on correlation filters. In this work, we exploit the computational efficiency of correlation filters to construct an ensemble tracker where each component tracker is based on features extracted from one convolutional layer of a CNN.

CNN based trackers. Hierarchical features learned from CNNs have been shown to be effective for numerous vision tasks, *e.g.*, classification and recognition [21, 26, 15] in recent years. Numerous methods have since been proposed to exploit CNN features [9, 30, 18] for visual tracking. In [9], Fan *et al.* utilize a pre-trained deep network for human tracking. Wang and Yeung [30] design an autoencoder network to learn representative features for generic objects. Hong *et al.* [18] construct a discriminative model with features from the first fully-connected layer of R-CNN [11] and a generative model with saliency map for visual tracking. While this method is effective for visual tracking, its computational complexity is high. We note that the aforementioned methods do not exploit features from different layers adequately. As shown in Figure 1, features from different layers are effective in different scenarios. Based on these observations, we use an ensemble of multiple CNN based trackers where each one is trained with features from one convolutional layer. We regard each one as a weak expert and hedge them adaptively for visual tracking.

Ensemble trackers. Ensemble approaches have been developed to combine multiple component trackers for visual tracking. Several ensemble tracking methods [2, 31, 12, 3] have been proposed using hand-crafted features. For example, ensemble methods [2, 12, 3] under the boosting framework [10] incrementally train each component weak tracker to classify the training samples that previous trackers misclassified. In [31], Wang and Yeung use a conditional particle filter to infer the target position and the reliability of each component tracker. Different from these works, we consider visual tracking as a decision-theoretic online learning task [5] that infers the tracked target using decisions from multiple expert trackers. That is, in every round each expert makes a decision and the final decision is determined by the weighted decisions of all experts.

3. Algorithmic Overview

As shown in Figure 2, the proposed approach consists of three steps: extracting CNN features, constructing weak trackers, and hedging weak trackers. The pre-trained VGG-Net [26] is used to extract feature maps of convolutional layers from image regions, which represent the tracked target at different resolutions and semantic levels. Each feature map is then convolved by correlation filters to generate response maps, from which a weak tracker is constructed with

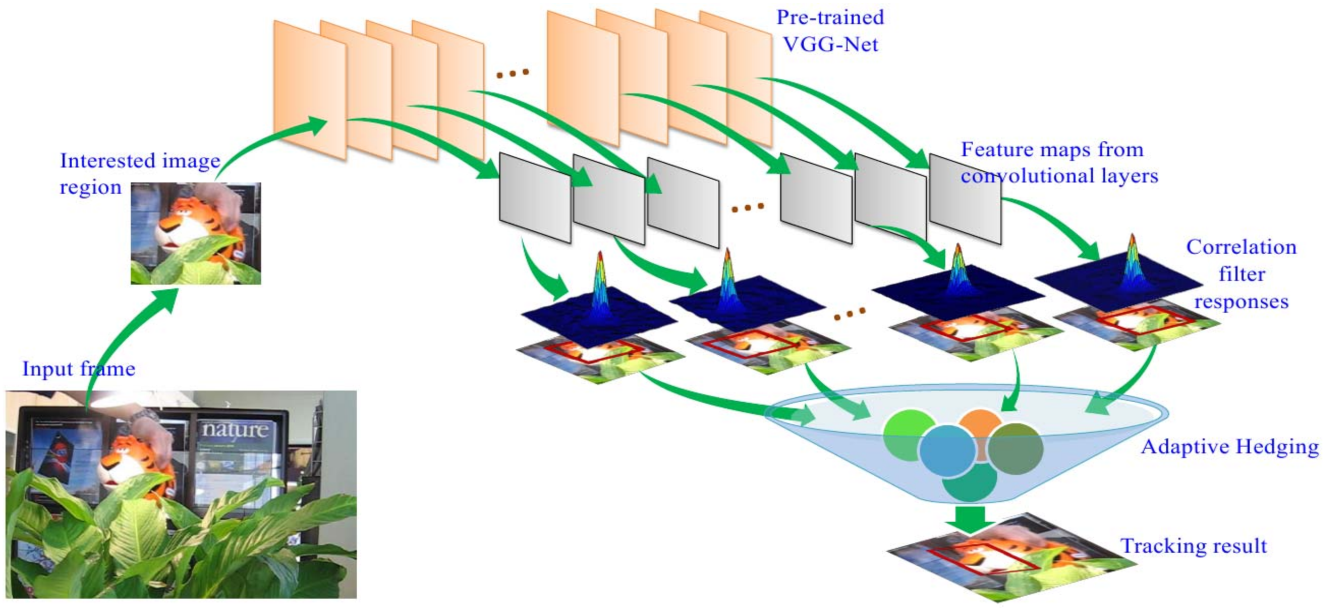


Figure 2. Main steps of the proposed algorithm. The proposed algorithm consists of three components: 1) extracting CNN features from different convolutional layers using the pre-trained VGG-Net (Section 4.1); 2) constructing weak trackers using correlation filters where each one is trained with CNN features from one layer (Section 4.2); 3) hedging weak trackers into a stronger one using an improved Hedge algorithm (Section 4.3).

moderate performance. All weak trackers are finally hedged into a stronger one using the proposed adaptive Hedge algorithm for visual tracking, which exploits the strength of all CNN layers.

4. Proposed Algorithm

In this section, we first present the technical details of the proposed algorithm and then describe the online update scheme.

4.1. Deep CNN features

CNN models, such as AlexNet [21], R-CNN [11], CaffeNet [19], and VGG-Net [26], have been developed for large-scale image classification and object recognition tasks. The proposed method is based on the VGG-Net, as it has a much deeper architecture (up to 19 weight layers) and hence can provide much richer features compared to most CNNs which usually have 5 or 7 layers. The VGG-Net is trained with 1.3 million images of the ImageNet dataset and achieves the state-of-the-art results on classification challenges [26].

Different from classification tasks which only require the extracted features to capture more category-level semantic information, visual tracking also requires the extracted features to have precise localization ability since a small drift from the tracked target to its surrounding background causes gradual degradation in tracking performance and eventual failure. The deep VGG-Net facilitates features extracted from different layers to describe target objects with greater details. However, tracking methods using CNN features from any layer alone are less effective (see Figure 1 for example of tracking failures).

4.2. Weak CNN based trackers

In this work, a module that makes use of correlation filters on CNN features extracted from one layer is used to build a weak tracker. Trackers based on correlation filters [4, 7, 17, 16] exploit the circulant structure of training and testing samples to greatly accelerate the training and testing processes with negligible precision loss. Let $X^k \in \mathbb{R}^{P \times Q \times D}$ denote the feature map extracted from the k-th convolutional layer and $Y \in \mathbb{R}^{P \times Q}$ be the gaussian shape label matrix, which is subject to a 2D Gaussian distribution with zero mean and standard deviation proportional to the target size. Let $\mathcal{X}^k = \mathcal{F}(X^k)$, $\mathcal{Y} = \mathcal{F}(Y)$, where $\mathcal{F}(\cdot)$ denotes the discrete Fourier transformation (DFT). The k-th filter can be modeled in the Fourier domain by

$$\mathcal{W}^k = \arg\min_{\mathcal{W}} \|\mathcal{Y} - \mathcal{X}^k \cdot \mathcal{W}\|_F^2 + \lambda \|\mathcal{W}\|_F^2, \qquad (1)$$

where

$$\mathcal{X}^k \cdot \mathcal{W} = \sum_{d=1}^{D} \mathcal{X}^k_{*,*,d} \odot \mathcal{W}_{*,*,d}, \qquad (2)$$

and the symbol $\odot$ denotes the element-wise product.

The optimization problem in (1) has a simple closed form solution, which can be efficiently computed in the Fourier domain by

$$\mathcal{W}^k_{*,*,d} = \frac{\mathcal{Y}}{\mathcal{X}^k \cdot \mathcal{X}^k + \lambda} \odot \mathcal{X}^k_{*,*,d}. \tag{3}$$

Given the testing data T^k from the output of the k-th layer, we first transform it to the Fourier domain $\mathcal{T}^k = \mathcal{F}(T^k)$, and then the responses can be computed by

$$S^k = \mathcal{F}^{-1}(\mathcal{T}^k \cdot \mathcal{W}^k), \tag{4}$$

where $\mathcal{F}^{-1}$ denotes the inverse of DFT.

The k-th weak tracker outputs the target position with the largest response

$$(x^k, y^k) = \arg\max_{x',y'} S^k(x', y'). \tag{5}$$

4.3. Hedging CNN based trackers

The standard parameter-free Hedge algorithm [5] is proposed to tackle decision-theoretic online learning problems in a multi-expert multi-round setting. Given the initial confidence weights of all experts, in the current round, a final decision is made based on the weighted decisions of all experts. The weights of all the experts are then updated to reflect each expert's decision loss. In the visual tracking scenario, it is natural to treat each CNN based tracker as an expert and then predict the target position in the t-th frame by

$$(x^*_t, y^*_t) = \sum\nolimits_{k=1}^{K} w^k_t \cdot (x^k_t, y^k_t), \tag{6}$$

where w^k_t is the weight of expert k and $\sum_{k=1}^{K} w^k_t = 1$. Once the ultimate target position is predicted, each expert will incur a loss.

The loss of expert k at frame t is computed as

$$\ell^k_t = \max(S^k_t) - S^k_t(x^*_t, y^*_t), \tag{7}$$

where $\max(\cdot)$ operates on a matrix and returns the largest element of the matrix and $S(x, y)$ denotes the element at position (x, y) of matrix S. The standard parameter-free Hedge algorithm generates a new weight distribution on all experts by introducing a *regret* measure defined by

$$r^k_t = \bar{\ell}^k_t - \ell^k_t, \tag{8}$$

where the weighted average loss among all experts is computed as $\bar{\ell}^k_t = \sum_{k=1}^{K} w^k_t \ell^k_t$.

By minimizing the cumulative regret

$$R^k_t = \sum\nolimits_{\tau=1}^{t} r^k_\tau, \tag{9}$$

to any expert k, for any round of t, the new weights $w^1_{t+1}, \cdots, w^K_{t+1}$ are generated.

Although the standard parameter-free Hedge algorithm performs well in the simulated one-dimension tracking experiment, where the target stays stationary or moves in a constant velocity [5], it is less effective for the real-world tracking tasks since it does not consider two crucial factors: (i) The target appearance usually changes at irregular pace (sometimes gradually and sometimes rapidly). This means that the proportion of the historic regret R^k_{t-1} should vary with time t to better reflect the current state for visual tracking. (ii) Since each expert captures a different aspect of the target, it is not effective to fix the ratio of the cumulative regret for all the experts. To address these issues, we propose an adaptive Hedge algorithm, which considers the difference of historic regrets over time t and expert k simultaneously.

As the object appearance usually does not change significantly at least in a short time period, we model the loss of each expert ℓ^k during the time period Δt via a Gaussian distribution with mean μ^k_t and standard variance σ^k_t

$$\mu^k_t = \frac{1}{\Delta t} \sum_{\tau=t-\Delta t+1}^{t} \ell^k_\tau, \tag{10}$$

$$\sigma^k_t = \sqrt{\frac{1}{\Delta t - 1} \sum_{\tau=t-\Delta t+1}^{t} (\ell^k_\tau - \mu^k_t)^2}. \tag{11}$$

We then measure the stability of expert k at time t using

$$s^k_t = \frac{|\ell^k_t - \mu^k_t|}{\sigma^k_t}. \tag{12}$$

A smaller s^k_t indicates that this expert tends to be more stable than the one with a larger s^k_t. Therefore, we prefer a larger proportion on its current regret. In contrast, a larger s^k_t means this expert varies greatly, and therefore we compute its cumulative regret mainly depending on its historic information. Based on this principle, we obtain the following adaptive cumulative regret

$$R^k_t = (1 - \alpha^k_t) R^k_{t-1} + \alpha^k_t r^k_t, \tag{13}$$

$$\alpha^k_t = \min\left(g, \exp\left(-\gamma s^k_t\right)\right), \tag{14}$$

where γ is a scale factor and g defines a maximum ratio on the current regret to avoid that no historic information is considered. We validate the effectiveness of the proposed adaptive Hedge compared to the original one in Section 5.4.

Since our adaptive Hedge algorithm adheres to the framework of the standard one, the solution to minimize the cumulative regret (13) has the same form,

$$w^k_{t+1} \propto \frac{[R^k_t]_+}{c_t} \exp \frac{([R^k_t]_+)^2}{2c_t}, \tag{15}$$

where $[R^k_t]_+$ denotes $\max\{0, R^k_t\}$, and c_t servers as a scale parameter like in [5], which is determined by solving $\frac{1}{K}\sum_{k=1}^{K} \exp(\frac{([R^k_t]_+)^2}{2c_t}) = e$.

Algorithm 1: Hedged deep tracking

```
1  Input: initial weights $w_1^1, \cdots, w_1^K$; target position $(x_1, y_1)$
   in the 1st frame; VGG-Net19; $R_1^k = 0, \ell_1^k = 0$;
2  Crop interested image region;
3  Initiate $K$ weak experts using (3);
4  for $t = 2, 3, \cdots$ do
5      Exploit the VGG-Net19 to obtain $K$ representations;
6      Compute correlation filter responses using (4);
7      Find target position predicted by each expert using (5);
8      if $t \neq 2$ then
9          Compute ultimate position using (6);
10     else
11         Set ultimate position with ground truth;
12     end
13     Compute experts' losses using (7);
14     Update stability models using (10) and (11);
15     Measure each expert's stability using (12);
16     Compute adaptive proportion of historic regret for each
       expert using (14);
17     Update cumulative regret of each expert using (13);
18     Update weights for each expert using (15) and
       normalize them to have a sum of 1;
19 end
```

4.4. Model update

Since the feature maps of VGG-Net have up to 512 channels, retraining the ridge regression models with the newly collected samples is impractical, especially when the amount of the training data becomes extremely large over time. In practice, we adopt an incremental update similar to that in [7], which only uses new samples $\bar{\mathcal{X}}^k$ in the current frame to partially update the previous models,

$$\mathcal{Z}^k_{*,*,d} = \frac{\mathcal{Y}}{\bar{\mathcal{X}}^k \cdot \bar{\mathcal{X}}^k + \lambda} \odot \bar{\mathcal{X}}^k_{*,*,d}, \quad (16)$$

$$\mathcal{W}^k_t = (1-\eta)\mathcal{W}^k_{t-1} + \eta \mathcal{Z}^k_t. \quad (17)$$

Algorithm 1 summarizes the main steps of the proposed tracking method.

5. Experimental Results

In this section, we present extensive experimental evaluations on the proposed hedged deep tracker (HDT). We first discuss the implementation details and the evaluation protocol. We then present two sets of experimental evaluations: one compared to several state-of-the-art trackers and the other one to several baseline trackers including component weak trackers and the hedged strong tracker using the standard parameter-free Hedge method [5].

5.1. Implementation details

For feature extraction, we crop an image patch with 2.2 times the size of the target bounding box and then resize it

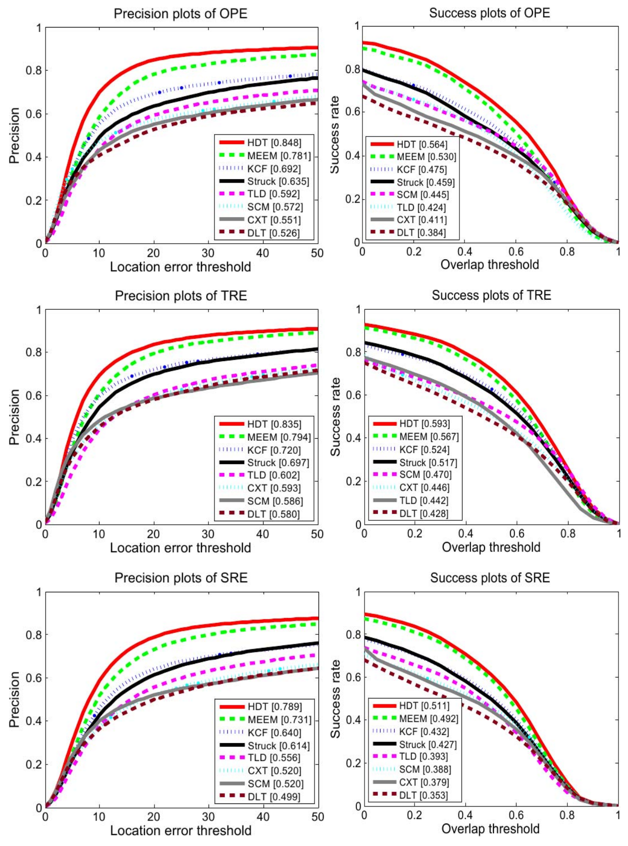


Figure 3. Evaluation results on 100 sequences.

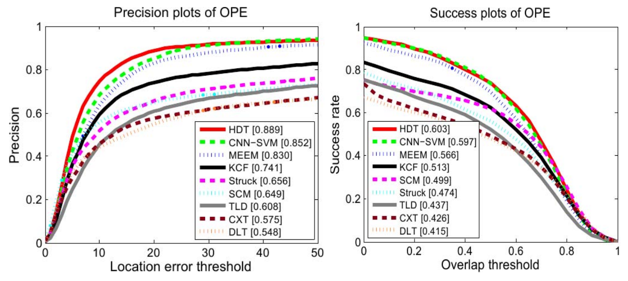


Figure 4. Evaluation results on 50 sequences.

to 224×224 pixels for the VGG-Net with 19 layers (16 convolutional layers and 3 fully-connected layers). After the forward propagation, we use the outputs from six convolutional layers (10th∼12th, 14th∼16th) as six types of feature maps and all feature maps are resized to the same size. This setting simultaneously takes the feature diversities and the computational cost into consideration. Since VGG-Net adopts very small convolutional filters (3×3 pixel size), the feature maps from first layers (*i.e.*, less than 10) have limited representation power (see Section 5.4). We implement our algorithm in MATLAB, and utilize the MatConvNet toolbox [29] in this work. Our implementation runs at 10 frames per second on a computer with an Intel I7-4790K

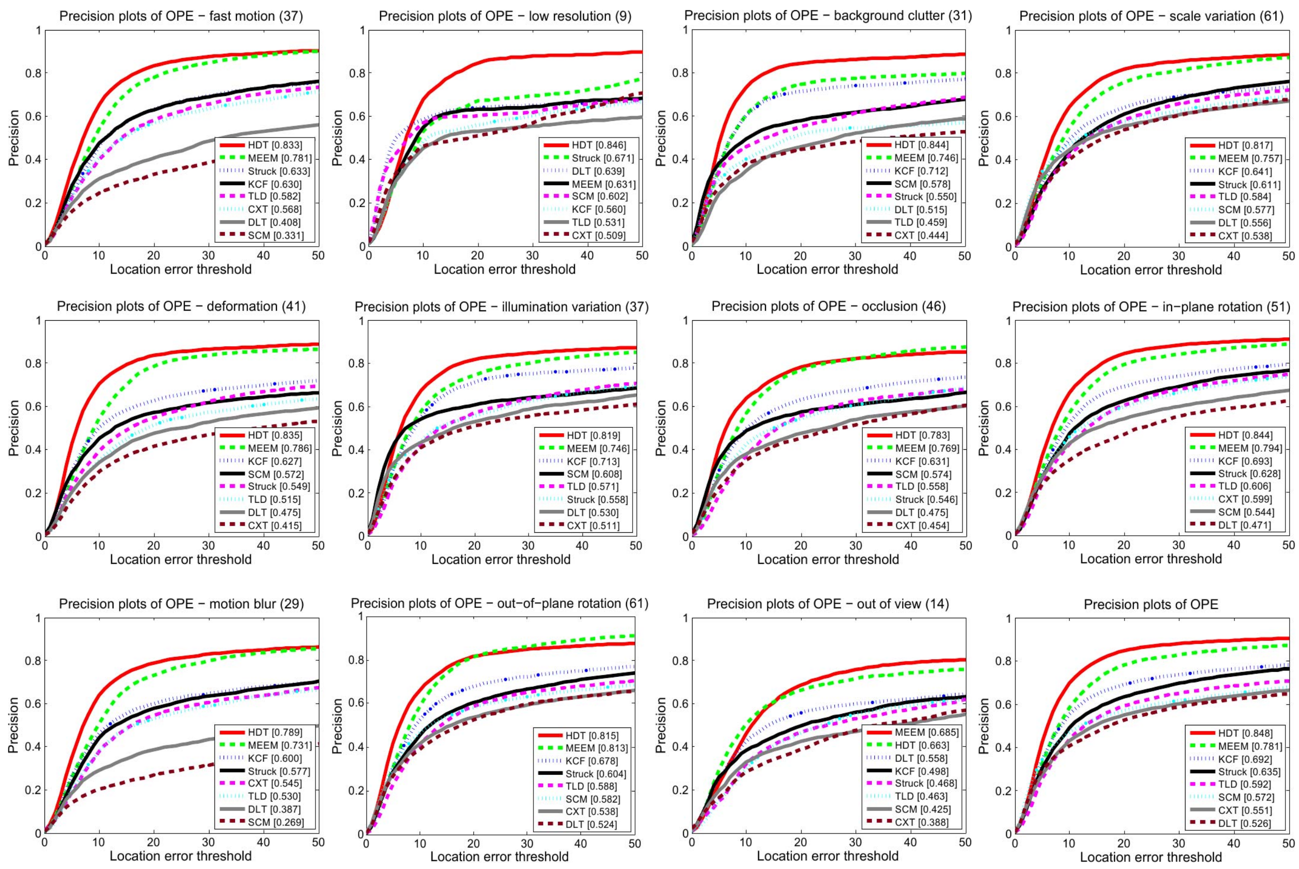


Figure 5. Attribute-based evaluation on 100 sequences. We also put the overall performance here (the last one) for comparison convenience facing a single challenge and their combination.

4.00 GHz CPU, 16GB RAM, and a GeForce GTX780Ti GPU card which is only used to compute the CNN features. We make MATLAB code available to the public (`http://faculty.ucmerced.edu/mhyang/pubs.html`).

All the following experiments are carried out using the following fixed parameters: the tradeoff parameter in (1) is set to $\lambda = 10^{-4}$; the time window in (10) is set to $\Delta t = 5$; the truncate threshold in (14) is set to $g = 0.97$; the learning rate in (17) is set to $\eta = 0.01$; and the initial weights of the six weak experts are empirically set to (1, 0.2, 0.2, 0.02, 0.03, 0.01).

5.2. Evaluation protocol

To fully assess our method, we use one-pass evaluation (OPE), temporal robustness evaluation (TRE), and spatial robustness evaluation (SRE) metrics on a large object tracking benchmark dataset [34] which contains 100 image sequences. These sequences involve 11 tracking challenges, such as illumination changes, camera shake, scale variation, pose variation, partial or full occlusion, and rotation, to name a few. Experimental results are reported using overlap success plots and center location error plots. The compared trackers are ranked in terms of area under the curve and distance precision at a threshold of 20 pixels, respectively. For completeness, we also report the results on the benchmark [33], which is a subset of [34]. More results and videos are presented in the supplementary material.

5.3. Comparisons to state-of-the-art trackers

We compare our algorithm to 8 recent state-of-the-art trackers: DLT [30], CNN-SVM [18], KCF [17], MEEM [36], Struck [14], CXT [8], TLD [20], and SCM [41]. DLT and CNN-SVM are based on deep learning; KCF is one of the best correlation filters based trackers; and the remaining trackers rank top 5 on the benchmark [34].

Quantitative evaluation. Figure 3 shows the OPE, TRE, and SRE results on 100 image sequences. It should be noted that the results of CNN-SVM are not included for fair comparisons as the source code is not available. We also provide a comparison on 50 sequences in Figure 4 with OPE results for CNN-SVM taken from [18]. Figure 3 and Figure 4 show that our HDT performs favorably against the state-of-the-art methods on all the three evaluation metrics. We note that HDT performs better in terms of tracking precision (than success rate), which indicates that HDT is able

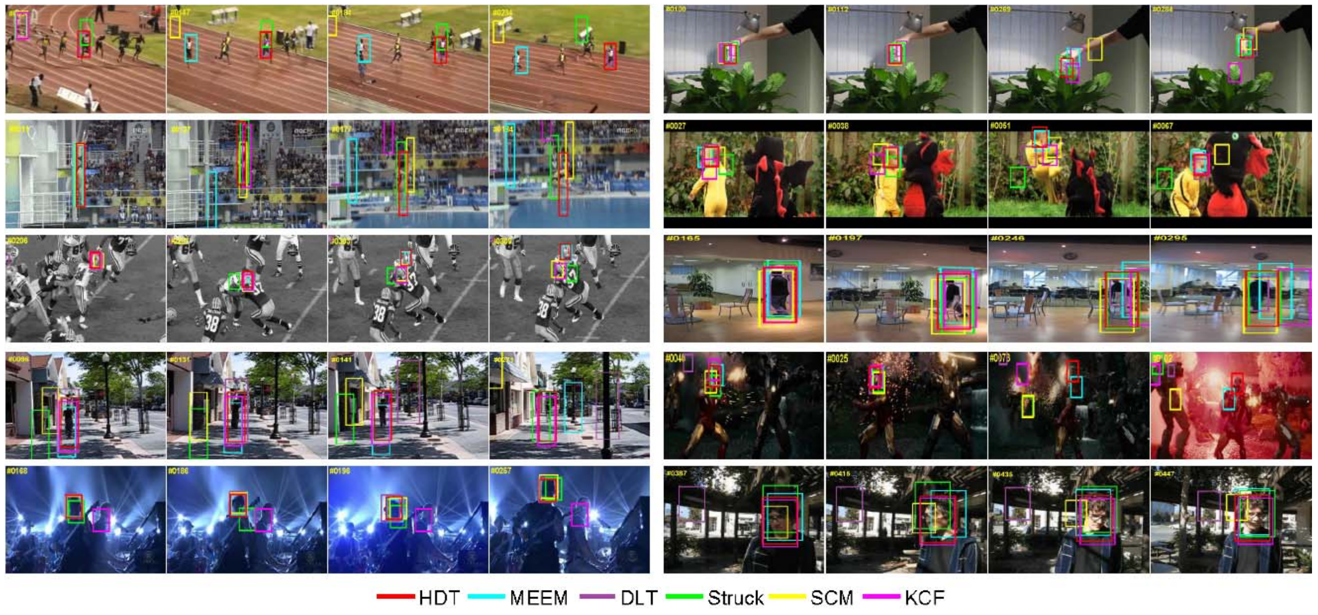
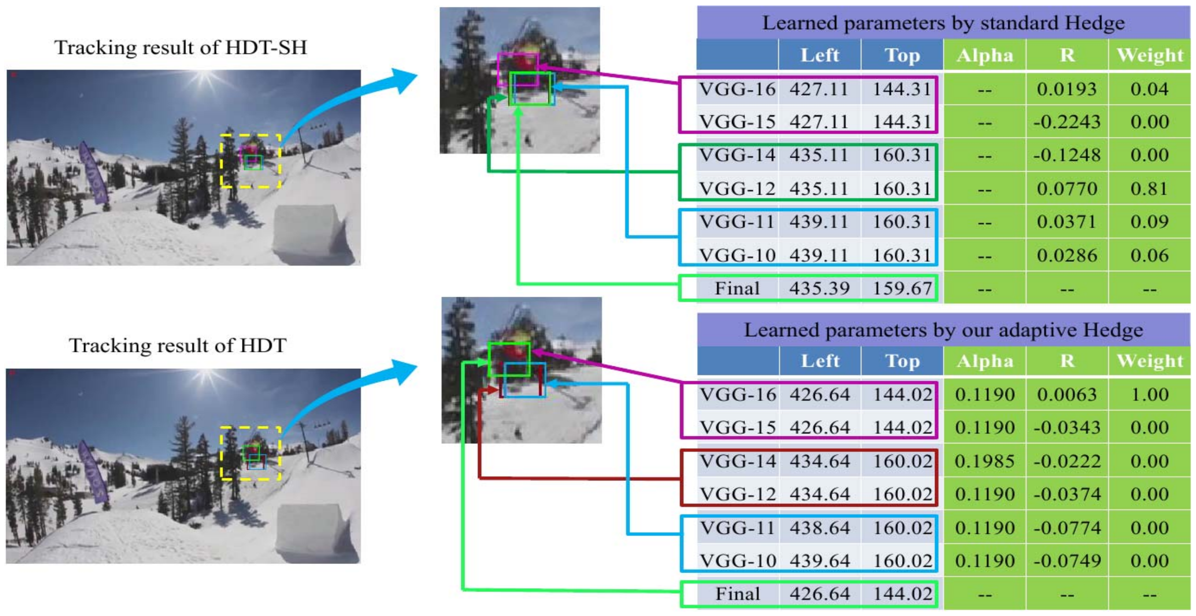


Figure 6. Bounding box comparison on several challenging image sequences (from left to right and top to down are *bolt2, coke, diving, dragonBaby, football, human2, human9, ironman, shaking, and trellis*, respectively).



Learned parameters by standard Hedge

	Left	Top	Alpha	R	Weight
VGG-16	427.11	144.31	--	0.0193	0.04
VGG-15	427.11	144.31	--	-0.2243	0.00
VGG-14	435.11	160.31	--	-0.1248	0.00
VGG-12	435.11	160.31	--	0.0770	0.81
VGG-11	439.11	160.31	--	0.0371	0.09
VGG-10	439.11	160.31	--	0.0286	0.06
Final	435.39	159.67	--	--	--

Learned parameters by our adaptive Hedge

	Left	Top	Alpha	R	Weight
VGG-16	426.64	144.02	0.1190	0.0063	1.00
VGG-15	426.64	144.02	0.1190	-0.0343	0.00
VGG-14	434.64	160.02	0.1985	-0.0222	0.00
VGG-12	434.64	160.02	0.1190	-0.0374	0.00
VGG-11	438.64	160.02	0.1190	-0.0774	0.00
VGG-10	439.64	160.02	0.1190	-0.0749	0.00
Final	426.64	144.02	--	--	--

Figure 7. Tracking results on the 12-th frame of the *skiing* sequence. We illustrating how the weights are assigned to the CNN based trackers by the proposed adaptive and the standard parameter-free Hedge methods.

to track target objects well but gives a less accurate bounding box since, for computational efficiency, HDT does not search over scales to determine the best one.

Attribute-based evaluation. To thoroughly evaluate the robustness of the proposed HDT in various scenes, we present tracking performance in terms of each tracking challenge on 100 image sequences in Figure 5. As illustrated in Figure 5, our algorithm performs well against other methods in almost all tracking challenges. In particular, HDT outperforms other methods by a huge margin in handling low resolution, which can be attributed to CNN features with rich spatial details from first layers and features with semantics from last layers. In contrast, DLT only takes advantage of last layers' features, and hence its performance is suffered. We also observed that HDT does not perform well in handling out-of-view challenge, as HDT does not search for the target in a whole frame in order to reduce the

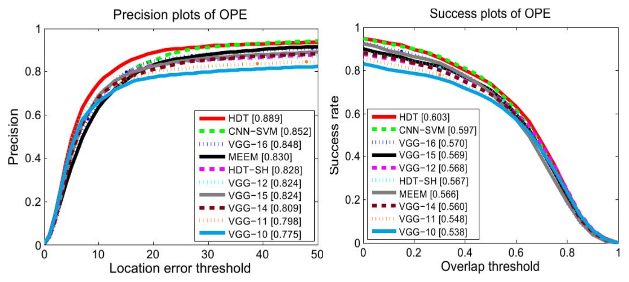


Figure 8. Comparison among our HDT and several baselines: all its constituent CNN based trackers and the one combined by standard parameter-free Hedge. For completeness, we also include two state-of-the-art methods, CNN-SVM and MEEM, in the plots.

computational load. Therefore, HDT may lose the target, even if it reappears somewhere else.

Qualitative evaluation. We present several tracking results from the evaluated methods in Figure 6. For presentation clarity, only results from the top six performing methods are shown. Overall, our tracker is able to localize the targets more precisely. However, almost all the other methods are unable to handle these complicated scenarios. The MEEM tracker performs well in presence of illumination variations, occlusion, and in-plane-rotation (*shaking, coke,* and *trellis*), as MEEM simultaneously maintains several target snapshots from different times. However, it tends to fail when similar objects appear, such as in sequence *bolt2* or *football*, since the features are not discriminative enough. When the background is cluttered, as in sequences *diving* or *ironman*, most of the compared methods are apt to lose the target. Although DLT adopts a deep autoencoder network, it usually fails on these challenging sequences. This is because its deep network has no shared weights and it is trained with small amount of data. Since our HDT hedges several weak CNN based trackers that perform well in different environments, it can overcome these challenges much better than other trackers. In addition, we note that in the *diving, human9,* and *trellis* sequences, even though HDT tracks targets accurately, some of its bounding boxes are not tight to the target since it does not search for the best scale as previously discussed.

5.4. Comparisons to baseline trackers

To evaluate the effectiveness of the proposed adaptive Hedge method, we compare the HDT against its component CNN based trackers denoted by VGG-10, VGG-11, VGG-12, VGG-14, VGG-15, and VGG-16, as well as the hedged CNN based tracker using the standard parameter-free Hedge [5], denoted by HDT-SH, on the benchmark [33].

Figure 8 shows the tracking results. When features from each convolutional layer are used solely for tracking, the performance generally increases as the depth of the layer is increased. But even the best component CNN based tracker VGG-16 still does not perform as good as CNN-SVM. This is because CNN-SVM takes advantages of both a R-CNN feature based discriminative model (features of fc_6 being used) and a back-project saliency map based generative model. Note that the performance of VGG-10 is far behind that of MEEM which is based on hand-crafted features. This explains why we train weak trackers only using convolutional features from the layers after the 10th layer. When combining these six component CNN based trackers using the standard Hedge, the tracking performance is below the best performed component tracker. In contrast, the proposed HDT achieves the best results, which demonstrates the effectiveness of the adaptive Hedge method.

To further explore the difference between the proposed adaptive and the standard parameter-free Hedge methods, we present a comparison of them on a typical frame at running time in Figure 7. Figure 7 shows that the proposed adaptive Hedge allocates more desirable weights to weak CNN based trackers than the standard one. The reason mainly lies in the computation of the accumulative regret R. The standard Hedge uses a fixed proportion of historical information R_{t-1} for all weak trackers at any time t. In contrast, we adaptively compute the proportion of historical information R_{t-1}, *i.e.*, we introduce a dynamic parameter α in (13) and model the α with a Gaussian distribution in a time window Δt. As demonstrated in Figure 8, the hedged tracker using the adaptive scheme performs better.

6. Conclusion

In this paper, we propose a novel CNN based tracking framework which uses an adaptive online decision learning algorithm to hedge weak trackers, obtained by correlation filters on CNN feature maps, into a stronger one to achieve better results. To the best of our knowledge, the proposed algorithm is the first to adaptively hedge features from different CNN layers in an online manner for visual tracking. Extensive experimental evaluations on a large-scale benchmark dataset demonstrate the effectiveness of the proposed hedged deep tracking algorithm.

7. Acknowledgments

This work was supported in part by National Basic Research Program of China (973 Program) 2015CB351802 and 2012CB316400, in part by National Natural Science Foundation of China 61332016, 61025011, 61133003, 61472103, 61390510, 61572465, and 61300111. J. Lim is supported partly by R&D programs by NRF (2014R1A1A2058501) and MSIP/NIPA (H8601-15-1005). M.-H. Yang is supported partly by the National Science Foundation CAREER Grant 1149783 and IIS Grant 1152576, and a gift from Adobe.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006. 1

[2] S. Avidan. Ensemble tracking. *TPAMI*, 29(2):261–271, 2007. 2

[3] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier. Randomized ensemble tracking. In *ICCV*, 2013. 2

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2, 3

[5] K. Chaudhuri, Y. Freund, and D. Hsu. A parameter-free hedging algorithm. In *NIPS*, 2009. 2, 4, 5, 8

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1

[7] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 2, 3, 5

[8] T. B. Dinh, N. Vo, and G. G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011. 6

[9] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *TNN*, 21(10):1610–1623, 2010. 1, 2

[10] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, 1995. 2

[11] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 3

[12] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006. 1, 2

[13] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008. 1

[14] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 1, 6

[15] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 2

[16] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012. 1, 2, 3

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015. 2, 3, 6

[18] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015. 1, 2, 6

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014. 3

[20] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, 2010. 1, 6

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3

[22] G. Li, L. Qin, Q. Huang, J. Pang, and S. Jiang. Treat samples differently: Object tracking with semi-supervised online covboost. In *ICCV*, 2011. 1

[23] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. van den Hengel. A survey of appearance models in visual object tracking. *ACM TIST*, 4(4):58, 2013. 2

[24] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 1

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

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 2, 3

[27] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *TPAMI*, 36(7):1442–1468, 2014. 1, 2

[28] G. Tian, R. Hu, Z. Wang, and Y. Fu. Improved object tracking algorithm based on new HSV color probability model. In *ISNN*, 2009. 1

[29] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for MATLAB. In *ACM Multimedia*, 2015. 5

[30] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013. 1, 2, 6

[31] N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time Series data. In *ICML*, 2014. 2

[32] L. Wen, D. Du, Z. Lei, S. Li, and M.-H. Yang. Jots: Joint online tracking and segmentation. In *CVPR*, 2015. 1

[33] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 6, 8

[34] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *TPAMI*, 37:1834–1848, 2015. 1, 2, 6

[35] B. Zhang, A. Perina, Z. Li, V. Murino, J. Liu, and R. Ji. Bounding multiple gaussians uncertanity with application to object tracking. *IJCV*, pages 1–16, 2016. 1

[36] J. Zhang, S. Ma, and S. Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014. 1, 6

[37] S. Zhang, S. Kasiviswanathan, P. C. Yuen, and M. Harandi. Online dictionary learning on symmetric positive definite manifolds with vision applications. In *AAAI*, 2015. 1

[38] S. Zhang, H. Yao, X. Sun, and X. Lu. Sparse coding based visual tracking: Review and experimental comparison. *Pattern Recognition*, 46:1772–1788, 2013. 1

[39] S. Zhang, H. Zhou, F. Jiang, and X. Li. Robust visual tracking using structurally random projection and weighted least squares. *IEEE Trans. Circuits Syst. Video Techn.*, 25:1749–1760, 2015. 1

[40] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, 2012. 1

[41] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012. 1, 6