Robust Visual Tracking Using Oblique Random Forests

Le Zhang[1], Jagannadan Varadarajan[1], Ponnuthurai Nagaratnam Suganthan[2],
Narendra Ahuja[1,3] and Pierre Moulin[1,3]
[1] *Advanced Digital Sciences Center, Singapore*, [2] *Nanyang Technology University, Singapore*
[3] *University of Illinois at Urbana-Champaign, IL USA*,
{zhang.le,vjagan}@adsc.com.sg, epnsugan@ntu.edu.sg, {moulin.ifp, n-ahuja}@illinois.edu

Abstract

Random forest has emerged as a powerful classification technique with promising results in various vision tasks including image classification, pose estimation and object detection. However, current techniques have shown little improvements in visual tracking as they mostly rely on piece wise orthogonal hyperplanes to create decision nodes and lack a robust incremental learning mechanism that is much needed for online tracking. In this paper, we propose a discriminative tracker based on a novel incremental oblique random forest. Unlike conventional orthogonal decision trees that use a single feature and heuristic measures to obtain a split at each node, we propose to use a more powerful proximal SVM to obtain oblique hyperplanes to capture the geometric structure of the data better. The resulting decision surface is not restricted to be axis aligned, and hence has the ability to represent and classify the input data better. Furthermore, in order to generalize to online tracking scenarios, we derive incremental update steps that enable the hyperplanes in each node to be updated recursively, efficiently and in a closed-form fashion. We demonstrate the effectiveness of our method using two large scale benchmark datasets (OTB-51 and OTB-100) and show that our method gives competitive results on several challenging cases by relying on simple HOG features as well as in combination with more sophisticated deep neural network based models. The implementations of the proposed random forest are available at https://github.com/ZhangLeUestc/Incremental-Oblique-Random-Forest.

1. Introduction

Visual tracking can be viewed as a method to estimate the coordinates of an object (*e.g.*, object bounding box in the image plane) consistently through a sequence of frames. Due to its far-reaching applications in domains including surveillance, human computer interaction, autonomous driving, and health care, tracking is a core problem in computer vision.

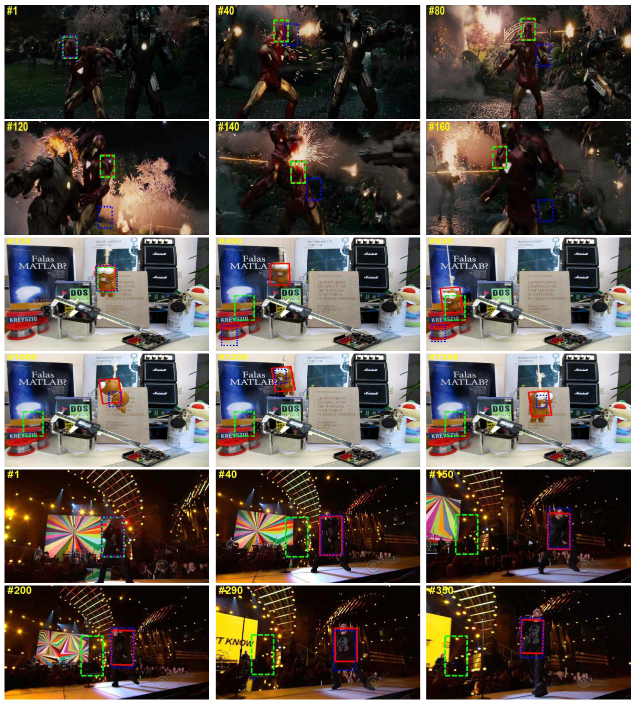


Figure 1. Results obtained from the proposed Obli-RaF on challenging scenarios from OTB-100 dataset. First – second row: the target undergoes severe motion problems, third – fourth row: Occlusion, fifth – sixth row: illumination variations. Our method (indicated by red bounding box) achieves more precise results when compared to HCF [32] (in green) and FCNT [47] (in blue). Best viewed in color and when zoomed.

In this paper, we study the problem of *'single-object model-free tracking'* which is widely addressed in the tracking literature. Here, the tracker is initialized with a bounding box of an arbitrary object of interest in the first frame. Given this *single* (labeled) instance, the goal is to predict the location of the object in an online manner in a *model-free* setting, *i.e.*, without using any explicit appearance or shape model. Whereas visual tracking in constrained settings with minimal occlusion may be relatively straightforward, single object tracking in an unconstrained, model-free setting involves several challenges (see Fig. 1) due to illumination and background variations, occlusions, unpredictable motion, motion blur, appearance changes owing to object deformations, and object drift.

Tracking methods can be classified as *generative*, *discriminative*, or *hybrid*. Generative trackers build an object appearance model based on some generative process and search for regions most similar to the target model without accounting for any information available from the background (*e.g.*, [6, 13, 1, 14, 31, 30, 62]). Discriminative trackers [3, 21, 4, 27, 53] address the problem by learning a classifier to distinguish the target from the background. Here, the classifier is typically evaluated exhaustively at many locations to detect the target in subsequent frames and updated online based on tracking results from each frame. Recent advances in modern visual tracking systems have seen the widespread adoption of discriminative trackers due to their ability to distinguish the target object from background and other distractors.

Recently, random forest (RaF) has emerged as a powerful classification method [20] with promising results in several computer vision applications [28] [12] [18] [43] [19]. Several interesting properties of RaFs including their efficiency in both training and classification, scalability, and robustness towards class imbalance make them potentially attractive for tracking applications too. However, conventional RaF methods have shortcomings as they only select one feature to conduct a split at each node based on some heuristic impurity measurement such as information gain or Gini-index [10], which results in a piecewise orthogonal hyperplane with a poor fit of data. This process ignores the geometric shape of the feature space but focuses only on improving the distribution of different classes (*impurity score*) on each side of the hyperplane [35].

To tackle the above problem, we propose a novel Oblique Random Forest (Obli-RaF) algorithm. Our approach differs from conventional RaF in many ways. Firstly, our Obli-RaF uses more than one feature to conduct a split resulting in an oblique hyperplane at each decision node. This is shown to be more accurate and efficient than the orthogonal RaF [59, 37, 60]. Fig. 2 illustrates the difference between an orthogonal and oblique decision tree using a toy dataset. Note that oblique decision tree with its oblique hyperplane captures the geometrical structure of the data better (red hyperplane in Fig. 2) than the piece-wise axis orthogonal hyperplane (in green). However, no attempts have been made in exploring its use in online scenarios so far. Secondly, in order to avoid the expensive feature selection process at each node, we propose to use *proximal support vector machines* (PSVM) [34] - a supervised clustering step to learn

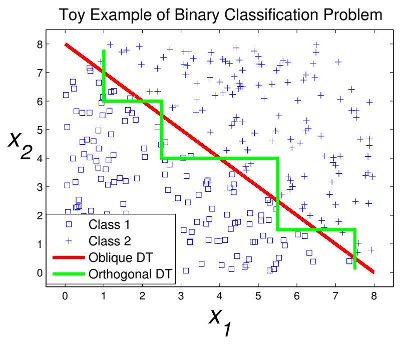


Figure 2. A toy example of classification boundary generated by orthogonal and oblique decision trees. Orthogonal RaF selects a single feature at each node to conduct a split. This results in an piece-wise axis orthogonal hyperplane (in green color). Oblique RaF, on the other hand, uses more than one feature at each node and thus results in an oblique hyperplane (in red color) that classifies the data better.

the hyperplanes at each decision node. This decision tree induction mechanism takes into account both the geometric structure of the feature space and the distance of the data samples to the hyperplane. Furthermore, we propose a closed form solution to recursively update the oblique hyperplanes at each node to adapt to appearance variations of the target object. The key contributions of this paper are summarized as follows:

- We propose a novel oblique random forest which can better capture the geometric structure of the data within each node of the decision tree without explicitly searching for a "good" candidate feature. Our method results in decision trees that are, on average, 5.89 times shorter and 3 times faster than the orthogonal ones.
- Instead of searching for an oblique hyperplane, which can be computationally prohibitive, we propose to learn a proximal hyperplane in order to efficiently cluster the samples at each node guided by their labels.
- We propose an efficient incremental [1] update strategy for the oblique random forest tracker that is conservative in its memory needs. Therefore, instead of storing all the values of the "optimal" feature for each instance in order to update the decision nodes, apart from the indices for the randomly selected features, our method needs to store only two extra matrices of size $(\log M)^2$ and $(\log M + 1)$, where M is the dimensionality of the input features.
- Through extensive experiments on two recently proposed tracking benchmark datasets [54, 55], we show that our method compares well with recent state-of-the-art tracking methods in terms of accuracy and robustness.

[1] In this paper we use "incremental" and "online" interchangeably in the context of visual tracking.

The rest of the paper is organized as follows: a review of relevant works is presented in the following section. In section 4, we introduce our oblique random forest for visual tracking along with online update steps. In section 6, we present experimental results and compare with state-of-the-art methods. Finally, our conclusions are given in section 7.

2. Related work

In this section we first present a detailed review of work pertaining to discriminative trackers and random forest and highlight the novelty of our approach with respect to existing trackers and oblique random forest methods.

Advances in Discriminative Trackers. Discriminative trackers have received wide attention in the recent past evidenced by several recent studies, which have established the superiority of discriminative trackers [2, 22, 3, 4, 24]. Along with the established discriminative trackers that include multiple-instance boosting [4], kernelized structured SVM [24] and dictionary based trackers [5, 36, 62], several ensemble methods were proposed in [2, 22, 3]

Following the success of RaF on many vision applications [7, 43, 23, 11, 29, 46, 42, 19], an online RaF was first introduced for visual tracking in [41]. However, difficulties in updating the decision node parameters online resulted in poor performance when compared to methods such as Struck [24], P-N learning [27], multiple instance learning [4] and Sparse Coding based classifiers [36][5].

Oblique random forest. Most earlier RaF based tracking [41, 45] including the evaluation study by [20] rely on orthogonal random forest (see Fig. 2 for details), where at each node of the decision tree, an orthogonal hyperplane is exhaustively searched based on the numerical values of each feature. However, these decision tree induction methods do not fit the data well and are reported to be less accurate when compared to oblique trees [59, 37, 60].

Earlier works on oblique random forest have mostly focused on different hyperplane learning strategies as searching for the optimal oblique hyperplane is computationally expensive. Thus, many heuristic search methods based on deterministic hill-climbing CART-LC [10] and randomized search OC1 [38] were proposed. However, these methods often give suboptimal solutions as they search one dimension at a time for node splitting, which is computationally cumbersome in high-dimensional feature space.

Other closely related works include [59, 37], where PCA and LDA were used at each node to build an oblique RaF, and [60], where a multi-surface proximal SVM (MPSVM) was used to learn two hyperplanes. The MPSVM approach adopted therein relaxed the parallel constrains of the conventional SVM and learned two non-parallel decision hyperplanes. These methods solve a generalized eigenvalue problem at each non-leaf node which makes them computationally expensive. Moreover, all the above methods work on batch mode and extending them to online scenarios is not straight forward.

To the best of our knowledge, our work is the first attempt to incrementally learn an oblique random forest. In contrast to the earlier Obli-RaF method, we use PSVM at each decision node to fit an proximal hyperplane that takes into consideration the shape of the feature manifold as well as the distance to the proximal hyperplanes. PSVMs are efficient and amenable to incremental training. By clustering the samples of each class in a supervised fashion they preclude a compute intensive hyperplane search process.

ConvNet based trackers. Recent advances in visual tracking have been obtained with deep convolutional neural network (ConvNet) models. ConvNet and transfer learning based tracking were used in [51, 49] where a ConvNet is pre-trained with a huge amount of labeled images and fine-tuned in the process of tracking. Along similar lines, visual priors were learned from generic real-world images and then transferred for representing objects in a scene in [52]. Deep learning without pre-training was introduced by [61, 58]. In [47], a fully-convolutional neural network for visual tracking was proposed. In [48], ConvNets trained within an ensemble framework were proposed for visual tracking.

In our view, learning deep models for visual tracking with limited samples is challenging. Using a pre-trained model partly alleviates this problem by transferring rich feature hierarchies from large-scale image dataset such as ImageNet [40]. However, there may exist a large divergence between the source and target domain. To this end, we also propose a collaborative tracker in the particle filter framework by combining the merits of ConvNet models from [47] and our oblique random forest classifier (Obli-RaF). The ConvNet works in a generative fashion to predict the probability of whether each particle belongs to the object being tracked, whereas the Obli-RaF works as a discriminative classifier to classify the object from the background. This combination results in superior tracking performance as demonstrated in section 6.

3. Particle filter tracking

Our tracking algorithm is formulated within a particle filtering framework [39]. Let z^t and X^t denote the state variable describing the parameters of the object and the observation respectively in the t^{th} frame. In the particle filter framework, the true posterior state distribution $p(z^t|X^{1:t})$ is approximated by a finite set of P samples $\{z_i^t\}_{i=1}^P$ (called particles) with corresponding normalized weights $\{W_i^t\}_{i=1}^P$. The particles are drawn from the proposal density function $q(z^t|z^{1:t-1}, X^{1:t})$ which is set to the state transitional probability $p(z^t|z^{t-1})$. In practice, weight

W_i of particle i is given by the observation (likelihood) model

$$W_i^t = W_i^{t-1} * p(X^t|z_i^t), \tag{1}$$

The observation likelihood is given by the classification result and given as: $p(\mathbf{x}^t|z^t) = \frac{1}{K}\sum_k I[\mathbf{g}_k(\mathbf{x}_t) > 0]$, where $I[.]$ is the indicator function and $\mathbf{g}_k(\mathbf{x})$ is the result of a binary classification from the k^{th} decision tree in the ensemble with K decision trees. We model state parameters consisting of six parameters: translation in x-axis, translation in y-axis, scale variations, rotation angle, aspect ratio and skew angle is modeled using a Gaussian distribution assuming that the dimensions are independent.

4. Incremental Oblique Random Forests

In this section, we give a detailed description of our oblique random forest method denoted as Obli-RaF.

In our particle filter framework, we assume that we have access to N training samples (particles), $X^t = \{\mathbf{x}_1^t, \ldots, \mathbf{x}_N^t\}$ at each instant t obtained from region proposals. The samples are M dimensional: $\mathbf{x}_i^t \in \mathcal{X} \subseteq I\!R^M, i \in \{1, \ldots, N\}$. Our objective is to classify the samples in X^t as belonging to class $y = 1$, indicating the object of interest, or to class $y = -1$, indicating background region. We achieve this by learning a mapping function $\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y} \in \{-1, +1\}$. We denote a generic data point by $\mathbf{x}$ and use $\mathbf{x}_\diamond$, with $\diamond$ denoting the placeholder for the index where ever necessary.

Our mapping function $\mathcal{G}$ is a random forest [9], composed of K base classifiers $\mathcal{G} = \{\mathbf{g}_k\}_{k=1}^K$, where the classifiers $\mathbf{g}_k : \mathcal{X} \rightarrow \mathcal{Y}, k \in \{1, \ldots, K\}$, called *decision trees* are combined using bagging [8]. Each decision tree $\mathbf{g}_k(\mathbf{x})$ classifies a sample $\mathbf{x} \in \mathcal{X}$ by routing it from the root to some leaf node, recursively, which provides a label for the instance. Specifically, each node j in the tree is associated with a binary split function: $f_{kj}(\mathbf{x}, \theta) \in \{-1, +1\}$, where θ is the parameter of the split function. Samples are sent to the right child node if $f_{kj}(\mathbf{x}) = 1$ and to the left if $f_{kj}(\mathbf{x}) = -1$ with the process terminating at a leaf (or pure) node. Given an input $\mathbf{x}$, the output of the tree is the prediction stored at the leaf reached by $\mathbf{x}$, which is a target label $y \in \mathcal{Y}$ in our case.

4.1. Oblique decision tree

For each $\mathbf{g}_k$, we employ an oblique decision tree that results in a non-orthogonal hyperplane at each decision node. More specifically, a linear combination of the attributes is tested as follows:

$$f_{kj}(\mathbf{x}_\diamond) = \begin{cases} 1 & \text{if } \sum_{m=1}^{M} \mathbf{w}_m * \mathbf{x}_{\diamond m} < b \\ -1 & \text{otherwise} \end{cases}, \tag{2}$$

where $\mathbf{w}$ and b are the parameters of the hyperplane. As we mentioned in Section 2, exhaustive search for an optimal oblique hyperplane is computationally infeasible. Instead, we propose to learn the hyperplane from the data by recursively clustering the data samples in a supervised manner. Although any clustering method may be integrated into this framework, we propose to use proximal support vector machine (PSVM) [34] due to its advantages detailed in section 2. It is important to note that orthogonal decision tree clusters data samples by employing only one feature whereas oblique decision trees use a linear combination of features to perform this task. In the following, we describe how PSVM can incrementally learn supervised clusters.

4.2. PSVM learning

PSVM classifies data points depending on proximity to either one of the two separation planes that are aimed to be pushed away as far apart as possible. The rationale behind PSVM is that the separation hyperplanes are not bounded planes anymore, as done in conventional SVM [15], but are "proximal" planes. An illustration of PSVM hyperplanes and its relation to SVM is shown in Fig. 4. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in I\!R^{N\times M}$ be an $N \times M$ matrix obtained by stacking N samples in X. The non-italicised $\top$ indicates a transpose operation. Let $\mathbf{Y} \in \{-1, +1\}^N$ be a vector obtained by stacking the labels of samples in X. Here, we drop the time index t in X^t, since the following formulation applies to any t. We define a diagonal matrix $\mathbf{D}$, whose diagonal entries $D_{i,i} = 1$ if $\mathbf{x}_i$ belongs to the positive class and -1 otherwise. Then, PSVM aims at solving the following problem:

$$\begin{aligned} &\min_{(\mathbf{w},b,\boldsymbol{\xi})} \frac{1}{2}||\boldsymbol{\xi}||^2 + \nu * \frac{1}{2}(\mathbf{w}^\top\mathbf{w} + b^2) \\ &\text{s.t. } \mathbf{D}(\mathbf{X}\mathbf{w} - b\mathbf{e}) + \boldsymbol{\xi} = \mathbf{e}. \end{aligned} \tag{3}$$

where $\boldsymbol{\xi}$ is the error vector and ν is the regularization parameter, $\mathbf{w}$ and b are the coefficient of the hyperplane, and $\mathbf{e}$ is a vector of all ones. The parameters of PSVM $\{\mathbf{w}, b\}$, can be computed in closed form and is given by [34]:

$$[\mathbf{w}; b]^\top = (\nu I + \mathbf{H}^\top\mathbf{H})^{-1}\mathbf{H}^\top\mathbf{D}\mathbf{e};\ \mathbf{H} = [\mathbf{X}, -\mathbf{e}] \tag{4}$$

As demonstrated in Fig. 4, unlike conventional SVM, the planes $\mathbf{w}^\top\mathbf{x} = b \pm 1$ are not bounding planes anymore. In fact, they can be regarded as "proximal" planes, around which the points of each class are clustered and which are pushed as far apart as possible by the term $(\mathbf{w}^\top\mathbf{w} + b^2)$ in the objective function Eq. 3.

Although it may seem counter-intuitive to use the hyperplane of Eqn. (4) to split the data points in each node of the decision tree due to $\mathcal{O}(M^3)$ time complexity of matrix inversions, it turns out to be more efficient in practice, since only a small number of features M_s (usually of the order of $\log M$) are used in each node.

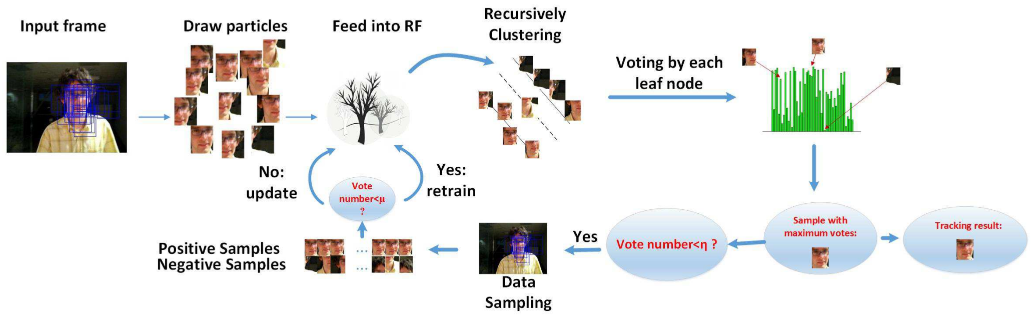
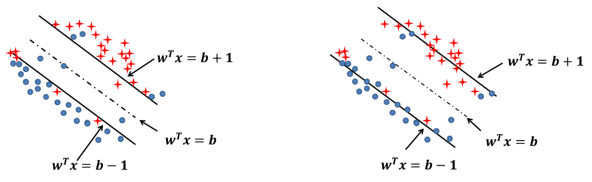


Figure 3. Overview of the proposed oblique random forest baseline tracker (Obli-RaF). In frame t, we sample 400 particles based on the result of previous tracking result. Those particles are fed into the oblique random forest classifier. Each tree in the forest recursively clusters the data samples. Each leaf node in the tree will vote for one of the two classes (target object or background). The one with maximum vote will be considered as the tracking result. The model is updates when the number of votes is less than a threshold η. Moreover, the model is retrained when the number of votes is less than μ ($\mu < \eta$). It can be easily generated by combining with other particle filter based trackers such as the ConvNets models in [47]. [Best viewed in color]



(a) SVM (b) PSVM

Figure 4. Separating hyperplanes of SVM and PSVM. Unlike conventional SVM which aims at maximizing the margin, PSVM classifies data points depending on proximity to either one of the two "clustering" planes that are aimed to be pushed away as far apart as possible.

4.3. Online updates

Model update avoids target drift and thus plays an important role in tracking performance. To this end, we propose an efficient method to update the PSVM model parameters when necessary.

Let $\mathbf{De} = \mathbf{Y}$ and suppose at time t, we have the solution $\boldsymbol{\beta}_t = [\mathbf{w_t}, b_t]^\top$. We can calculate $\boldsymbol{\beta}_{t+1}$ at time $t+1$ with new available data recursively from $\boldsymbol{\beta}_t$ without directly solving Eq. (4). The key problem in updating the parameters is the calculation of $(\mathbf{H}^\top\mathbf{H}+\nu I)^{-1}$. If the features corresponding to the new available data is $\mathbf{H}_{t+1}$ at time $(t+1)$, then the problem of estimating $\boldsymbol{\beta}_{t+1}$ becomes:

$$\underset{\boldsymbol{\beta}_{t+1}}{\text{minimize}} \left\| \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix} \boldsymbol{\beta}_{t+1} - \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+1} \end{bmatrix} \right\|^2 + \nu ||\boldsymbol{\beta}_{t+1}||^2 \quad (5)$$

This is a least squares minimization problem, which leads to the following online update of the parameters based on recursive least squares (RLS)[2]:

$$\begin{aligned} \boldsymbol{\varphi}_{t+1} &= \boldsymbol{\varphi}_t - \boldsymbol{\varphi}_t \mathbf{H}_{t+1}^\top (I + \mathbf{H}_{t+1}\boldsymbol{\varphi}_t \mathbf{H}_{t+1}^\top)^{-1} \mathbf{H}_{t+1}\boldsymbol{\varphi}_t \\ \boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t + \boldsymbol{\varphi}_{t+1}\mathbf{H}_{t+1}^\top (Y_{t+1} - \mathbf{H}_{t+1}\boldsymbol{\beta}_t), \end{aligned} \quad (6)$$

where

$$\boldsymbol{\varphi}_{t+1} = \left[\begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix} + \nu I \right]^{-1}, \quad (7)$$

The term $\mathbf{Y}_{t+1} - \mathbf{H}_{t+1}\boldsymbol{\beta}_t$ in Eqn. (6) will be called the *innovation*. Parameters over subsequent iterations remain the same when this term is zero.

Convergence of RLS is an established theory in signal processing and it is proven to be faster than least mean square methods for streaming data. Furthermore, our quadratic objective function leads to a global optimum solution The proposed incremental Obli-RaF is straightforward to implement. We firstly train an oblique random forest from scratch. At each step t, the random forest receives a block of incoming examples $\mathbf{X}_t$, whose labels are predicted. When the true labels $\mathbf{Y}_t$ are revealed, the oblique random forest updates its each decision tree using Eqn. (6) with the available misclassified data samples.

5. Implementation

We test the feasibility of the proposed Obli-RaF classifier for visual tracking under two different scenarios: i) simple Obli-Raf tracker and ii) Obli-Raf tracker with ConvNet models. We present details of both the implementations below:

The Obli-RaF tracker has a simple implementation. The video frames are first converted to gray scale, and the state

[2]Details regarding this derivation are provided in the supplementary material.

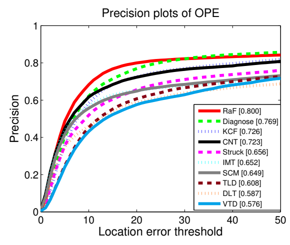
of the target (*i.e.*, size and location) in the first frame is given by the ground truth. The size of the warped image is set to 32×32. The six target state parameters for each state are set as $[8, 8, 0.03, 0.005, 0.005, 0.005]$, respectively. We use histogram of gradients (HOG) [16] features obtained with a cell size of 4 and 9 different orientations to represent every bounding box. We set the number of decision trees K to 100, the maximum depth of a tree Γ_{depth} to 400 and the number of features used at each decision node M_s to $\text{round}(\log(M))$. At each step, we use 100 positive and 100 negative samples to train an Obli-RaF. The confidence threshold η in Fig. 3 is set to 85, which is updated using 20 positive and 20 negative samples whenever necessary. We retrain the random forest when the maximum number of vote is less than 20. The number of particles are set to be 100. The above parameters are fixed for all experiments. A detailed sensitivity analysis of these parameters are provided in the supplementary material.

Obli-RaFT with ConvNet We also propose a second implementation of our tracker motivated by the recent successes of ConvNets. This bears some similarity to collaborative models of [63]. To realize this, we follow the same pipeline as in [47] to train two tiny ConvNets on feature maps from conv4-3 and conv5-3 layers of VGG-16 [44] model. The two ConvNets are then used to estimate a heat map of the target object in a *generative* manner. Meanwhile the proposed Obli-RaF works in a *discriminative* manner to predict whether one particle belongs to the object of interest or the background. The final confidence of the particle is obtained as the sum of the confidence from our Obli-RaF method and the confidence of the generative ConvNets. For the ConvNets part from we adopt the same parameter setting as in [47].

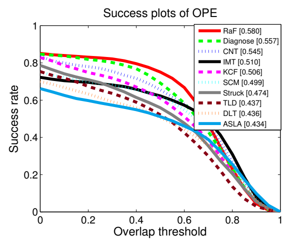
The proposed simple Obli-RaF tracker is implemented in MATLAB and runs at around 3 frames per second on a PC with Intel i7 3770 CPU (3.4 GHz). The ConvNets based Obli-RaF tracker is implemented in MATLAB and Caffe [26] and runs at around 2 frames per second on the same machine with 2 TitanX GPUs.

6. Experimental Results

We firstly demonstrate the superiority of the proposed incremental Obli-RaF method over other machine learning algorithms with the OTB-51 benchmark[54]. Then we explore more details on the ConvNets based Obli-RaF tracker with both OTB-51 and OTB-100 [54, 55]. Both benchmarks contain video sequences with a variety of challenges for visual tracking. The OTB-51 [54] evaluates 29 state-of-the-art trackers with 51 challenging video sequences. The OTB-100 [55] extends OTB-51 by including another challenging video sequences. To better evaluate and analyze the strengths and weaknesses of different tracking approaches,



(a) Distance Precision



(b) Overlap Success

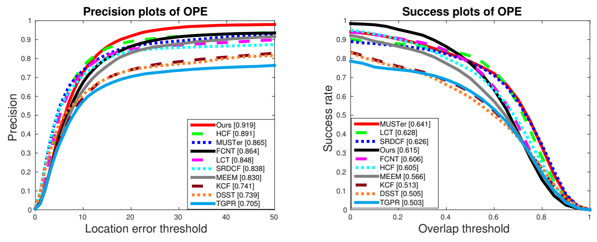
Figure 5. Comparison of the simple Obli-RaF tracker to other methods on OTB-51 [54]. Following the work of [54], precision plots obtained with a threshold of 20 pixels are shown on the left. Success plots measured using AUC values are shown on the right. We see that our simple Obli-RaF method outperforms all existing methods. Best viewed in color.

the videos are categorized with 11 attributes based on different challenging factors including low resolution (LR), in-plane rotation (IPR), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), deformation (DEF), background clutters (BC), illumination variation (IV), motion blur (MB), fast motion (FM), and out-of-view (OV). We also report the results on VOT2016 in the supplementary file.

We quantitatively evaluate trackers using precision curve and success plot. We follow the protocol in [54, 55] and use same parameter values for all the following evaluations. More specifically, the precision curve uses a threshold of 20 pixels to rank different method while in success plot, AUC value is employed. For a comparative evaluation of our method, we include several recent state-of-the-art trackers including KCF [25], DLT [51], and CNT [58], IMT [56],LCT [33] and the “Diagnose” framework of [50] in our comparative analysis. Note that all the performance plots are generated using the code library from the benchmark evaluation [54], and the results of KCF [25], and DLT [51], IMT [56], CNT [58] methods are provided by the authors.

Simple Obli-RaF tracker. We evaluate the simple Obli-RaF tracker throughout a test sequence with initialization from the ground truth position in the first frame and report the average precision or success rate. Fig. 5 shows the results of the simple Obli-RaF tracker under one-pass evaluation (OPE).

The proposed oblique random forest tracker ranks first in both cases. In the precision plot, the proposed RaF tracker achieves a precision score of 0.8, which outperforms the best results of [50] by 4%. Furthermore, the RaF tracker achieves an AUC value of 0.58, which outperforms the second best method by around 2%. Note that the second best tracker is obtained by the best combination of 5 different feature extraction, 4 different classification/regression models, 3 different motion models and 2 different motion update strategies. Most of the trackers involved in this evaluation are based on hand-crafted features. For example, the well-known Struck tracker [24] employs Haar-like features and relies on structural SVM for online learning. These promising results demonstrate the effectiveness of the proposed online Obli-RaF methods.



(a) Distance Precision on OTB-51 (b) Overlap Success on OTB-51

Figure 6. The precision (left) and success plots (right) of OPE of the proposed method and other advanced trackers on OTB-51.

Obli-RaF Vs Orth-RaF. In order to assess the advantages of the online Obli-RaF, we also implemented an orthogonal random forest (Orth-RaF) tracker based on [41]. Table 1 compares the precision score (within 20 pixels) and Table 2 compares the success rate (AUC). Clearly, the proposed oblique random forest outperforms the orthogonal random forest in all cases.

As noted earlier (section 1), there are two main reasons for the observed performance gap: (i) flexibility of Obli-RaF which is not restricted to be axis aligned to the coordinate system of the input features, and (ii) efficient online update procedure of our method which better captures variations in the target object. On the other had Orth-RaF [41] lacks both (i) and (ii).

Obli-RaF tracker with ConvNets. Next, we evaluate our Obli-RaF tracker in combination with ConvNets and compare it with recently proposed trackers including MEEM [57], SRDCF [17], HCF [32], and FCNT [47] on both OTB-51 and OTB-100 datasets.

From Fig. 6 we see that our proposed collaborative (Obli-RaF + ConvNets) method outperforms all other methods in terms of precision curve and achieves competitive results on success plots. More specifically, our collaborative model, which consists of FCNT [47] and Obli-RaF tracker, achieves 5.5% and 1% improvement over FCNT tracker on precision and success measures respectively. We present a more detailed analysis on the advantage of this combination in supplementary file.

We also report the performance of the proposed Obli-RaF + ConvNets tracker in terms of temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). They evaluate trackers by starting at different frames (perturbation by sampling initialization temporally) and initializing with different bounding boxes (perturbation by sampling initialization spatially), respectively. Results are presented in Fig. 7. Our proposed method achieves the first and second rank on the precision curve of SRE and TRE, respectively. In Fig. 8, 9 and 10, we also present our results comparing the proposed Obli-RaF + ConvNets tracker with several state-of-the-art methods on OTB-100 dataset.

We observe that the proposed method achieves competitive results compared to other state-of-the-art methods. More specifically, our method achieves the best performance in terms of precision plot of OPE. As shown in Fig.8(a), our result is 1.4% better than the second best. It also achieves the second best on the TRE and SRE metric in Fig. 9(a) and Fig. 10(a). For the success plot, which uses AUC value to rank each method, our method's performance is still below that of the SRDCF [17] tracker. However, the SRDCF tracker estimates scale variations explicitly by employing a dedicated model. However, this strategy also works to its disadvantage in the precision curves of Fig.8(a), 9(a) and 10(a), where it is 6.2% lower than our method. Note that our method estimates the object's state variation using only the commonly used particle filter and no other model to estimate scale changes. Nevertheless, it outperforms SRDCF in the precision curves in all the cases and performs competitively on the success plots of Fig.8(b), 9(b) and 10(b).

Computational complexity. The overall complexity of our oblique decision tree it is $\mathcal{O}(N * M_{\mathrm{s}}^3)$, while the complexity of an orthogonal decision tree is $\mathcal{O}(M_{\mathrm{s}} N(\log^2 N))$. However, only a few features ($M_{\mathrm{s}} = \mathrm{logM}$ in this study) are sampled at each internal node for many tasks such as visual tracking where a large number of training samples accumulate over time, resulting in $M_{\mathrm{s}} \ll \log N$. Thus, in practice, Obli-RaF has a smaller time complexity than Orth-RaF. This is also far less compared to CART-LC [10] Obli-RaF method which has $\mathcal{O}(N^{M_{\mathrm{s}}})$ complexity. Moreover, our tree induction method also results in shallow trees and thus, more efficient. On average, the proposed simple Obli-RaF tracker runs 3 times faster than the orthogonal one (more details on this and sensitivity analysis are in supplementary material).

7. Conclusion

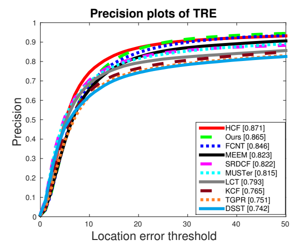
In this paper, we proposed a novel oblique random forest for visual tracking. In each node of the tree, a decision was learned by clustering the data reaching the node into two groups via efficient PSVM classifier. The resulting Obli-RaF can better capture the geometric structure of the data and usually leads to smoother decision boundaries. This is in contrast to previous methods which exhaustively search for the best feature to split the data. Furthermore, we also proposed an algorithm to efficiently update the classifier in order to alleviate tracking drift. Through extensive experimental evaluation we showed that the proposed Obli-

Table 1. Comparison between Orthogonal Random Forest and Oblique Random Forest on the precision score of OTB-51 (within 20 pixels).

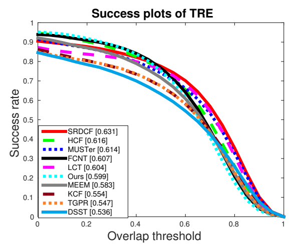
Method	overall	IV	SV	Occ	Def	MB	FM	IR	OR	OV	BC	LR
Orth-RaF	67.4	60.0	62.5	66.8	59.5	57.4	54.9	57.0	66.4	45.9	58.8	58.0
Obli-RaF	80.0	80.5	80.0	73.5	70.9	73.6	73.0	79.1	77.8	68.8	77.1	60.3

Table 2. Comparison between Orthogonal Random Forest and Oblique Random Forest on the success rate of OTB-51 (AUC).

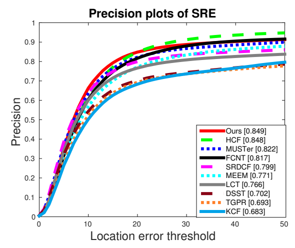
Method	overall	IV	SV	Occ	Def	MB	FM	IR	OR	OV	BC	LR
Orth-RaF	48.1	42.8	44.7	47.8	41.8	41.6	40.4	42.9	47.6	37.3	42.7	28.0
Obli-RaF	58.0	58.8	58.2	53.4	52.2	56.0	55.2	56.7	55.6	54.5	55.7	46.1



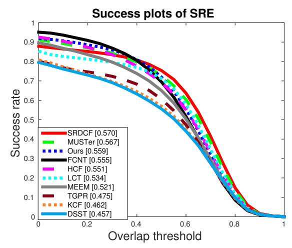
(a) Distance Precision of TRE



(b) Overlap Success of TRE

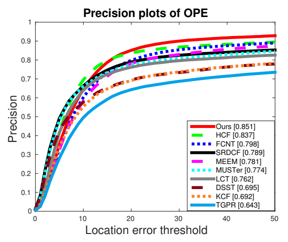


(c) Distance Precision of SRE

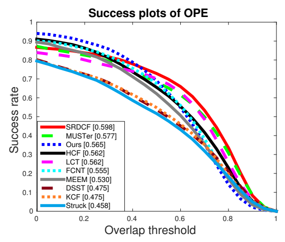


(d) Overlap Success of SRE

Figure 7. Precision (left) and success plots (right) of TRE and SRE for the proposed method and other advanced trackers on OTB-51.

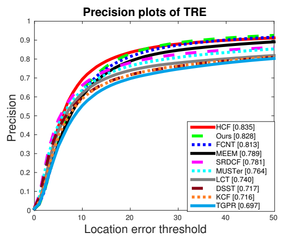


(a) Distance Precision of OPE

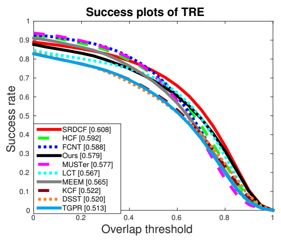


(b) Overlap Success of OPE

Figure 8. OPE on OTB-100.



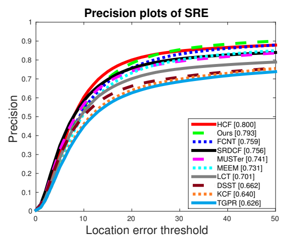
(a) Distance Precision of TRE



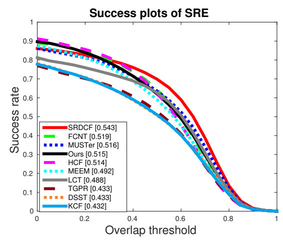
(b) Overlap Success of TRE

Figure 9. TRE on OTB-100.

RaF tracker with simple HOG features outperforms many other state-of-the-art methods. When combined with a generative ConvNet based model, in a collaborative manner, our method resulted in a more powerful tracking framework comparing well with other advanced trackers.



(a) Distance Precision of SRE



(b) Overlap Success of SRE

Figure 10. SRE on OTB-100.

Acknowledgment

This study is supported by the research grant for the Human Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapores Agency for Science, Technology and Research (A*STAR).

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 798–805. IEEE, 2006.

[2] S. Avidan. Support vector tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, Aug. 2004.

[3] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007.

[4] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.

[5] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1830–1837. IEEE, 2012.

[6] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26:6384, 1998.

[7] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proceedings of the IEEE Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[8] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[9] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[11] S. Bulo and P. Kontschieder. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–88, 2014.

[12] S. Bul and P. Kontschieder. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–88, June 2014.

[13] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[14] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[15] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[16] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[17] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[18] M. Dantone, J. Gall, C. Leistner, and L. V. Gool. Human pose estimation using body parts dependent joint regressors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3041–3048, June 2013.

[19] A. Dapogny, K. Bailly, and S. Dubuisson. Pairwise conditional random forests for facial expression recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3783–3791, 2015.

[20] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[21] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260–267. IEEE, 2006.

[22] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings of British Machine Vision Conference*, volume 1, page 6, 2006.

[23] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1732–1740, 2015.

[24] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *Proceedings of IEEE Conference on Computer Vision*, pages 263–270. IEEE, 2011.

[25] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[27] Z. Kalal, J. Matas, and K. Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–56. IEEE, 2010.

[28] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo. Deep neural decision forests. In *Proceedings of IEEE Conference on Computer Vision*, December 2015.

[29] E. Krupka, A. Vinnikov, B. Klein, A. Hillel, D. Freedman, and S. Stachniak. Discriminative ferns ensemble for hand pose recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3670–3677, 2014.

[30] J. Kwon and K. M. Lee. Visual tracking decomposition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1269–1276. IEEE, 2010.

[31] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of International Joint Conference on Artifical Intelligence*, volume 81, pages 674–679, 1981.

[32] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *IEEE International Conference on Computer Vision*, pages 3074–3082, 2015.

[33] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5388–5396, 2015.

[34] O. L. Mangasarian and E. W. Wild. Proximal support vector machine classifiers. In *Proceedings KDD-2001: Knowledge Discovery and Data Mining*. Citeseer, 2001.

[35] N. Manwani and P. Sastry. Geometric decision tree. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,*, 42(1):181–192, 2012.

[36] X. Mei, H. Ling, Y. Wu, E. P. Blasch, and L. Bai. Efficient minimum error bounded particle resampling l1 tracker with occlusion detection. *IEEE Transactions on Image Processing*, 22(7):2661–2675, 2013.

[37] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[38] S. K. Murthy, S. Kasif, S. Salzberg, and R. Beigel. Oc1: A randomized algorithm for building oblique decision trees. In *Proceedings of AAAI*, volume 93, pages 322–327. Citeseer, 1993.

[39] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[41] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *2009 IEEE 12th International Conference on Computer Vision Workshops*, pages 1393–1400. IEEE, 2009.

[42] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3799, 2015.

[43] S. Schulter, C. Leistner, P. Wohlhart, P. Roth, and H. Bischof. Accurate object detection with joint classification-regression random forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 923–930, 2014.

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

[45] D. J. Tan and S. Ilic. Multi-forest tracker: A chameleon in tracking. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1202–1209, 2014.

[46] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class hough forests for 3d object detection and pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 462–477. Springer, 2014.

[47] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *IEEE International Conference on Computer Vision*, pages 3119–3127, 2015.

[48] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, 2016.

[49] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015.

[50] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia. Understanding and diagnosing visual tracking systems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3101–3109, 2015.

[51] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, pages 809–817, 2013.

[52] Q. Wang, F. Chen, J. Yang, W. Xu, and M.-H. Yang. Transferring visual prior for online object tracking. *IEEE Transactions on Image Processing*, 21(7):3296–3305, 2012.

[53] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *Proceedings of IEEE Conference on Computer Vision*, pages 1323–1330. IEEE, 2011.

[54] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, 2013.

[55] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[56] J. H. Yoon, M. H. Yang, and K. J. Yoon. Interacting multiview tracker. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(5):903–917, May 2016.

[57] J. Zhang, S. Ma, and S. Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *European Conference on Computer Vision*, pages 188–203. Springer, 2014.

[58] K. Zhang, Q. Liu, Y. Wu, and M. H. Yang. Robust visual tracking via convolutional networks without training. *IEEE Transactions on Image Processing*, 25(4):1779–1792, April 2016.

[59] L. Zhang and P. N. Suganthan. Random forests with ensemble of feature spaces. *Pattern Recognition*, 47(10):3429–3437, 2014.

[60] L. Zhang and P. N. Suganthan. Oblique decision tree ensemble via multisurface proximal support vector machine. *IEEE Transactions on Cybernetics*, 45(10):2165–2176, 2015.

[61] L. Zhang and P. N. Suganthan. Visual tracking with convolutional random vector functional link neural network. *IEEE Transactions on Cybernetics*, 2016.

[62] T. Zhang, S. Liu, N. Ahuja, M.-H. Yang, and B. Ghanem. Robust visual tracking via consistent low-rank sparse learning. *International Journal of Computer Vision*, 111(2):171–190, 2015.

[63] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparse collaborative appearance model. *IEEE Transactions on Image Processing*, 23(5):2356–2368, 2014.