Randomized Ensemble Tracking

Qinxun Bai[1], Zheng Wu[1], Stan Sclaroff[1], Margrit Betke[1], Camille Monnier[2]
[1] Boston University, Boston, MA 02215, USA
[2] Charles River Analytics, Cambridge, MA 02138, USA

Abstract

We propose a randomized ensemble algorithm to model the time-varying appearance of an object for visual tracking. In contrast with previous online methods for updating classifier ensembles in tracking-by-detection, the weight vector that combines weak classifiers is treated as a random variable and the posterior distribution for the weight vector is estimated in a Bayesian manner. In essence, the weight vector is treated as a distribution that reflects the confidence among the weak classifiers used to construct and adapt the classifier ensemble. The resulting formulation models the time-varying discriminative ability among weak classifiers so that the ensembled strong classifier can adapt to the varying appearance, backgrounds, and occlusions. The formulation is tested in a tracking-by-detection implementation. Experiments on 28 challenging benchmark videos demonstrate that the proposed method can achieve results comparable to and often better than those of state-of-the-art approaches.

1. Introduction

Many tracking-by-detection methods have been developed for visual tracking applications [2, 3, 8, 17, 20, 21]. The motivation to treat tracking as a detection problem is to avoid having to model object dynamics especially when abrupt motion and occlusions can occur.

Tracking-by-detection requires training of a classifier for detecting the object in each frame. One common approach for detector training is to use a detector ensemble framework that linearly combines the weak classifiers with different associated weights, e.g., [2, 8]. A larger weight implies that the corresponding weak classifier is more discriminative and thus more useful. To date, most previous efforts have focused on adapting offline ensemble algorithms into online mode. This strategy, despite its success in many online visual learning tasks, has limitations in the visual tracking domain.

First, the common assumption that the observed data, examples and their labels, have an unknown but *fixed* joint distribution does not apply in tracking scenarios where the appearance of an object can undergo such significant changes that a negative example in the current frame looks more similar to the positive example identified in the past (Fig. 1). Given the uncertainty in the appearance changes that may occur over time and the difficulty of estimating the non-stationary distribution of this observed data directly, we propose a method that models how the classifier weights evolve according to a non-stationary distribution.

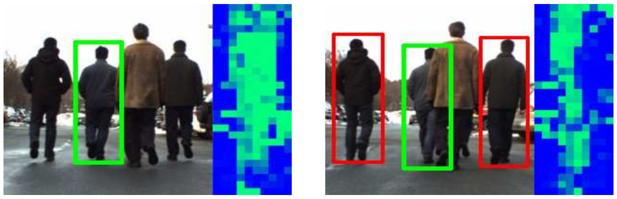


Figure 1. The proposed method can track a target person (green box) during partial occlusion and in the presence of distractors (red boxes). It weights the "reliability" of the weak classifiers within the box (green means high weight; blue means low weight). In the left image, the person is unoccluded; therefore, high weights are associated with weak classifiers that cover the body. In the right image, the person is partially occluded; consequently, "reliability" weights for the occluded portion decrease while weights for the unoccluded portion are reinforced. Thus, the ensemble tracker can distinguish the tracked person from the distractors.

Second, many online self-learning methods update the weights of their classifiers by first computing the importance weights of the incoming data. As noted by Grabner and Bischof [8], however, there are difficulties in computing these weights. This becomes even more challenging when it is recognized that the distribution that generated the data is non-stationary. We suggest that this is an inherent challenge for online self-learning methods and propose an approach for estimating the ensemble weights that is Bayesian and ensures that the update of the ensemble weights is smooth.

Our method models the weights of the classifier ensemble with a non-stationary distribution, where the weight vector is a random variable whose instantiation can be interpreted as a representation of the "hidden state" of the combined strong classifier. Our method detects an object of interest by inferring the posterior distribution of the ensemble weights and computing the expected output of the ensemble classifier with respect to this “hidden state”. This strategy is similar to the technique of filtering used by non-stationary systems described in the tracking literature. Our method differs, however, by its focus on estimating the state of the classifier, not the state of the object (position, velocity, etc). In summary, our contributions are:

1. We propose a classifier ensemble framework for tracking-by-detection that uses Bayesian estimation theory to estimate the non-stationary distribution of classifier weights.
2. Our randomized classifier encodes the “relative reliability” among a pool of weak classifiers, which provides a probabilistic interpretation of which features of the object are relatively more discriminative.
3. By integrating a performance measure of the weak classifiers with a fine-grained object representation, our ensemble tracker is able to identify the most informative local patches of the object and successfully interpret partial occlusion and detect ambiguities due to distractors.

We evaluate the implementation of our method on 28 challenging benchmark video sequences. Our experiments demonstrate that the method can detect an object in tracking scenarios where the object undergoes strong appearance changes, where it moves and deforms, where the background changes significantly, and where distracting objects appear and interact with the object of interest. Our method attains results that are comparable to and often better than state-of-the-art tracking approaches.

2. Related Work

A tracking-by-detection method usually has two major components: object representation and model update. Previous methods employ various object representations [21, 20, 1, 4, 7, 16], and our approach is most related to methods that use local patches [1, 4]. However, instead of sampling local patches either randomly [1] or in a controlled way [4], our representation exploits local patches according to a fine grid of the object template. This makes the ensemble weights for weak classifiers informative, indicating the spatial spread of “discriminative ability” over the object template.

The model update scheme for tracking-by-detection has also been widely studied in literature. An online feature selection mechanism was proposed by Collins et al. [5] who evaluated feature discriminability during tracking. Avidan [2], who was the first to explicitly apply ensemble methods to tracking-by-detection, extended the work of [5] by adopting the Adaboost algorithm to combine a set of weak classifiers maintained with an online update strategy. Along this thread, a boosting process by Grabner et al. [8] was extended from the online boosting algorithm [15] by introducing feature selection from a maintained pool of features for weak classifiers. Several other extensions to online boosting also exist, including the work by Saffari et al. [17] who proposed an online multi-class boosting model, and by Babenko et al. [3] who adopted Multiple Instance Learning in designing weak classifiers. In a different approach [18], Random Forests undergo online update to grow and/or discard decision trees during tracking.

Our online ensemble method is most related with online boosting scheme, in the sense that we adopt weighted combination of weak classifiers. However, we characterize the ensemble weight vector as a random variable and evolve its distribution with recursive Bayesian estimation. As a result, the final strong classifier is an expectation of the ensemble with respect to the weight vector, which is approximated by an average of instantiations of the randomized ensemble. To the best of our knowledge, in the context of tracking-by-detection, we are the first to present such an online learning scheme that characterizes the uncertainty of a self-learning algorithm and enables a Bayesian update of the classifier.

3. Randomized Ensemble Tracker

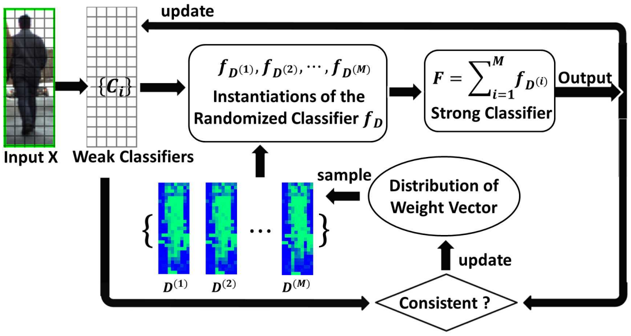

update
$\{C_i\}$
$f_{D^{(1)}}, f_{D^{(2)}}, \cdots, f_{D^{(M)}}$
Instantiations of the Randomized Classifier f_D
$F = \sum_{i=1}^{M} f_{D^{(i)}}$
Strong Classifier
Output
Input X
Weak Classifiers
sample
Distribution of Weight Vector
$\{$ ... $\}$
$D^{(1)}$
$D^{(2)}$
$D^{(M)}$
update
Consistent ?

Figure 2. Overview of our method.

We now give an overview of our tracking system (diagram shown in Fig. 2). At each time step, our method starts with the pool of weak classifiers $\mathcal{C} = \{c_1, c_2, \cdots, c_N\}$, a distribution $Dir(D)$ over the weight vector D and input data $\mathbf{x}$. Our method divides the input $\mathbf{x}$ into a regular grid of small patches, and sends the feature extracted from each small patch to its corresponding weak classifier. At the same time, our method also samples the distribution $Dir(D)$ to obtain M instantiations $D^{(1)}, D^{(2)}, \cdots, D^{(M)}$ of the weight vector D (color maps in Fig. 2) and combines them with the output of weak classifiers to yield M ensembles of weak classifiers $f_{D^{(1)}}, f_{D^{(2)}}, \cdots, f_{D^{(M)}}$. These M ensembles can be interpreted as M instantiations of the randomized classifier f_D and are used to compute the approximation F of the expected output of the randomized classifier f_D. The approx-

Table 1. Notation for our classification method

c_i	Weak classifier
D	N-dimensional weight vector
d_i	Component of weight vector associated with c_i
α	Concentration parameter for the Dirichlet distribution
H	Base distribution of the Dirichlet distribution
f_D	Randomized classifier that depends on a weight vector D
F	Final strong classifier
g_i	Performance measure of weak classifier c_i

imation F is considered the output of the strong classifier created by our ensemble scheme for input data $\mathbf{x}$. To evaluate new input data from the next frame, our method updates the distribution $Dir(D)$ in a Bayesian manner by observing the agreement of each weak classifier with the strong classifier ensemble. The method also updates the pool of weak classifiers according to the output of the strong classifier.

The weight vector D is a distribution over a finite discrete space, here the index space $\mathbf{I} = \{1, \dots, N\}$ of our classifier pool. We chose to model the distribution over the weight vector D as a Dirichlet distribution $Dir(D)$. The Dirichlet-multinomial conjugacy [14] makes the Bayesian posterior update of D simple and evolvable. Initialization of the model is performed in the first frame of the image sequence, where the pool of weak classifiers is initialized with the given ground truth and the Dirichlet prior for weight vector D is initialized uniformly.

3.1. Classification by Voting

We now describe our classification method, which is summarized in Algorithm-1. Our notation is listed in Table 1. Each weak classifier c_i of the pool $\mathcal{C}$ is a binary classifier and outputs a label 1 or 0 for each input data.

Given a weight vector D, we obtain an ensemble binary classifier f_D of the pool by thresholding the linear combination of outputs $c_i(\mathbf{x})$ of all weak classifiers:

$$f_D(\mathbf{x}) = \begin{cases} 1 & \text{if} \sum_{i=1}^{N} d_i c_i(\mathbf{x}) \geq \tau \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathbf{x}$ denotes the input data and threshold τ is a model parameter. Notice that f_D is a function of a draw of the random variable D and is therefore a randomized classifier.

ALGORITHM-1:CLASSIFICATION BY CLASSIFIER ENSEMBLE

```
Given input $\mathbf{x}$, weak classifier pool $\mathcal{C}$, distribution $Dir(D;\alpha,H)$:
  • Step 1: Draw $D^{(1)}, D^{(2)}, \cdots, D^{(M)}$ independently from $Dir(D;\alpha,H)$; Each draw is a $N$ dimensional weight vector.
  • Step 2: For each draw, compute $f_{D^{(i)}}(\mathbf{x})$ using Eq. 1.
  • Step 3: Compute ensemble output $F(\mathbf{x})$ by voting (Eq. 7).
```

If we denote the series of sequentially arriving data sets as $\mathbf{S}^{(0)}, \mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \cdots$ (where $\mathbf{S}^{(i)}$ in our application is the set of all scanning windows in the i-th frame), at time step t, given input data $\mathbf{x} \in \mathbf{S}^{(t)}$, our prediction of its probabilistic label value y^* is computed as follows:

$$\begin{aligned} y^* = E[y|\mathbf{x}] &= \int y \, p(y|\mathbf{x}, \mathbf{S}^{(0)} \cdots \mathbf{S}^{(t-1)}) dy \\ &= \int y \int p(y|\mathbf{x}, D) \, p(D|\mathbf{S}^{(0)} \cdots \mathbf{S}^{(t-1)}) dD dy \\ &= \int p(D|\mathbf{S}^{(0)} \cdots \mathbf{S}^{(t-1)}) (\int y \, p(y|\mathbf{x}, D) dy) dD \end{aligned} \tag{2}$$

where $\int y \, p(y|\mathbf{x}, D)dy$ is the expectation $E(y|\mathbf{x}, D)$ of y conditioned on a given weight vector D. Given that y takes discrete values from $\{0, 1\}$, this conditional expectation is:

$$\begin{aligned} E(y|\mathbf{x}, D) &= p(y = 1|\mathbf{x}, D) \\ &= p(\sum_{i=1}^{N} c_i(\mathbf{x}) d_i \geq \tau | \mathbf{x}, D) \\ &= f_D(\mathbf{x}). \end{aligned} \tag{3}$$

Applying this to Eq. 2 yields the expectation

$$y^* = \int f_D(\mathbf{x}) \, p(D|\mathbf{S}^{(0)} \cdots \mathbf{S}^{(t-1)}) dD, \tag{4}$$

where $p(D|\mathbf{S}^{(0)} \cdots \mathbf{S}^{(t-1)})$ follows the Dirichlet distribution $Dir(D;\alpha,H)$.

Since $f_D(\mathbf{x})$ is nonlinear, Eq. 4 does not have a closed-form solution and we approximate it by sampling the Dirichlet distribution $Dir(D;\alpha,H)$. In particular, an instantiation $(d_1, d_2, \cdots, d_N)$ of the random variable D is drawn from a Dirichlet distribution,

$$(d_1, d_2, \cdots, d_N) \sim Dir(D;\alpha,H), \tag{5}$$

$$Dir(D;\alpha,H) = \frac{\Gamma(\alpha)}{\prod_{i=1}^{N} \Gamma(\alpha h_i)} \prod_{i=1}^{N} d_i^{\alpha h_i - 1}. \tag{6}$$

The parameter $H = (h_1, h_2, \cdots, h_N)$ is the base distribution, which is the expectation of vector D. Scalar α is the concentration parameter, which characterizes how closely a sample from the Dirichlet distribution is related to H. Both H and α are updated online for incoming data.

Our method computes M i.i.d. samples from Dir, i.e., $D^{(1)}, D^{(2)}, \cdots, D^{(M)}$. To obtain the final ensemble classifier for input $\mathbf{x}$, our method approximates Eq. 4 by voting $f_{D^{(1)}}, f_{D^{(2)}}, \cdots, f_{D^{(M)}}$ and thresholding as follows:

$$F(\mathbf{x}) = \begin{cases} 1 & \text{if} \frac{1}{M} \sum_{j=1}^{M} f_{D^{(j)}}(\mathbf{x}) \geq 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

When multiple positive predictions are presented, our method selects the one with the highest real-valued score produced by voting before thresholding (0.5).

3.2. Model Update

Our method updates both the Dirichlet distribution of weight vectors and the pool of weak classifiers after the classification stage in each time step, so that the model can evolve. It updates the Dirichlet parameters α and H in a Bayesian manner. In fact, our online ensemble method as well as its update scheme does not enforce any constraints on the form of the weak classifiers, as long as each weak classifier is able to cast a vote for every input sample. The construction of weak classifiers and the mechanism of updating them can be chosen in an application-specific way.

For each step, after performing the classification, our method obtains the labels of data predicted by our strong classifier F and the observation of performance of weak classifiers, that is, the prediction consistency of weak classifiers with respect to the strong classifier. Throughout this paper, we use the terms "consistency" or "consistent" to indicate agreement with the strong classifier F.

We formulate our observation model as a multinomial-like distribution, which enables a simple posterior update due to multinomial-Dirichlet conjugacy. Since the weight vector D is a measure of the "relative reliability" over the pool of classifiers, its posterior is a function of the "observation of relative reliability of each classifier." To formally represent it, we consider a *performance measure* of each weak classifier c_i, which we denote as g_i, while $(g_1, g_2, ..., g_N)$ forms an observation of D. Given a weight vector D, g_i should have an expectation proportional to the weight value d_i. Recall that the expectation of occurrence rate of a particular outcome in a multinomial distribution is just the distribution parameter for that outcome. Hence, if we regard a given weight vector D as multinomial parameter, g_i could be regarded as the "rate of being a reliable classifier" as analogous to occurrence rate. For ease of computation, if we further multiply g_i by the total number of observations, then it becomes the "number of occurrences that are reliable classifiers," which leads to the following multinomial-like distribution for the observation model

$$p(g_1 \cdots g_N|D) = k\prod_{i=1}^{N}(d_i)^{g_i}, \tag{8}$$

where k is a normalization constant. Then the posterior distribution of D can be obtained by Bayes rule

$$\begin{aligned} p(D|\alpha, H, g_{1:N}) &\propto p(g_1 \cdots g_N|D)p(D|\alpha, H) \\ &\propto \prod_{i=1}^{N}(d_i)^{\alpha h_i + g_i - 1} \\ &= Dir(D; \alpha', H'). \end{aligned} \tag{9}$$

The updated base distribution H' in Eq. 9 is given by

$$H' = \frac{\alpha H + \sum_{i=1}^{N}\delta_i g_i}{\alpha + \sum_{i=1}^{N} g_i}, \tag{10}$$

where δ_i is an unit vector where the i-th entry equals 1.

In defining the *performance measure* g_i, there are two concerns. First, g_i should be nonnegative, since it is the outcome of multinomial trials. Second, "positive" and "negative" weak classifiers should be evaluated symmetrically with respect to some neutral value; neutral values account for cases where the observations of classifier performance may be ambiguous or missing, e.g., during occlusion. We choose the following function:

$$\begin{aligned} g : \{1, 2, \cdots, N\} &\rightarrow [0, 2] \\ g_i = g(i) &= \frac{2}{1 + e^{-s_i w_i}}, \end{aligned} \tag{11}$$

where s_i and w_i denote the sign and weight respectively, whose values are determined by comparing the output of the voting classifier F with the output of the weak classifier c_i. In particular, if c_i correctly recognizes the target object, then s_i is set to be 1, otherwise, it is set to be -1. The weight w_i is then set to the margin accordingly indicating "goodness" of a positive weak classifier or "badness" of a negative weak classifier, as described in Algorithm-2. Note that the range of g_i is a nonnegative real interval instead of the nonnegative integers in the conventional multinomial distribution. The scale of g_i does not matter due to the normalization in Eq. 10.

Parameter α is initialized as a small positive number in the first frame. In following frames, if the distribution of the weight vector is stationary, then the value of the concentration parameter α should be accumulated with observation growth and we have $\alpha' = \alpha + \sum_{i=1}^{N} g_i$. However, as explained in Section 1, this distribution is non-stationary and previous accumulation of observations does not increase the confidence of the current estimate of the weight vector, therefore such an update is improper. For simplicity, our method first updates H using Eq. 10 and then performs a maximum-likelihood estimate of α based on our observations $\{g_1, g_2, \cdots, g_N\}$ of the current time step, i.e., to maximize the likelihood function $p(\{g_1, g_2, \cdots, g_N\}|\alpha)$. It is well known that a closed-form estimate is not available for this Dirichlet-multinomial likelihood, hence, an iterative approximation is commonly used. We use an efficient fixed point iteration method [13] to estimate it. The details of the Dirichlet update process are summarized in Algorithm-2.

We now describe issues about the update of weak classifiers. Once a new set of positive and negative samples is identified, a tracking system should decide whether or not to use it to update the classifier(s). This is commonly controlled by a learning rate parameter, which depends on both

```
ALGORITHM-2 DIRICHLET UPDATE
Given Dirichlet distribution $D \sim Dir(\alpha^{t-1}, H^{t-1})$, classification results of $F$ and each $c_1, \cdots, c_N$, detected object $\mathbf{x}_L$
For each $i = 1, \cdots, N$,
    • Step 1: compute the sign $s_i$ for each weak learner $c_i$,
      $s_i = \begin{cases} 1 & c_i(\mathbf{x}_L) = 1 \\ -1 & c_i(\mathbf{x}_L) = 0 \end{cases}$
    • Step 2: compute weight $w_i$,
      $w_i = \begin{cases} \sharp \text{ of consistent negative outputs,} & s_i = 1 \\ \sharp \text{ of inconsistent positive outputs,} & s_i = -1 \end{cases}$ .
End
Step 3: update Dirichlet base distribution $H$ via Eq. 10, where $g_i$ is given by Eq. 11
Step 4: update Dirichlet concentration parameter $\alpha$ by
$\alpha' = arg \max_{\alpha} p(\{g_{1:N} | \alpha\})$
```

the rate of appearance changes and the possible occlusion of the object. In our method, the normalized base distribution H characterizes the expected “relative reliability” of the weak classifiers. After multiplying H with the concentration parameter α, which characterizes our confidence in H, an “expected performance state” of each weak classifier could be obtained, i.e. $\{\alpha h_i\}$. Comparing it with 1 (the neutral value of the performance measure), we can decide whether a weak classifier is better than a random guess, i.e., “good.” By default, when the proportion of “good” weak classifiers is less than 50%, our system decides not to update the weak classifiers because the detected object is very likely occluded. This design turned out to be effective in helping our tracker recover from long-term full occlusions in our experiments.

4. Experiments

We tested our tracker on 28 video sequences, 27 of which are publicly available. We used 11 of 12 sequences from Babenko, et al. [3][1], the full dataset from Santner, et al. [19], the full VTD dataset [12], and “ETH” from the “Linthescher” sequence[2]. “Walking” is our own collected sequence. Our code and dataset are available.[3] The first two datasets assume a fixed scale of the target, while the remaining sequences show large scale variations. The rationale for selecting these test sequences is follows. Firstly, nearly all of these videos are widely used as benchmark sequences in the recent literature, and this allows us to make a fair assessment in comparison to state-of-the-art algorithms. Secondly, the sequences present a diverse set of challenges, including complicated motion, illumination changes, motion blur, a moving camera, cluttered backgrounds, the presence of similar objects, partial or full occlusions, etc.

4.1. Implementation Details

In our object representation, the object bounding box is divided into a regular grid of 8×8 small patches. Therefore, the size of the weak classifier pool depends on the size of the bounding box given in the first frame. For each small patch, our method extracts its 64-bins HSV color/grayscale histogram and standard histogram of gradients (HOG) [6], which yields a 100-dimensional descriptor. Our method also selects larger patches that cover different portions of the object. Similar to the pyramid representation of [9], our method divides the bounding box into 2×2 and 4×4 evenly spaced regions. Including the entire bounding box itself, 21 additional weak classifiers are produced in these three scales. The descriptor for each large patch is the concatenation of descriptors from the small patches it covers.

Each weak classifier corresponding to the local patch is a standard linear SVM, which is trained with its own buffer of 50 positive and 50 negative examples. The buffers and the weak classifiers are initialized with the ground truth bounding box and its shifted versions in the first frame. During tracking, whenever a new example is added to the buffer, the weak classifier is retrained.

The threshold parameter explained in Sec. 3.2 that controls the learning rate of the classifier was set to be 0.5 by default. This parameter was set to 0.6 on the VTD dataset, because the dataset presents fast appearance variations. A more conservative value of 0.4 was used for the “ETH” and “lemming” sequences, where we observed long-term complete occlusions. In general, values between $0.4 \sim 0.6$ offer robust tracking performance. The value is fixed over the whole sequence. Dynamic adaptation of this parameter during tracking is an interesting topic for future work.

Given an incoming frame, our method searches for the target of interest in a standard sliding-window fashion. For efficiency, our method only searches in the neighborhood around the bounding box predicted in the previous frame. The gating radius is proportional to the size of the bounding box. We set the ratio to be 1 on fixed-scale datasets and 0.5 on varying-scale datasets. Our method also searches three neighboring scales with scale step ± 1.2. Note that the overall performance of all published tracking-by-detection methods is sensitive to their gating parameters, depending on how fast the object can move.

4.2. Evaluation Protocol

For a quantitative evaluation of our method, we compared with eight leading algorithms that fall into three broad categories: i.) complete tracking systems, which include Visual Tracker Decomposition (VTD) [12], Tracking-Learning-Detection (TLD) [11] and Parallel Robust Online Simple Tracking (PROST) [19]; ii.) simple trackers that focus more on object representation, which include Compressive Tracker (CT) [21], Distribution Field (DF) [20] and

[1] We excluded “cliffbar” because the ground truth is not tight around the object, and for many frames it only covers a portion of the object.

[2] http://www.vision.ee.ethz.ch/ aess/dataset/

[3] http://www.cs.bu.edu/groups/ivc/software/RET/

Fragments-based tracker (Frag) [1]; iii.) online-learning based trackers, which include the Multiple Instance Learning based tracker (MIL) [3] and Structured output tracker (Struck) [10]. It is worth mentioning that these eight methods use a variety of different features and/or object representations, which are chosen to be compatible with the particulars of each overall tracker design; this interdependence makes it difficult to separately evaluate benefits of a particular tracking strategy vs. the features employed.

For detailed analysis, we developed two baseline algorithms. A color/grayscale histogram and HOG feature vector are extracted for each local patch in the grid. The first baseline (SVM) concatenates these features into a combined vector for all local patches within the window and applies a linear SVM to train the appearance model. The second baseline (OB) employs the same object representation and weak classifiers used in our tracker and the ensemble strategy is online boosting [15, 8, 3].

The implementation of our randomized ensember tracker (RET) employs 5000 samples drawn from the Dirichlet distribution. Across all datasets tested, the accuracy did not increase substantially when more than 1000 samples were used. Our current Matlab implementation computes features for an entire image with 5 scales in 2 seconds, and performs the detection and model update in 1 second for each image frame. We also tested a deterministic version of our tracker (DET) that replaces the sampling step by using the mean distribution H directly (i.e. $F(\mathbf{x}) = f_H(\mathbf{x})$ instead of Eq. 7), and updates the model without the control of a learning rate; thus, DET is a deterministic approximation of our RET algorithm.

In quantitative evaluation of tracker performance, we used three widely accepted evaluation metrics from tracking literature [11, 20]: the successful tracking rate (TA), the average center location errors (ACLE), and the average bounding box overlap ratio (AOR) according to the Pascal VOC criteria. We considered TA and AOR, with ideal values equal to 1, as more informative metrics than ACLE, because when the tracker drifts the ACLE score can grow arbitrarily large. When highlighting the equivalent top performances on each testing sequence, we only allowed a 1% difference in TA.

4.3. Experimental Results

The quantitative results of our comparative experiments are reported in Tables 2 and 3 . The results are grouped according to the assumptions made on the scale of the object in the benchmark video sequences tested. Table 2 reports results on fixed-scale sequences, whereas Table 3 reports results for varying scale sequences. Detecting and tracking varying-scale objects is more challenging, and most competing algorithms do not consider scale variation in their current implementations. All randomized algorithms were evaluated by taking the average over five runs. Our randomized ensemble tracker showed top/equivalently top performance on 14 out of 28 sequences. Our tracker attained high accuracy and robustness across diverse sequences; this is particularly good, considering that our method does not rely on motion prediction.

The superior performance of the baseline linear SVM on certain sequences suggests that representing the object holistically with a high-dimensional feature is good enough for scenarios where there are few distractors or background clutter, and the object is less likely to be occluded. The HOG feature itself also contributes because it is less sensitive to the spatial alignment. However, we witnessed that this baseline almost failed every time when part of the object experienced a fast change or a misleading distractor appeared, such as in the sequences, “girl,” “sylv,” “liquor,” etc. By using a fine-grained representation and identifying the most discriminative local patches, our tracker is less likely to be affected by local drastic changes.

The comparison between our method and online boosting (OB) suggests the advantage of our learning strategy. The online boosting tracker evaluates the weak classifier by its error rate on training examples; such estimation makes sense only if the training data is generated from a fixed joint distribution and the label for the training data is given for sure. As a result, when there are examples with confusing labels because the appearance looks similar to “distractors” or is polluted by occlusion, such as in the “face,” “board,” “liquor” sequences in the PROST dataset and many others from VTD dataset, the error tends to have larger impact on model update. In contrast with OB, our method evaluates the performance of a weak classifier based upon its consistency, a completely different strategy, and the strong classifier is updated implicitly by Bayesian filtering the weighting vector (“hidden state”) smoothly. Therefore, our tracker is less vulnerable to a time-varying joint distribution.

The randomized and deterministic variants of our ensemble trackers (RET, DET) are roughly comparable. Using the mean distribution as weights already improves the performance over OB in many sequences, and it is more efficient than sampling. However, we found that it is still less accurate than the randomized version, when the appearance of the object changes fast or undergoes a severe partial occlusion, such as in “skating2,” “ETH” and “walking”. In addition, we would like to point out that the superior performance of VTD algorithm on its own dataset partially lies in the fact that it has a strong motion dynamics model and carefully sets the gating parameters. Using their scale search parameters only, we were able to improve the alignment of predicted bounding box significantly with DET on several sequences, which we listed as DET$*$ in Table 3 for readers to better understand the data.

For insight into why our method is effective in challeng-

Table 2. Tracking performance on datasets with fixed scale objects. Each entry in the table reports the ACLE and TA performance measure as ACLE (TA). Underlined numbers are from the authors' original papers or computed from the results provided by the authors (PROST did not make their code publicly available). The baseline methods are linear SVM (SVM) and online boosting (OB). RET and DET are the randomized and deterministic variants of our ensemble tracker formulation. Results shown in red suggest comparable top performance. The implementation of DF does not consider color so it does not work well on the last four sequences.

	TLD [11]	PROST [19]	CT [21]	DF [20]	Frag [1]	MIL [3]	Struck [10]	SVM (Baseline 1)	OB (Baseline 2)	DET (Ours)	RET (Ours)
Coke	11 (.68)	-	16 (.30)	7 (.76)	61 (.06)	21 (.21)	7 (.76)	12 (.24)	20 (.12)	14 (.22)	13 (.23)
David	4 (1)	15 (.80)	16 (.89)	10 (1)	46 (.47)	23 (.60)	7 (.98)	4 (1)	11 (1)	7 (1)	6 (1)
Dollar	6 (1)	-	20 (.92)	5 (1)	33 (.66)	15 (.93)	14 (1)	5 (1)	7 (1)	5 (1)	4 (1)
Face1	15 (.99)	7 (1)	19 (.89)	5 (1)	7 (1)	27 (.78)	9 (1)	7 (1)	24 (.81)	8 (.99)	7 (1)
Face2	13 (.97)	17 (.82)	10 (1)	11 (.99)	45 (.48)	20 (.82)	7 (.98)	7 (1)	26 (.60)	10 (1)	9 (1)
Girl	18 (.93)	19 (.89)	21 (.78)	22 (.73)	27 (.70)	32 (.56)	10 (1)	56 (.26)	25 (.89)	34 (.72)	19 (.84)
Sylv	6 (.97)	11 (.67)	9 (.75)	16 (.67)	11 (.73)	11 (.74)	10 (.87)	22 (.60)	8 (.88)	10 (.82)	12 (.80)
Tiger1	6 (.89)	7 (.79)	10 (.78)	7 (.89)	20 (.40)	15 (.57)	7 (.85)	5 (.97)	34 (.35)	4 (.97)	4 (.92)
Tiger2	29 (.26)	-	13 (.60)	7 (.82)	39 (.09)	17 (.63)	12 (.60)	5 (.90)	6 (.86)	4 (.96)	4 (.96)
Twinings	16 (.52)	-	9 (.89)	11 (.77)	15 (.69)	10 (.85)	7 (.98)	24 (.45)	28 (.43)	15 (.57)	21 (.63)
Surfer	4 (.97)	-	19 (.13)	5 (.95)	139 (.20)	9 (.76)	8 (.74)	3 (.97)	3 (.99)	3 (.99)	3 (.99)
Board	11 (.87)	39 (.75)	62(.53)	-	90 (.68)	51 (.68)	37 (.78)	59 (.70)	244 (.11)	39 (.84)	38 (.86)
Box	17 (.92)	13 (.91)	14 (.89)	-	57 (.61)	105 (.25)	140 (.37)	106 (.40)	13 (.90)	13 (.96)	10 (.97)
Lemming	16 (.86)	25 (.71)	63 (.31)	-	83 (.55)	15 (84)	31 (.69)	82 (.46)	88 (.26)	80 (.47)	16 (.82)
Liquor	7 (.92)	22 (.85)	180 (.21)	-	31 (.80)	165 (21)	74 (.60)	82 (.52)	26 (.24)	13 (.95)	13 (.96)

Table 3. Tracking performance (AOR (TA)) on datasets with varying, sometimes significant changes in object scales. Among the few algorithms available, we chose VTD [12] and TLD [11] as representative competing algorithms. Underlined numbers were given by the authors. Results shown in red suggest comparable top performance. We were not able to fully evaluate TLD because it fails quickly on certain sequences.

	VTD [12]	TLD [11]	SVM (B1)	OB (B2)	DET (Ours)	DET* (Ours)	RET (Ours)
Animal	.65 (.92)	.48 (.76)	.73 (1)	.62 (.94)	.72 (1)	.7 (1)	.72 (1)
Basketball	.72 (.98)	-	.43 (.36)	.51 (.50)	.53 (.63)	.62 (.92)	.54 (.64)
Football	.66 (.78)	.55 (.77)	.56 (.78)	.69 (.93)	.61 (.74)	.66 (.96)	.62 (.82)
Shaking	.75 (.99)	.12 (.16)	.20 (.21)	.03 (.04)	.55 (.64)	.60 (.80)	.44 (.53)
Singer1a	.82 (1)	.66 (.93)	.70 (.98)	.46 (.37)	.70 (.90)	.70 (.89)	.73 (.97)
Singer1b	.59 (.63)	.11 (.10)	.20 (.12)	.20 (.12)	.70 (.89)	.70 (.93)	.69 (.93)
Singer2	.74 (.97)	-	.29 (.23)	.69 (.93)	.07 (.06)	.08 (.06)	.38 (.50)
Skating1a	.68 (.92)	.39 (.43)	.48 (.39)	.48 (.38)	.56 (.55)	.58 (.64)	.48 (.52)
Skating1b	.67 (.90)	.42 (.58)	.34 (.42)	.44 (.27)	.43 (.45)	.54 (.58)	.46 (.52)
Skating2	.57 (.68)	-	.54 (.63)	.40 (.39)	.45 (.48)	.55 (.71)	.61 (.75)
Soccer	.39 (.32)	-	.15 (.17)	.34 (.25)	.12 (.14)	.40 (.35)	.27 (.30)
ETH	.34 (.31)	.51 (.63)	.56 (.62)	.57 (.61)	.50 (.39)	-	.65 (.92)
walking	.33 (.22)	.19 (.20)	.59 (.67)	.28 (.08)	.54 (.68)	-	.77 (1)

ing conditions, we took snapshots of our learned ensemble classifier and show in Fig. 3 the base distribution H of the Dirichlet distribution, which characterizes the relative importance of each weak classifier. Being more important means accumulatively higher frequency of agreement with the strong classifier, as indicated by Eq. 10 and Eq. 11.

In Fig. 3(a), the bounding box is not tight around the circuit board object, so patches in the box corners are actually background. After sequential learning over a few frames, our method is able to identify the background regions within the bounding box since they do not consistently contribute to the detection of the object. As a result, their weights are lowered, which is reflected in the base distribution.

In Fig. 3(b), an occlusion of the face causes the weak classifiers that account for the occluded region of the face to disagree with the strong classifier. This disagreement makes them less important and reduces the weights. A similar situation happens in Fig. 3(c) where the black box is occluded.

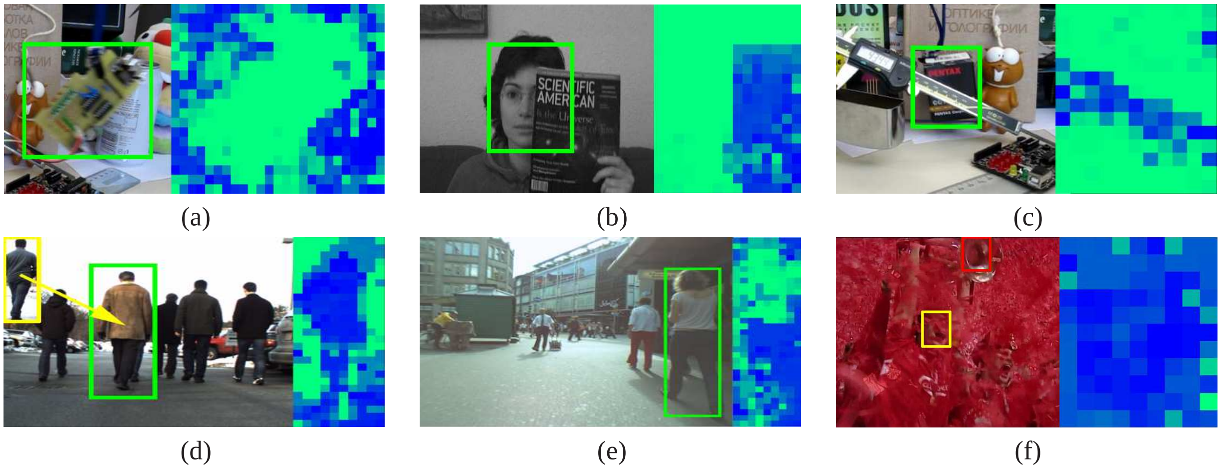
In Fig. 3(d), a pedestrian (shown on the top left corner) is completely occluded by a distracter (man with beige jacket), so the majority of the weak classifiers disagrees with the strong classifier and the weights for the weak classifiers in the corner of the bounding box are strengthened. This base distribution suggests that our model is not polluted by the distracter, and our method is able to re-identify the target in a later frame once it is visible.

In Fig. 3(e), although the pedestrian is not occluded, the bottom half of her body looks similar to the nearby shadow. Therefore, the corresponding weights are successfully reduced to avoid the confusion.

In Fig. 3(f), we also show a representative failure case, where our classifier produces false detection (red). The distracter in the background looks extremely similar to the object. In this situation, identifying the correct object is difficult. We also witnessed a large variance of our RET tracker on "singer2" and "shaking" when non-smooth environment changes occur all the time. Although we could easily handle these cases by using some spatial information and a motion model in consecutive frames, we preferred not to do so in our reported experiments in order to focus on the evaluation of the detection strength of our formulation.

5. Discussion

We proposed a tracker that exploits a novel online randomized classifier ensemble method that naturally evolves the classifier in a Bayesian manner. Instead of trying to



(a) (b) (c)

(d) (e) (f)

Figure 3. Sample images with true detections (green), false alarms (red), and ground truth (yellow) and snapshots of base distribution H of Dirichlet distribution (greener means higher weight of the associated weak classifier and its higher discriminate ability, bluer means lower weight). A detailed discussion of this figure is in the text.

compute deterministic optimal weights for the weak classifiers, we characterize their uncertainty by introducing the Dirichlet distribution, and draw random samples to form a randomized voting classifier. Our randomized ensemble tracker was tested in experiments on numerous tracking sequences, demonstating the robustness of our method compared to state-of-the-art approaches, even without motion prediction.

Our framework is flexible, since our learning strategy does not restrict the type of weak classifier that can be used. In future work, we are interested in building a larger pool of weak classifiers and experimenting with different features. For visual tracking, we will explore the integration of a strong motion prediction model [10]. Since our method is general, we plan to apply it to other tasks where online classifier ensembles are used.

Acknowledgments. This material is based on work supported by the US National Science Foundation under Grant Nos. 0910908 and 0855065 and the US Army under TARDEC STTR Contract No. W56HZV-12-C-0049.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006. 2, 6, 7

[2] S. Avidan. Ensemble tracking. *PAMI*, 29, 2007. 1, 2

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. *CVPR*, 2009. 1, 2, 5, 6, 7

[4] L. Cehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *ICCV*, 2011. 2

[5] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27, 2005. 2

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 5

[7] J. Fan, X. Shen, and Y. Wu. Scribble tracker: A matting-based approach for robust tracking. *PAMI*, 34, 2012. 2

[8] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, 2006. 1, 2, 6

[9] K. Grauman. *Matching Sets of Features for Efficient Retrieval and Recognition*. PhD thesis, MIT, USA, 2006. 5

[10] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 6, 7, 8

[11] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 34(7), 2012. 5, 6, 7

[12] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, 2010. 5, 7

[13] T. P. Minka. Estimating a Dirichlet distribution. Technical report, Microsoft Research, 2003. 4

[14] K. Ng, G. Tian, and M. Tang. *Dirichlet and Related Distributions: Theory, Methods and Applications*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2011. 3

[15] N. Oza. Online bagging and boosting. In *IEEE Intl' conf. on Systems, man and cybernetics*, pages 2340–2345, 2005. 2, 6

[16] F. Pernici. Facehugger: The ALIEN tracker applied to faces. In *ECCV Workshops and Demonstrations*. 2012. 2

[17] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online multi-class LPBoost. In *CVPR*, 2010. 1, 2

[18] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCV*, 2009. 2

[19] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *CVPR*, 2010. 5, 7

[20] L. Sevilla-Lara and E. G. Learned-Miller. Distribution fields for tracking. In *CVPR*, 2012. 1, 2, 5, 6, 7

[21] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, 2012. 1, 2, 5, 7