# Robust Object Tracking with Online Multiple Instance Learning

Boris Babenko, *Student Member, IEEE,* Ming-Hsuan Yang, *Senior Member, IEEE* and Serge Belongie, *Member, IEEE*

### Abstract

In this paper we address the problem of tracking an object in a video given its location in the first frame and no other information. Recently, a class of tracking techniques called "tracking by detection" has been shown to give promising results at real-time speeds. These methods train a discriminative classifier in an online manner to separate the object from the background. This classifier bootstraps itself by using the current tracker state to extract positive and negative examples from the current frame. Slight inaccuracies in the tracker can therefore lead to incorrectly labeled training examples, which degrade the classifier and can cause drift. In this paper we show that using Multiple Instance Learning (MIL), instead of traditional supervised learning, avoids these problems and can therefore lead to a more robust tracker with fewer parameter tweaks. We propose a novel online MIL algorithm for object tracking that achieves superior results with real-time performance. We present thorough experimental results (both qualitative and quantitative) on a number of challenging video clips.

### Index Terms

Visual Tracking, Multiple Instance Learning, Online Boosting

- *B. Babenko and S. Belongie are with the Department of Computer Science and Engineering, University of California, San Diego.*
- *M.-H. Yang is with the Department of Computer Science, University of California, Merced.*

# Robust Object Tracking with Online Multiple Instance Learning

## 1 Introduction

Object tracking is a well studied problem in computer vision and has many practical applications. The problem and its difficulty depend on several factors, such as the amount of prior knowledge about the target object and the number and type of parameters being tracked (e.g. location, scale, detailed contour). Although there has been some success with building trackers for specific object classes (e.g. faces [1], humans [2], mice [3], rigid objects [4]), tracking generic objects has remained challenging because an object can drastically change appearance when deforming, rotating out of plane, or when the illumination of the scene changes.

A typical tracking system consists of three components: (1) an appearance model, which can evaluate the likelihood that the object of interest is at some particular location; (2) a motion model, which relates the locations of the object over time; and (3) a search strategy for finding finding the most likely location in the current frame. The contributions of this paper deal with the first of these three components; we refer the reader to [5] for a thorough review of the other components. Although many tracking methods employ static appearance models that are either defined manually or trained using only the first frame [2], [4], [6], [7], [8], [9], these methods are often unable to cope with significant appearance changes. These challenges are particularly difficult when there is limited a priori knowledge about the object of interest. In this scenario, it has been shown that an adaptive appearance model, which evolves during the tracking process as the appearance of the object changes, is the key to good performance [10], [11], [12]. Training adaptive appearance models, however, is itself a difficult task with many questions yet to be answered. Such models often involve many parameters that must be tuned to get good performance (e.g. "forgetting factors" that control how fast the appearance model can change), and can suffer from drift problems when an object undergoes partial occlusion.

In this paper we focus on the problem of tracking an arbitrary object with no prior knowledge other than its location in the first video frame (sometimes referred to as "model-free" tracking). Our goal is to develop a more robust way of updating an adaptive appearance model; we would like our system to be able to handle partial occlusions without significant drift, and for it to work well with minimal parameter tuning. To do this, we turn to a discriminative learning paradigm called Multiple Instance Learning (MIL) [13] that can handle ambiguities in the training data. This technique has found recent success in other computer vision areas, such as object detection [14], [15] and object recognition [16], [17], [18].

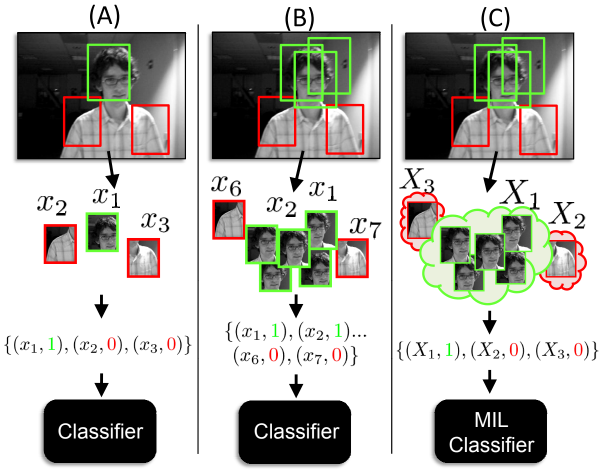


Fig. 1. **Updating a discriminative appearance model:** (A) Using a single positive image patch to update a traditional discriminative classifier. The positive image patch chosen does not capture the object perfectly. (B) Using several positive image patches to update a traditional discriminative classifier. This can make it difficult for the classifier to learn a tight decision boundary. (C) Using one positive bag consisting of several image patches to update a MIL classifier. See Section 4 for empirical results of these three strategies.

We will focus on the problem of tracking the location and scale of a single object, using a rectangular bounding box to approximate these parameters. It is plausible that the ideas presented here can be applied to other types of tracking problems like tracking multiple objects (e.g. [19]), tracking contours (e.g. [20], [21]), or tracking deformable objects (e.g. [22]), but this is outside the scope of our work.

The remainder of this paper is organized as follows: in Section 2 we review the current state of the art in adaptive appearance models; in Section 3 we introduce our tracking algorithm; in Section 4 we present qualitative and quantitative results of our tracker on a number of challenging video clips. We conclude in Section 5.

## 2 Adaptive Appearance Models

An important choice in the design of appearance models is whether to model only the object [12], [23], or both the object and the background [24], [25], [26], [27], [28], [29], [30]. Many of the latter approaches have shown that training a model to separate the object from the background via a discriminative classifier can often achieve superior results. These methods are closely related to object detection – an area that has seen great progress in the last decade. In fact, some of these methods are referred to as "tracking-by-detection" or "tracking by repeated recognition" [31]. In particular, the recent advances in face detection [32] have inspired some successful real-time tracking algorithms [25], [26].

A major challenge that is often not discussed in the literature is how to choose positive and negative examples when updating the adaptive appearance model. Most commonly this is done by taking the current tracker location as one positive example, and sampling the neighborhood around the tracker location for negatives. If the tracker location is not precise, however, the appearance model ends up getting updated with a sub-optimal positive example. Over time this can degrade the model, and can cause drift. On the other hand, if multiple positive examples are used (taken from a small neighborhood around the current tracker location), the model can become confused and its discriminative power can suffer (cf. Fig. 1 (A-B)). Alternatively, Grabner et al. [33] recently proposed a semi-supervised approach where labeled examples come from the first frame only, and subsequent training examples are left unlabeled. This method is particularly well suited for scenarios where the object leaves the field of view completely, but it throws away a lot of useful information by not taking advantage of the problem domain (e.g., it is safe to assume small interframe motion).

Object detection faces issues similar to those described above, in that it is difficult for a human labeler to be consistent with respect to how the positive examples are cropped. In fact, Viola et al. [14] argue that object detection has inherent ambiguities that cause difficulty for traditional supervised learning methods. For this reason they suggest the use of a Multiple Instance Learning (MIL) [13] approach for object detection. We give a more formal definition of MIL in Section 3.2, but the basic idea of this learning paradigm is that during training, examples are presented in sets (often called "bags"), and labels are provided for the bags rather than individual instances. If a bag is labeled positive it is assumed to contain at least one positive instance, otherwise the bag is negative. For example, in the context of object detection, a positive bag could contain a few possible bounding boxes around each labeled object (e.g. a human labeler clicks on the center of the object, and the algorithm crops several rectangles around that point). Therefore, the ambiguity is passed on to the learning algorithm, which now has to figure out which instance in each positive bag is the most "correct". Although one could argue that this learning problem is more difficult in the sense that less information is provided to the learner, in some ways it is actually easier because the learner is allowed some flexibility in finding a decision boundary. Viola et al. present convincing results showing that a face detector trained with weaker labeling (just the center of the face) and a MIL algorithm outperforms a state of the art supervised algorithm trained with explicit bounding boxes.

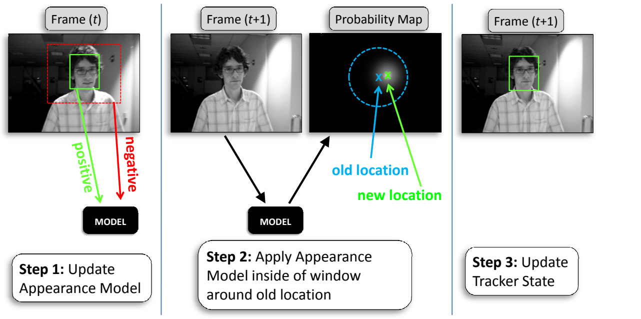

Frame (t)
positive
negative
MODEL
Step 1: Update Appearance Model
Frame (t+1)
Probability Map
old location
new location
MODEL
Step 2: Apply Appearance Model inside of window around old location
Frame (t+1)
Step 3: Update Tracker State

Fig. 2. **Tracking by detection with a greedy motion model:** an illustration of how most tracking by detection systems work.

**Algorithm 1** MILtrack

```
Input: Video frame number $k$
  Crop out a set of image patches, $X^s = \{x : ||\ell(x) - \ell^*_{t-1}|| < s\}$ and compute feature vectors.
  Use MIL classifier to estimate $p(y = 1|x)$ for $x \in X^s$.
  Update tracker location $\ell^*_t = \ell\Big(\text{argmax}_{x \in X^s}\, p(y|x)\Big)$.
  Crop out two sets of image patches $X^r = \{x : ||\ell(x) - \ell^*_t|| < r\}$ and $X^{r,\beta} = \{x : r < ||\ell(x) - \ell^*_t|| < \beta\}$.
  Update MIL appearance model with one positive bag $X^r$ and $|X^{r,\beta}|$ negative bags, each containing a single image patch from the set $X^{r,\beta}$.
```

In this paper we make an analogous argument to that of Viola et al. [14], and propose to use a MIL based appearance model for object tracking (cf. Fig. 1(C)). In fact, in the object tracking domain there is even more ambiguity than in object detection because the tracker has no human input and has to bootstrap itself. Therefore, we expect the benefits of a MIL approach to be even more significant than in the object detection problem. In order to incorporate MIL into a tracker, an online MIL algorithm is required. The algorithm we propose (to our knowledge this is the first online MIL algorithm in the literature) is based on boosting and is related to the MILBoost algorithm [14] as well as the Online AdaBoost algorithm [34]. We present empirical results on challenging video sequences, which show that using an online MIL based appearance model can lead to more robust and stable tracking than existing methods in the literature.

## 3 Tracking with Online MIL

In this section we introduce our tracking algorithm, MILTrack, which uses a MIL based appearance model. We begin with an overview of our tracking system which includes a description of the motion model we use. Next we review the MIL problem and briefly describe the MILBoost algorithm [14]. We then review online boosting [25], [34] and present a novel boosting based algorithm for online MIL. Finally, we review various implementation details.

### 3.1 System Overview and Motion Model

The basic flow of the tracking system we implemented in this work is illustrated in Fig. 2 and summarized in Algorithm 1. Our image representation consists of a set of Haar-like features that are computed for each image patch [32], [35]; this is discussed in more detail in Section 3.6. The appearance model is composed of a discriminative classifier which is able to return $p(y=1|x)$ (we will use $p(y|x)$ as shorthand), where $x$ is an image patch (or the representation of an image patch in feature space) and $y$ is a binary variable indicating the presence of the object of interest in that image patch. At every time step $t$, our tracker maintains the object location $\ell_t^*$. Let $\ell(x)$ denote the location of image patch $x$ (for now let's assume this consists of only the $(x,y)$ coordinates of the patch center, and that scale is fixed; below we consider tracking scale as well). For each new frame we crop out a set of image patches $X^s = \{x : ||\ell(x) - \ell_{t-1}^*|| < s\}$ that are within some search radius $s$ of the current tracker location, and compute $p(y|x)$ for all $x \in X^s$. We then use a greedy strategy to update the tracker location:

$$\ell_t^* = \ell\Big(\operatorname*{argmax}_{x\in X^s} p(y|x)\Big) \tag{1}$$

In other words, we do not maintain a distribution of the target's location at every frame, and our motion model is such that the location of the tracker at time $t$ is equally likely to appear within a radius $s$ of the tracker location at time $(t-1)$:

$$p(\ell_t^*|\ell_{t-1}^*) \propto \begin{cases} 1 & \text{if } ||\ell_t^* - \ell_{t-1}^*|| < s \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This could be extended with something more sophisticated, such as a particle filter, as is done in [12], [29], [36]; however, we again emphasize that our focus is on the appearance model.

Once the tracker location is updated, we proceed to update the appearance model. We crop out a set of patches $X^r = \{x : ||\ell(x) - \ell_t^*|| < r\}$, where $r < s$ is a scalar radius (measured in pixels), and label this bag positive (recall that in MIL we train the algorithm with labeled bags). In contrast, if a standard learning algorithm were used, there would be two options: set $r = 1$ and use this as a single positive instance, or set $r > 1$ and label all these instances positive. For negatives we crop out patches from an annular region $X^{r,\beta} = \{x : r < ||\ell(x) - \ell_t^*|| < \beta\}$, where $r$ is same as before, and $\beta$ is another scalar. Since this generates a potentially large set, we then take a random subset of these image patches and label them negative. We place each negative example into its own negative bag, though placing them all into one negative bag yields the same result (this is discussed in more detail in Section 3.2).

Incorporating scale tracking into this system is straightforward. First, we define an extra parameter $\lambda$ to be the scale space step size. When searching for the location of the object in a new frame, we crop out image patches from the image at the current scale, $\ell_t^s$, as well as one scale step larger and smaller, $\ell_t^s \pm \lambda$; once we find the location with the maximum response, we update the current state (both position and scale) accordingly. When updating the appearance model, we have the option of cropping training image patches only from the current scale, or from the neighboring scales as well; in our current implementation we do the former.

It is important to note that tracking in scale-space is a double-edged sword. In some ways the problem becomes more difficult because the parameter space becomes larger, and consequently there is more room for error. However, tracking this additional parameter may mean that the image patches we crop out are better aligned, making it easier for our classifier to learn the correct appearance. In our experiments we have noticed both behaviors – sometimes adding scale tracking helps, and other times it hurts performance.

Details on how all of the above parameters were set are in Section 4, although we use the *same* parameters throughout *all* the experiments. We continue with a more detailed review of MIL.

### 3.2 Multiple Instance Learning

Traditional discriminative learning algorithms for training a binary classifier that estimates $p(y|x)$ require a training data set of the form $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ where $x_i$ is an instance (in our case a feature vector computed for an image patch), and $y_i \in \{0,1\}$ is a binary label. In Multiple Instance Learning training data has the form $\{(X_1,y_1),\ldots,(X_n,y_n)\}$ where a bag $X_i = \{x_{i1},\ldots,x_{im}\}$ and $y_i$ is a bag label. The bag labels are defined as:

$$y_i = \max_j(y_{ij}) \tag{3}$$

where $y_{ij}$ are the instance labels, which are not known during training. In other words, a bag is considered positive if it contains at least one positive instance. Numerous algorithms have been proposed for solving the MIL problem [13], [14], [16]. The algorithm that is most closely related to our work is the MILBoost algorithm proposed by Viola et al. in [14]. MILBoost uses the the gradient boosting framework [37] to train a boosting classifier that maximizes the log likelihood of bags:

$$\mathcal{L} = \sum_i \Big(\log p(y_i|X_i)\Big) \tag{4}$$

Notice that the likelihood is defined over bags and not instances, because instance labels are unknown during training, and yet the goal is to train an instance classifier that estimates $p(y|x)$. We therefore need to express $p(y_i|X_i)$, the probability of a bag being positive, in terms of its instances. In [14] the Noisy-OR (NOR) model is adopted for doing this:

$$p(y_i|X_i) = 1 - \prod_j \Big(1 - p(y_i|x_{ij})\Big) \tag{5}$$

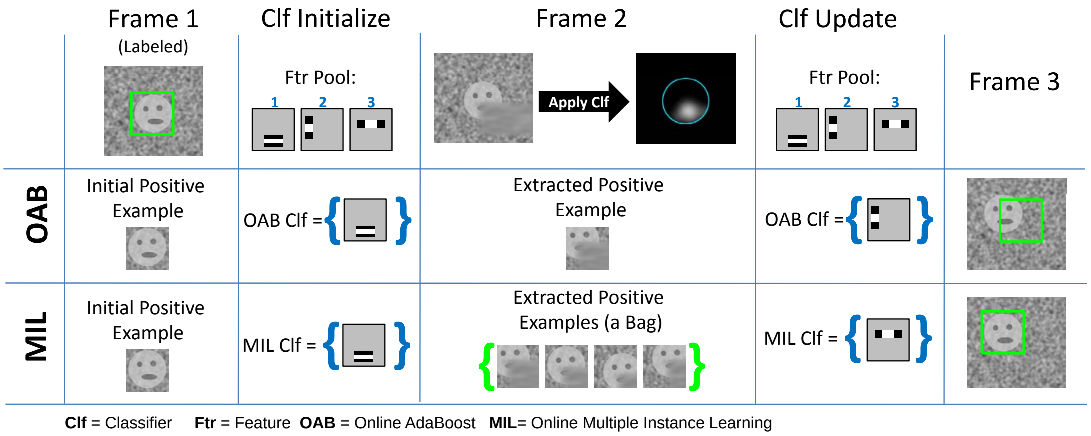


Fig. 3. An illustration of how using MIL for tracking can deal with occlusions. **Frame 1:** Consider a simple case where the classifier is allowed to only pick one feature from the pool. The first frame is labeled. One positive patch and several negative patches (not shown) are extracted, and the classifiers are initialized. Both OAB and MIL result in identical classifiers - both choose feature #1 because it responds well with the mouth of the face (feature #3 would have performed well also, but suppose #1 is slightly better). **Frame 2:** In the second frame there is some occlusion. In particular, the mouth is occluded, and the classifier trained in the previous step does not perform well. Thus, the most probable image patch is no longer centered on the object. OAB uses just this patch to update; MIL uses this patch along with its neighbors. Note that MIL includes the correct image patch in the positive bag. **Frame 3:** When updating, the classifiers try to pick the feature that best discriminates the current example as well the ones previously seen. OAB has trouble with this because the current and previous positive examples are too different. It chooses a bad feature. MIL is able to pick the feature that discriminates the eyes of the face, because one of the examples in the positive bag was correctly cropped (even though the mouth was occluded). MIL is therefore able to successfully classify future frames. Note that if we assign positive labels to the image patches in the MIL bag and use these to train OAB, it would still have trouble picking a good feature.

although other models could be swapped in (e.g. [38]). The equation above has the desired property that if one of the instances in a bag has a high probability, the bag probability will be high as well. As mentioned in [14], with this formulation, the likelihood is the same whether we put all the negative instances in one bag, or if we put each in its own bag. Intuitively this makes sense because no matter how we arrange things, we know that every instance in a negative bag is negative. We refer the reader to [14] for further details on MILBoost. Finally, we note that MILBoost is a batch algorithm (meaning it needs the entire training data set at once) and cannot be trained in an online manner as we need in our tracking application. Nevertheless, we adopt the loss function in Equation 4 and the bag probability model in Equation 5 when we develop our online MIL algorithm in Section 3.4.

### 3.3 Online Boosting

Our algorithm for online MIL is based on the boosting framework [39] and is related to the work on Online AdaBoost [34] and its adaptation in [25]. The goal of boosting is to combine many weak classifiers $\mathbf{h}(x)$ (usually decision stumps) into an additive strong classifier:

$$\mathbf{H}(x) = \sum_{k=1}^{K} \alpha_k \mathbf{h}_k(x) \tag{6}$$

where $\alpha_k$ are scalar weights. There have been many boosting algorithms proposed to learn this model in batch mode [39], [40]; typically this is done in a greedy manner where the weak classifiers are trained sequentially. After each weak classifier is trained, the training examples are re-weighted such that examples that were previously misclassified receive more weight. If each weak classifier is a decision stump, then it chooses one feature that has the most discriminative power for the entire weighted training set. In this case boosting can be viewed as performing feature selection, choosing a total of $K$ features, which is generally much smaller than the size of the entire feature pool. This has proven particularly useful in computer vision because it creates classifiers that are efficient at run time [32].

In [34], Oza develops an online variant of the popular AdaBoost algorithm [39], which minimizes the exponential loss function. This variant requires that all $\mathbf{h}$ can be trained in an online manner. The basic flow of Oza's algorithm is as follows: for an incoming example $x$, each $\mathbf{h}_k$ is updated sequentially and the weight of example $x$ is adjusted after each update. Since the formulas for the example weights and classifier weights in AdaBoost depend only on the error of the weak classifiers, Oza proposes to keep a running average of the error of each $\mathbf{h}_k$, which allows the algorithm to estimate both the example weight and the classifier weights in an online manner.

In Oza's framework if every $\mathbf{h}$ is restricted to be a decision stump, the algorithm has no way of choosing the most discriminative feature because the entire training set is never available at one time. Therefore, the features for each $\mathbf{h}_k$ must be picked a priori. This is a potential problem for computer vision applications, since they often rely on the feature selection property of boosting. Grabner et al. [25] proposed an extension of Oza's algorithm which performs feature selection by maintaining a pool of $M > K$ candidate weak stump classifiers $h$. When a new example is passed in, all of the candidate weak classifiers are updated in parallel. Then, the algorithm sequentially chooses $K$ weak classifiers

**Algorithm 2** Online MILBoost (OMB)

```
Input: Dataset $\{X_i, y_i\}_{i=1}^N$, where $X_i = \{x_{i1}, x_{i2}, \ldots\}, y_i \in \{0, 1\}$
Update all $M$ weak classifiers in the pool with data $\{x_{ij}, y_i\}$
Initialize $H_{ij} = 0$ for all $i, j$
for $k = 1$ to $K$ do
    for $m = 1$ to $M$ do
        $p_{ij}^m = \sigma\big(H_{ij} + h_m(x_{ij})\big)$
        $p_i^m = 1 - \prod_j \big(1 - p_{ij}^m\big)$
        $\mathcal{L}^m = \sum_i \Big(y_i \log(p_i^m) + (1 - y_i)\log(1 - p_i^m)\Big)$
    end for
    $m^* = \operatorname{argmax}_m \mathcal{L}^m$
    $\mathbf{h}_k(x) \leftarrow h_{m^*}(x)$
    $H_{ij} = H_{ij} + \mathbf{h}_k(x)$
end for
Output: Classifier $\mathbf{H}(x) = \sum_k \mathbf{h}_k(x)$, where $p(y|x) = \sigma\big(\mathbf{H}(x)\big)$
```

from this pool by keeping running averages of errors for each, as in [34], and updates the weights of $\mathbf{h}$ accordingly. We employ a similar feature selection technique in our Online MIL algorithm, although the criteria for choosing weak classifiers is different.

### 3.4 Online Multiple Instance Boosting

The algorithms in [34] and [25] rely on the special properties of the exponential loss function of AdaBoost, and therefore cannot be readily adapted to the MIL problem. We now present our novel online boosting algorithm for MIL. As in [40], we take a statistical view of boosting, where the algorithm is trying to optimize a specific objective function $J$. In this view, the weak classifiers are chosen sequentially to optimize the following criteria:

$$(\mathbf{h}_k, \alpha_k) = \underset{\mathbf{h} \in \mathcal{H}, \alpha}{\operatorname{argmax}}\, J(\mathbf{H}_{k-1} + \alpha \mathbf{h}) \tag{7}$$

where $\mathbf{H}_{k-1}$ is the strong classifier made up of the first $(k-1)$ weak classifiers, and $\mathcal{H}$ is the set of all possible weak classifiers. In batch boosting algorithms, the objective function $J$ is computed over the entire training data set.

In our case, for the current video frame we are given a training data set $\{(X_1, y_1), (X_2, y_2) \ldots\}$, where $X_i = \{x_{i1}, x_{i2} \ldots\}$. We would like to update our classifier to maximize log likelihood of this data (Equation 4). We model the instance probability as

$$p(y|x) = \sigma\big(\mathbf{H}(x)\big) \tag{8}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function; the bag probabilities $p(y|X)$ are modeled using the NOR model in Equation 5. To simplify the problem, we absorb the scalar weights $\alpha_t$ into the weak classifiers, by allowing them to return real values rather than binary.

At all times our algorithm maintains a pool of $M > K$ candidate weak stump classifiers $h$. To update the classifier, we first update all weak classifiers in parallel, similar to [25]. Note that although instances are in bags, the weak classifiers in a MIL algorithm are instance classifiers, and therefore require instance labels $y_{ij}$. Since these are unavailable, we pass in the bag label $y_i$ for all instances $x_{ij}$ to the weak training procedure. We then choose $K$ weak classifiers $\mathbf{h}$ from the candidate pool sequentially, by maximizing the log likelihood of bags:

$$\mathbf{h}_k = \underset{h \in \{h_1, \ldots, h_M\}}{\operatorname{argmax}}\, \mathcal{L}(\mathbf{H}_{k-1} + h) \tag{9}$$

See Algorithm 2 for the pseudo-code of Online MILBoost and Fig. 3 for an illustration of tracking with this algorithm.

### 3.5 Discussion

There are a couple important issues to point out about this algorithm. First, we acknowledge the fact that training the weak classifiers with positive labels for all instances in the positive bags is sub-optimal because some of the instances in the positive bags may actually not be "correct". The algorithm makes up for this when it is choosing the weak classifiers $\mathbf{h}$ based on the *bag* likelihood loss function. We have also experimented using online GradientBoost [41] to compute weights (via the gradient of the loss function) for all instances, but found this to make little difference in accuracy while making the system slower. Second, if we compare Equations 7 and 9 we see that the latter has a much more restricted choice of weak classifiers. This approximation does not seem to degrade the performance of the classifier in practice, as noted in [42]. Finally, we note that the likelihood being optimized in Equation 9 is computed only on the current examples. Thus, it has the potential of overfitting to current examples, and not retaining information about previously seen data. This is averted by using online weak classifiers that do retain information about previously seen data, which balances out the overall algorithm between fitting the current data and retaining history.

### 3.6 Implementation Details

#### 3.6.1 *Weak Classifiers*

Recall that we require weak classifiers $h$ that can be updated online. In our system each weak classifier $h_k$ is composed of a Haar-like feature $f_k$ and four parameters $(\mu_1, \sigma_1, \mu_0, \sigma_0)$ that are estimated online. The classifiers return the log odds ratio:

$$h_k(x) = \log\left[\frac{p_t\big(y = 1|f_k(x)\big)}{p_t\big(y = 0|f_k(x)\big)}\right] \tag{10}$$

where $p_t\big(f_t(x)|y = 1\big) \sim \mathcal{N}(\mu_1, \sigma_1)$ and similarly for $y = 0$. We let $p(y = 1) = p(y = 0)$ and use Bayes rule to compute the above equation. When the weak classifier receives new data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ we use the following update rules:

$$\begin{aligned} \mu_1 &\leftarrow \gamma\mu_1 + (1-\gamma)\frac{1}{n}\sum_{i|y_i=1} f_k(x_i) \\ \sigma_1 &\leftarrow \gamma\sigma_1 + (1-\gamma)\sqrt{\frac{1}{n}\sum_{i|y_i=1}\big(f_k(x_i)-\mu_1\big)^2} \end{aligned}$$

where $0 < \gamma < 1$ is a learning rate parameter. The update rules for $\mu_0$ and $\sigma_0$ are similarly defined.

### 3.6.2 Image Features

We represent each image patch as a vector of Haar-like features [32], which are randomly generated, similar to [35]. Each feature consists of 2 to 4 rectangles, and each rectangle has a real valued weight. The feature value is then a weighted sum of the pixels in all the rectangles. These features can be computed efficiently using the integral image trick described in [32].

## 4 Experiments

We tested our MILTrack system on several challenging video sequences, some of which are publicly available. For comparison, we implemented a tracker based on the Online AdaBoost (OAB) algorithm described in [25]. We plugged this learning algorithm into our system, and used the same features and motion model as for MILTrack (See Section 3.1). We acknowledge the fact that our implementation of the OAB tracker achieves worse performance than is reported in [25]; this could be because we are using simpler features, or because our parameters were not tuned per video sequence. However, our study is still valid for comparison because only the learning algorithm changes between our implementation of the OAB tracker and MILTrack, and everything else is kept constant. This allows us to isolate the appearance model to make sure that it is the cause of the performance difference.

One of the goals of this work is to demonstrate that using MIL results in a more robust and stable tracker. For this reason ***all algorithm parameters were fixed for all the experiments.*** This holds for all algorithms we tested. For MILTrack and OAB the parameters were set as follows. The search radius $s$ is set to $35$ pixels. For MILTrack we sample positives in each frame using a radius $r = 4$ (we found that the algorithm is fairly robust for a range of values). This generates a total of 45 image patches comprising one positive bag (for clarity, we call this MILTrack(45)). For the OAB tracker we tried two variations. In the first variation we set $r = 1$ generating only one positive example per frame (we call this OAB(1)); in the second variation we set $r = 4$ as we do in MILTrack (although in this case each of the 45 image patches is labeled positive); we call this OAB(45). The reason we experimented with these two versions was to show that the superior performance of MILTrack is not simply due to the fact that we extract multiple positive examples per frame. In fact, as we will see shortly, when multiple positive examples are used for the OAB tracker, its performance degrades[1] (cf. Table 1 and Fig. 4). The scalar $\beta$ for sampling negative examples was set to $50$, and we randomly sample $65$ negative image patches from the set $X^{r,\beta}$ (though during initialization with the first frame we sample 1000 patches). The learning rate $\gamma$ for the weak classifiers is set to $0.85$. Finally, the number of candidate weak classifiers $M$ was set to 250, and the number of chosen weak classifiers $K$ was set to 50.

To gauge absolute performance we also compare our results to three other algorithms, using code provided by the respective authors. The first of these is the SemiBoost tracker [33][2]; as mentioned earlier, this method uses label information from the first frame only, and then updates the appearance model via online semi-supervised learning in subsequent frames. This makes it particularly robust to scenarios where the object leaves the scene completely. However, the model relies strongly on the prior classifier (trained using the first frame). We found that on clips exhibiting significant appearance changes this algorithm often lost the object. The second algorithm is FragTrack [9][3]. This algorithm uses a static appearance model based on integral histograms, which have been shown to be very efficient. The appearance model is part based, which makes it robust to occlusions. For both algorithms, we use the default parameters provided by the authors for all of our experiments. For experiments where we track both location and scale we compare to IVT [12], setting the parameters such that only location and scale are tracked (rather than a full set of affine parameters). For the trackers than involve randomness, all results are averaged over 5 runs.

The system was implemented in C++ (code and data available on our project website: http://vision.ucsd.edu/project/tracking-online-multiple-instance-learning), and runs at about 25 frames per second (FPS).

### 4.1 Evaluation Methodology

Evaluating a tracking algorithm is itself a challenge. Qualitative comparison on test video clips is most common; quantitative comparison typically involves plotting the center location error versus frame number. Since these plots can be difficult to interpret, it is useful to summarize performance by computing the mean error over all the frames of the video. However, this value sometimes fails to correctly capture tracker performance. For example, if a tracker tracks an object closely for

[1] We also experimented with the LogitBoost loss function (as in [41], which penalizes noisy examples less harshly, and although it worked better than OAB, it did not outperform MILTrack. We omit the detailed results due to space constraints.

[2] Code available at http://www.vision.ee.ethz.ch/boostingTrackers/download.htm.

[3] Code available at http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm.

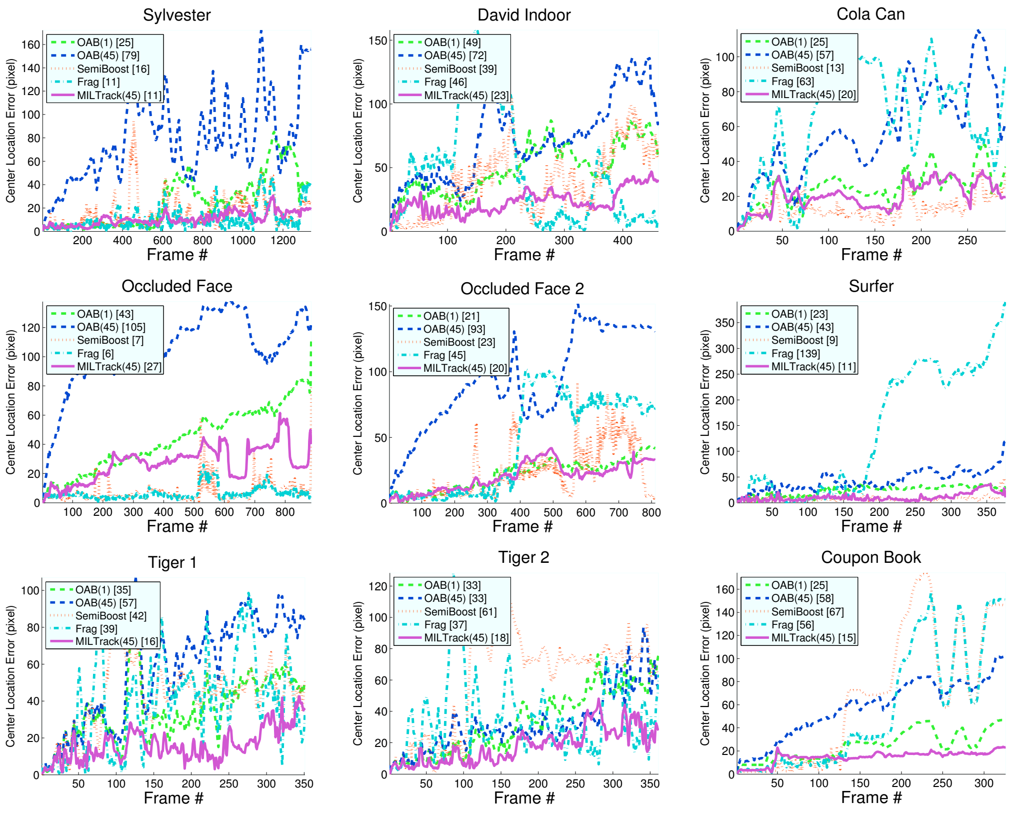


Fig. 4. **Tracking Object Location:** Location Error Plots. See text for details.

| Video Clip | OAB(1) | OAB(45) | SemiBoost | Frag | MILTrack(45) |
|---|---|---|---|---|---|
| Sylvester | 25 | 79 | 16 | **11** | **11** |
| David Indoor | 49 | 72 | *39* | 46 | **23** |
| Cola Can | 25 | 57 | **13** | 63 | *20* |
| Occluded Face | 43 | 105 | *7* | **6** | 27 |
| Occluded Face 2 | *21* | 93 | 23 | 45 | **20** |
| Surfer | 23 | 43 | **9** | 139 | *11* |
| Tiger 1 | *35* | 57 | 42 | 39 | **16** |
| Tiger 2 | *33* | *33* | 61 | 37 | **18** |
| Coupon Book | *25* | 58 | 67 | 56 | **15** |

TABLE 1

**Tracking Object Location:** average center location errors (pixels). **Bold green** font indicates best performance, *red italics* font indicates second best.

most of the video, but loses track completely on the last several frames, the mean location error may be higher than a tracker that sticks with the object, though not as precisely. The preference between these two behaviors inevitably depends on the final application.

For the above reasons, in addition to presenting screen shots and location error analysis, we include precision plots, similar to the analysis in [43], and suggested in [5]. These plots show the percentage of frames for which the estimated object location was within some threshold distance of the ground truth. To summarize these plots, we chose the threshold 20 and report the precision at this point in the curve (e.g. this is the percent of frames for which the tracker was less than 20 pixels off from the ground truth); this threshold roughly corresponds to at least a 50% overlap between the tracker bounding box and the ground truth. Note that we could have used the PASCAL [44] overlap criteria throughout our evaluation; however, this would require us to label full bounding boxes (which is more time consuming), and would make it difficult to compare trackers that do and do not return estimated scale. Finally, note that when multiple trails were done, we computed error for each trial and averaged the errors rather than averaging the tracker outputs and computing error.

### 4.2 Tracking Object Location

We perform our experiments on 3 publicly available video sequences, as well as 6 of our own. For all sequences we labeled the ground truth center of the object for every 5 frames, and interpolated the location in the other frames (with the exception of the "Occluded Face" sequence, for which the authors of [9] provided ground

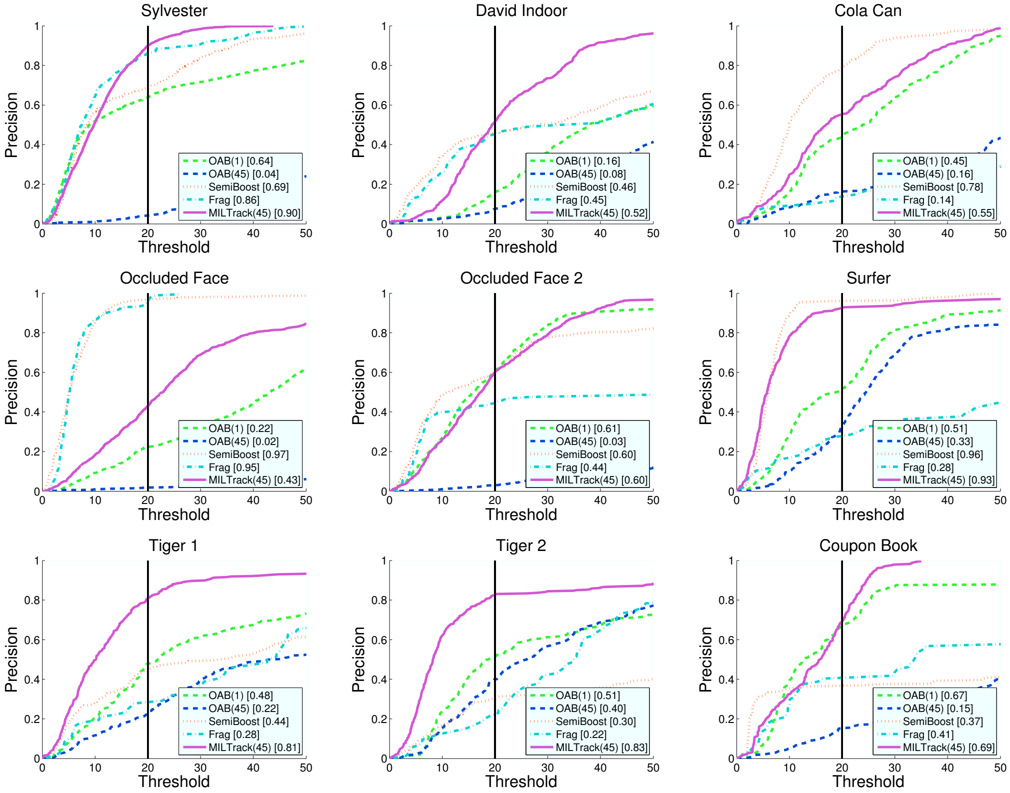


Fig. 5. **Tracking Object Location:** Precision plots. See text for details.

| Video Clip | OAB(1) | OAB(45) | SemiBoost | Frag | MILTrack(45) |
|---|---|---|---|---|---|
| Sylvester | 0.64 | 0.04 | 0.69 | *0.86* | **0.90** |
| David Indoor | 0.16 | 0.08 | *0.46* | 0.45 | **0.52** |
| Cola Can | 0.45 | 0.16 | **0.78** | 0.14 | *0.55* |
| Occluded Face | 0.22 | 0.02 | **0.97** | *0.95* | 0.43 |
| Occluded Face 2 | **0.61** | 0.03 | *0.60* | 0.44 | *0.60* |
| Surfer | 0.51 | 0.33 | **0.96** | 0.28 | *0.93* |
| Tiger 1 | *0.48* | 0.22 | 0.44 | 0.28 | **0.81** |
| Tiger 2 | *0.51* | 0.40 | 0.30 | 0.22 | **0.83** |
| Coupon Book | *0.67* | 0.15 | 0.37 | 0.41 | **0.69** |

TABLE 2

**Tracking Object Location:** precision at a fixed threshold of 20. **Bold green** font indicates best performance, *red italics* font indicates second best.

truth). All video frames were converted to gray scale prior to processing.

The quantitative results are summarized in Tables 1 and 2, and plots are shown in Fig. 4 and 5; Fig. 6,7 and 8 show screen captures for some of the clips. Below is a more detailed discussion of the video sequences.

#### 4.2.1 Sylvester & David Indoor

These two video sequences have been used in several recent tracking papers [12], [24], [25], and they present challenging lighting, scale and pose changes. Our algorithm achieves the best performance (tying FragTrack on the "Sylvester" sequence).

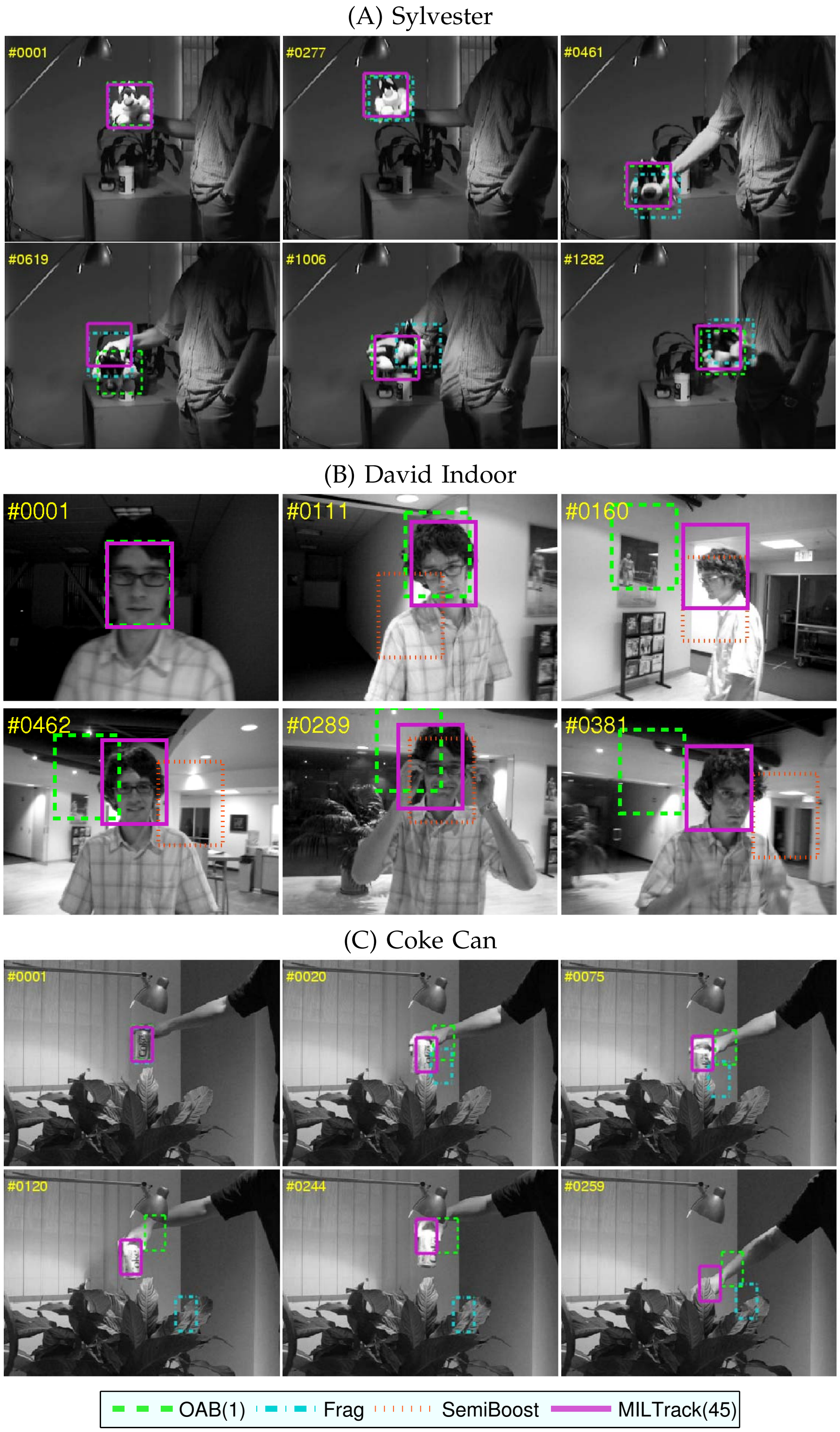
#### 4.2.2 Occluded Face, Occluded Face 2

In the "Occluded Face" sequence, which comes from the authors of [9], FragTrack performs the best because it is specifically designed to handle occlusions via a part-based model. However, on our similar, but more challenging clip, "Occluded Face 2", FragTrack performs poorly because it cannot handle appearance changes well (e.g. when the subject puts a hat on, or turns his face). This highlights the advantages of using an adaptive appearance model.

#### 4.2.3 CokeCan, Surfer

The Coke Can sequence contains a specular object, which adds some difficulty. The "Surfer" clip was downloaded from Youtube; this clip would be easier to track if color information were used[4], but since we use grayscale images for all experiments this clip is fairly challenging. Both MILTrack and the SemiBoost tracker perform well on these clips (cf. Fig. 5).

[4] It would be straightforward to extend our system to use color – e.g. compute Haar features over color channels.

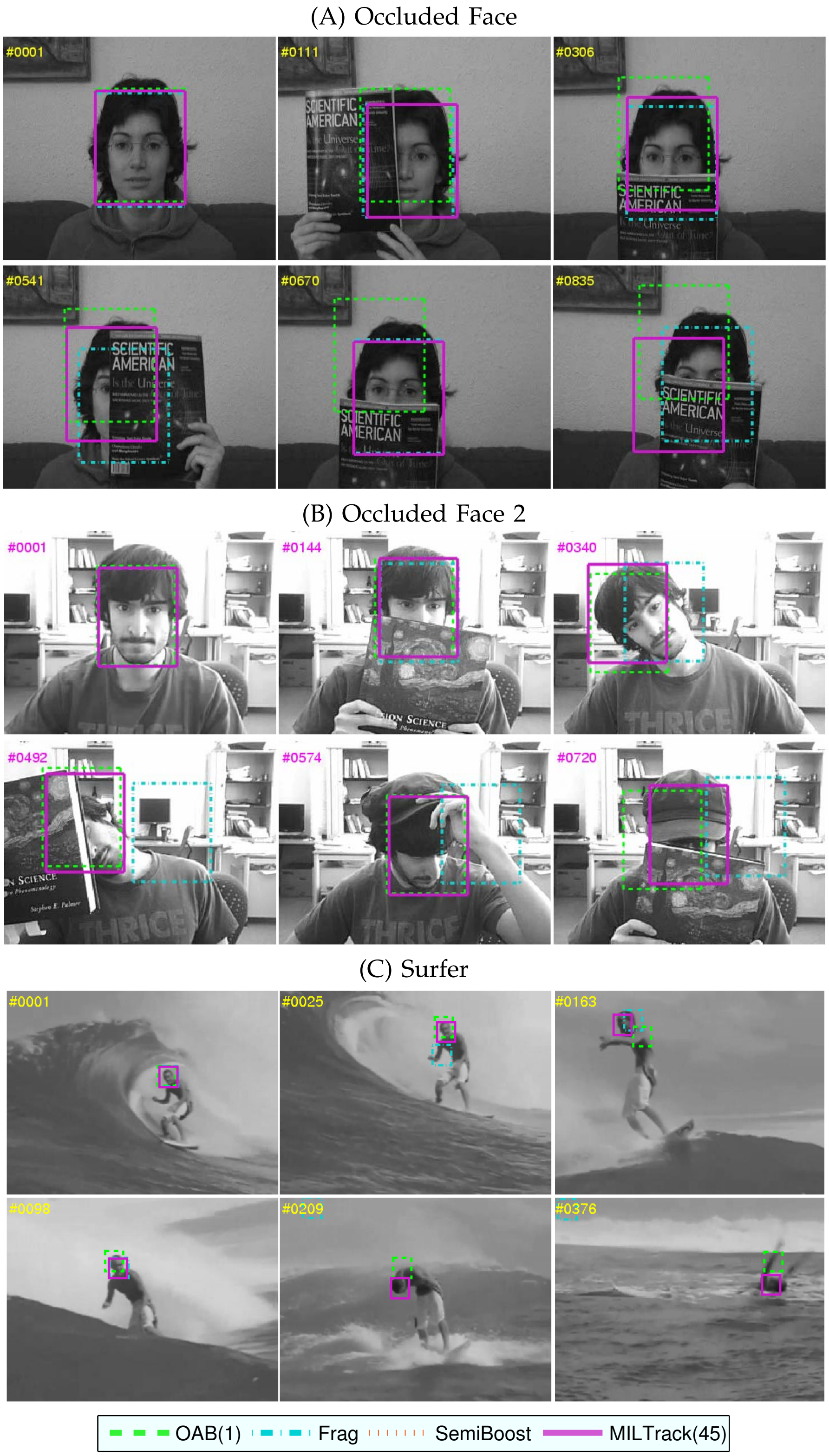


(A) Sylvester

(B) David Indoor

(C) Coke Can

Fig. 6. **Tracking Object Location:** screenshots of tracking results, highlighting instances of out-of-plane rotation, occluding clutter, scale and illumination change. For the sake of clarity we only show three trackers per video clip.


(A) Occluded Face

(B) Occluded Face 2

(C) Surfer

Fig. 7. **Tracking Object Location:** screenshots of tracking results, highlighting instances of out-of-plane rotation, occluding clutter, scale and illumination change. For the sake of clarity we only show three trackers per video clip.

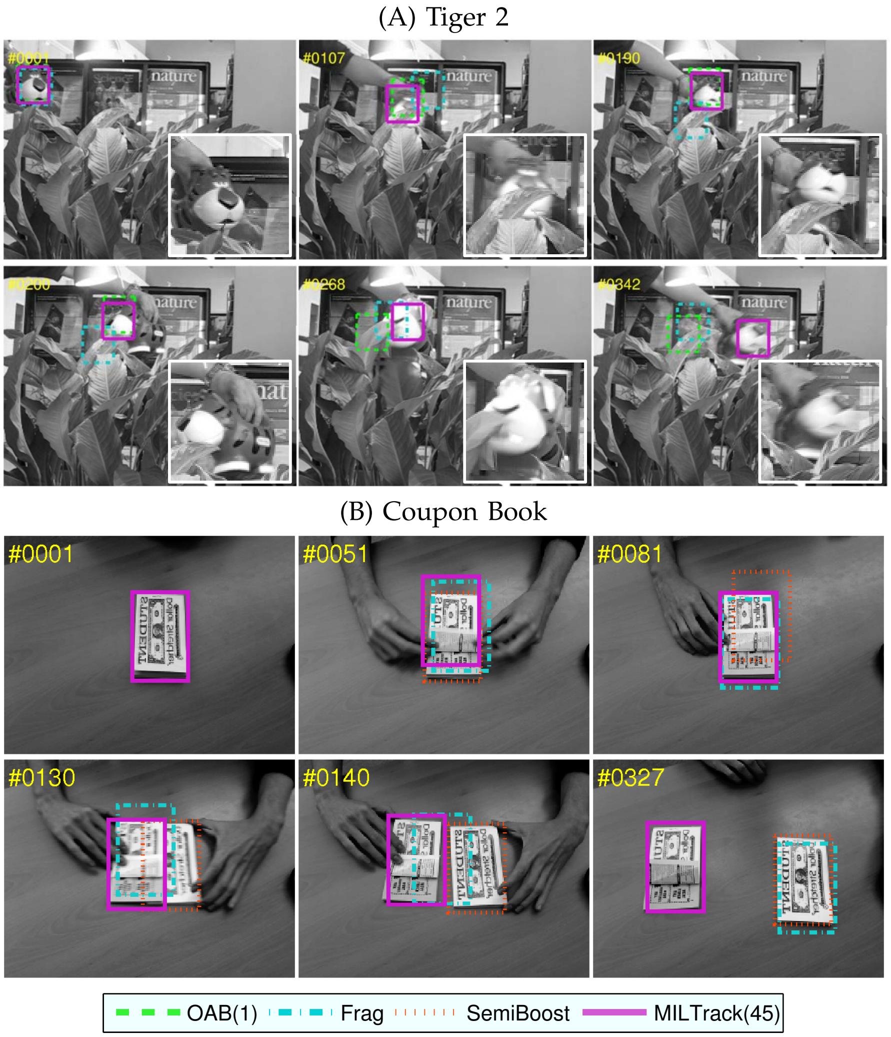


Fig. 8. **Tracking Object Location:** screenshots of tracking results, highlighting instances of out-of-plane rotation, occluding clutter, scale and illumination change. For the Tiger 2 clip we also include close up shots of the object to highlight the wide range of appearance changes. For the sake of clarity we only show three trackers per video clip.

#### 4.2.4 Tiger 1, Tiger 2

These sequences exhibit many challenges, and contain frequent occlusions and fast motion (which causes motion blur). The two sequences show the toy tiger in many different poses, and include out of plane rotations (cf. Fig. 8 (A)). Our algorithm outperforms the others, often by a large margin.

#### 4.2.5 Coupon Book

This clip illustrates a problem that arises when the tracker relies too heavily on the first frame. The appearance of the coupon book is changed after about 50 frames, by folding one of its pages; then an "imposter" coupon book is introduced to distract the trackers. MILTrack successfully tracks the correct coupon book, while FragTrack and the SemiBoost tracker are confused by the impostor object.

### 4.3 Tracking Object Location & Scale

Here we present results for both location and scale tracking. Scale tracking is independent of the appearance model, so our implementation of scale tracking for MILTrack is easily carried over to the OAB tracker. Note that the quantitative results we present are still based on object center location only; we do not measure error of scale estimation. This allows us to compare results of trackers that estimate scale and those with a fixed scale. Furthermore, gathering ground truth for object center is less time consuming than for a full bounding box.

#### 4.3.1 David Indoor

This is the same clip that we studied in the previous section. Here we see a big advantage of using scale tracking – MILTrack with scale performs better than MILTrack without scale, and it performs better than OAB(1) with scale. However, the IVT tracker achieves the best result on this video clip. We believe IVT is particularly well suited to faces since it uses a subspace (PCA) appearance model. We will see in the next experiments that IVT does not work well in other scenarios.

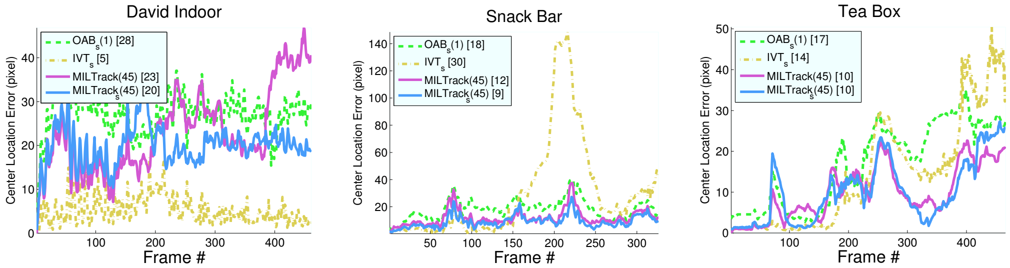


Fig. 9. **Tracking Object Location & Scale:** average center location errors. See text for details.

| Video Clip | OAB$_s$(1) | IVT$_s$ | MILTrack(45) | MILTrack$_s$(45) |
|---|---|---|---|---|
| David Indoor | 28 | **5** | 23 | *20* |
| Snack Bar | 18 | 30 | *12* | **9** |
| Tea Box | 17 | 14 | **10** | **10** |

TABLE 3

**Tracking Object Location & Scale:** location mean error. **Bold green** font indicates best performance, *red italics* font indicates second best.

### 4.3.2 Snack Bar

In this clip the goal is to track an object that changes in scale and moves against a background that is very similar in texture. We see that the IVT tracker fails in this case, when the object is turned upside down. The IVT tracker uses a generative model, rather than discriminative, so it does not take into account the negative examples from the image. Because the background is so similar to the object of interest in this video clip, IVT ultimately loses the object and snaps to some part of the background. As before, we see that MILTrack with scale performs better than MILTrack without scale and OAB(1) with scale; overall MILTrack achieves the best performance on this clip.

### 4.3.3 Tea Box

This clip again shows the shortcomings of IVT – the clip shows a box of tea which is moved around and rotated (exposing new faces of the box). IVT fails when these out of plane rotations take place (see Fig. 10(C), frame #240 and beyond). Though the center location error is similar for both version of MILTrack (Fig. 9), we can see the version that includes scale search results in more satisfactory results (e.g. frame #134).

## 5 Discussion/Conclusions

In this paper we presented a novel way of updating an adaptive appearance model of a tracking system. We argued that using Multiple Instance Learning to train the appearance classifier results in more robust tracking, and presented an online boosting algorithm for MIL. We presented empirical results on many challenging video clips where we measured quantitative performance of our tracker compared to a number of competing state of the art algorithms; these results show that our tracker is, on average, the most robust with respect to partial occlusions, and various appearance changes.

There are still some interesting unanswered questions about adaptive appearance models. Although our method results in more robust tracking, it cannot completely avoid the types of problems that adaptive appearance trackers suffer from. In particular, if an object is completely occluded for a long period of time, or if the object leaves the scene completely, any tracker with an adaptive appearance model will inevitably start learning from incorrect examples and lose track of the object. Some interesting work exploring ways to deal with this issue has been presented in [33] and more recently in [45]. These methods attempt to combine a pre-trained object detector with an adaptively trained tracker. One interesting avenue for future work would be to combine these ideas with the ones presented in this paper. Another challenge is to track articulated objects which cannot be easily delineated with a bounding box. These types of objects may require a part-based approach, such as the recent methods in object detection [15], [46].

Finally, online algorithms for Multiple Instance Learning could be useful in areas outside of visual tracking. Work on better algorithms and theoretical analysis relating offline/batch MIL and online MIL is already under way (e.g. [47]), and we suspect more is to come.

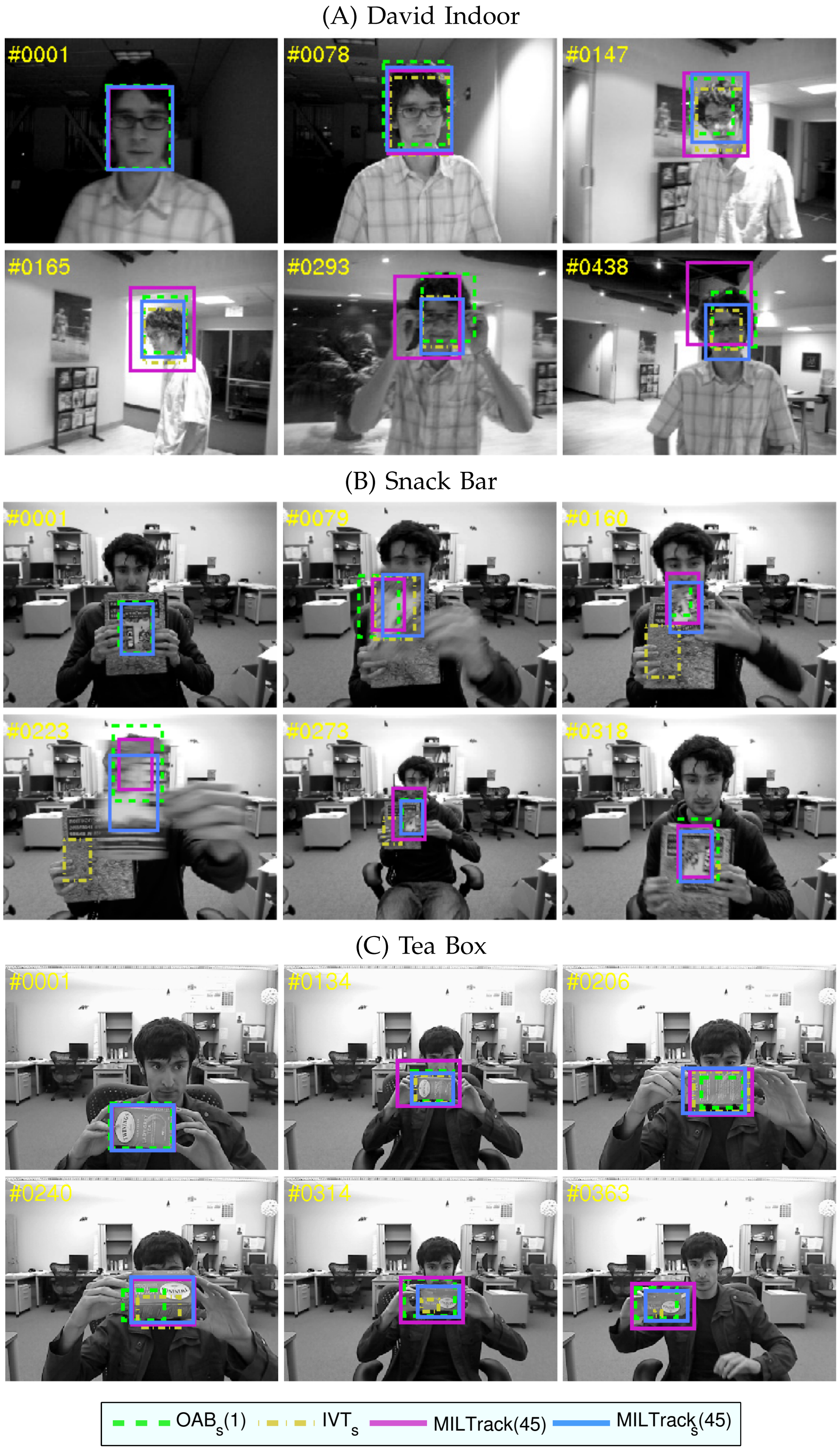
## Acknowledgements

Authors would like to thank Kristin Branson, Piotr Dollár, David Ross and the anonymous reviewers for valuable input. This research has been supported by NSF CAREER Grant #0448615, NSF IGERT Grant DGE-0333451, and ONR MURI Grant #N00014-08-1-0638. M.H.Y. is supported in part by a UC Merced faculty start-up fund and a Google faculty award. Part of this work was done while B.B. and M.H.Y. were at Honda Research Institute, USA.

## References

[1] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *CVPR*, 1998, pp. 232–237.

[2] M. Isard and J. Maccormick, "Bramble: a bayesian multiple-blob tracker," in *ICCV*, vol. 2, 2001, pp. 34–41.


(A) David Indoor

(B) Snack Bar

(C) Tea Box

Fig. 10. **Tracking Object Location & Scale:** screenshots showing results for tracking both location and scale of objects. Note that the localization is much more precise when scale is one of the tracked parameters.

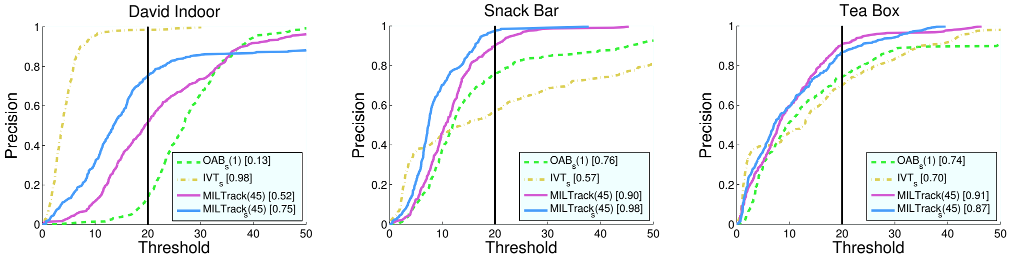


Fig. 11. **Tracking Object Location & Scale:** Precisions plots. See text for details.

| Video Clip | OAB$_s$(1) | IVT$_s$ | MILTrack(45) | MILTrack$_s$(45) |
|---|---|---|---|---|
| David Indoor | 0.13 | **0.98** | 0.52 | *0.75* |
| Snack Bar | 0.76 | 0.57 | *0.90* | **0.98** |
| Tea Box | 0.74 | 0.70 | **0.91** | *0.87* |

TABLE 4

**Tracking Object Location & Scale:** precision at a fixed threshold of 20. **Bold green** font indicates best performance, *red italics* font indicates second best.

[3] K. Branson and S. Belongie, “Tracking multiple mouse contours (without too many samples),” in *CVPR*, vol. 1, 2005.

[4] V. Lepetit and P. Fua, “Keypoint recognition using randomized trees,” *PAMI*, vol. 28, no. 9, p. 1465, 2006.

[5] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, 2006.

[6] G. Hager and P. Belhumeur, “Efficient region tracking with parametric models of geometry and illumination,” *PAMI*, vol. 20, no. 10, pp. 1025–1039, 1998.

[7] M. Black and A. Jepson, “Eigentracking: Robust matching and tracking of articulated objects using a view-based representation,” *IJCV*, vol. 26, no. 1, pp. 63–84, 1998.

[8] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” in *CVPR*, vol. 2, 2000, pp. 142–149.

[9] A. Adam, E. Rivlin, and I. Shimshoni, “Robust fragments-based tracking using the integral histogram,” in *CVPR*, vol. 1, 2006, pp. 798–805.

[10] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, “Robust online appearance models for visual tracking,” *PAMI*, vol. 25, no. 10, pp. 1296–1311, 2003.

[11] I. Matthews, T. Ishikawa, and S. Baker, “The template update problem,” *PAMI*, pp. 810–815, 2004.

[12] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *IJCV*, vol. 77, no. 1, pp. 125–141, 2008.

[13] T. G. Dietterich, R. H. Lathrop, and L. T. Perez, “Solving the multiple-instance problem with axis parallel rectangles,” *Artificial Intelligence*, pp. 31–71, 1997.

[14] P. Viola, J. C. Platt, and C. Zhang, “Multiple instance boosting for object detection,” in *NIPS*, 2005, pp. 1417–1426.

[15] P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu, “Multiple component learning for object detection,” in *ECCV*, 2008.

[16] S. Andrews, I. Tsochantaridis, and T. Hofmann, “Support vector machines for multiple-instance learning,” in *NIPS*, 2003, pp. 577–584.

[17] C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie, “Weakly Supervised Object Recognition and Localization with Stable Segmentations,” in *ECCV*, 2008.

[18] S. Vijayanarasimhan and K. Grauman, “Keywords to Visual Categories: Multiple-Instance Learning for Weakly Supervised Object Categorization,” in *CVPR*, 2008.

[19] K. Okuma, A. Taleghani, N. De Freitas, J. Little, and D. Lowe, “A boosted particle filter: Multitarget detection and tracking,” *ECCV*, pp. 28–39, 2004.

[20] M. Isard and A. Blake, “Contour tracking by stochastic propagation of conditional density,” *ECCV*, vol. 1064, pp. 343–356, 1996.

[21] L. Vese and T. Chan, “A multiphase level set framework for image segmentation using the Mumford and Shah model,” *IJCV*, vol. 50, no. 3, pp. 271–293, 2002.

[22] M. Salzmann, V. Lepetit, and P. Fua, “Deformable surface tracking ambiguities,” in *CVPR*, 2007.

[23] A. O. Balan and M. J. Black, “An adaptive appearance model approach for model-based articulated object tracking,” in *CVPR*, vol. 1, 2006, pp. 758–765.

[24] R. Lin, D. Ross, J. Lim, and M.-H. Yang, “Adaptive Discriminative Generative Model and Its Applications,” in *NIPS*, 2004, pp. 801–808.

[25] H. Grabner, M. Grabner, and H. Bischof, “Real-time tracking via on-line boosting,” in *BMVC*, 2006, pp. 47–56.

[26] X. Liu and T. Yu, “Gradient feature selection for online boosting,” in *ICCV*, 2007, pp. 1–8.

[27] S. Avidan, “Ensemble tracking,” in *CVPR*, vol. 2, 2005, pp. 494–501.

[28] ——, “Support vector tracking,” *PAMI*, vol. 26, no. 8, pp. 1064–1072, 2004.

[29] J. Wang, X. Chen, and W. Gao, “Online selecting discriminative tracking features using particle filter,” in *CVPR*, vol. 2, 2005, pp. 1037–1042.

[30] R. T. Collins, Y. Liu, and M. Leordeanu, “Online selection of discriminative tracking features,” *PAMI*, vol. 27, no. 10, pp. 1631–1643, 2005.

[31] G. Mori and J. Malik, “Recovering 3d human body configurations using shape contexts,” *PAMI*, vol. 28, no. 7, pp. 1052–1062, 2006.

[32] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *CVPR*, vol. 1, 2001, pp. 511–518.

[33] H. Grabner, C. Leistner, and H. Bischof, “Semi-supervised on-line boosting for robust tracking,” in *ECCV*, 2008.

[34] N. C. Oza, “Online Ensemble Learning,” *Ph.D. Thesis, University of California, Berkeley*, 2001.

[35] P. Dollár, Z. Tu, H. Tao, and S. Belongie, “Feature mining for image classification,” in *CVPR*, 2007.

[36] Z. Khan, T. Balch, and F. Dellaert, “A rao-blackwellized particle filter for eigentracking,” in *CVPR*, vol. 2, 2004.

[37] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[38] B. Babenko, P. Dollár, Z. Tu, and S. Belongie, “Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning,” in *Faces in Real-Life Images*, 2008.

[39] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[40] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting,” *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[41] C. Leistner, A. Saffari, P. Roth, and H. Bischof, “On robustness of on-line boosting-a competitive study,” in *3rd IEEE ICCV Workshop on On-line Computer Vision*, 2009.

[42] H. Grabner and H. Bischof, “On-line boosting and vision,” in *CVPR*, Washington, DC, USA, 2006, pp. 260–267.

[43] A. Chan and N. Vasconcelos, “Modeling, clustering, and segment-

ing video with mixtures of dynamic textures," *PAMI*, vol. 30, no. 5, pp. 909–926, 2008.

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results." [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html

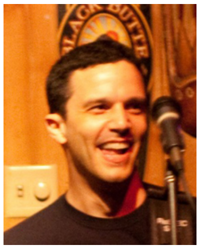
[45] S. Stalder, H. Grabner, and L. van Gool, "Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but not Simpler than Recognition," in *Online Learning in Computer Vision (OLCV) Workshop*, 2009.

[46] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model." CVPR, 2008.

[47] L. Mu, J. Kwok, and L. Bao-liang, "Online Multiple Instance Learning with No Regret," in *CVPR*, 2010.

**Boris Babenko** graduated summa cum laude from U.C. San Diego, earning a B.S. degree in Computer Science in 2006. He is currently a Ph.D. student at U.C. San Diego. He is a recipient of a 2007 "Vision and Learning in Humans and Machines" NSF IGERT Award and the 2010 Google Research Fellowship in Computer Vision. His research interests include computer vision and machine learning.

**Ming-Hsuan Yang** is an assistant professor in EECS at University of California, Merced. He received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2000. He studied at the National Tsing-Hua University, Taiwan, the University of Southern California, and the University of Texas at Austin. He was a senior research scientist at the Honda Research Institute working on vision problems related to humanoid robots. In 1999, he received the Ray Ozzie fellowship for his research work. He coauthored the book Face Detection and Gesture Recognition for Human-Computer Interaction (Kluwer Academic 2001) and edited special issue on face recognition for Computer Vision and Image Understanding in 2003. He served as an area chair for the IEEE Conference on Computer Vision and Pattern Recognition, and the Asian Conference on Computer Vision. He is an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence, and Image and Vision Computing. He is a senior member of the IEEE and the ACM.



**Serge Belongie** received the B.S. degree (with honor) in Electrical Engineering from the California Institute of Technology in 1995 and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences (EECS) at U.C. Berkeley in 1997 and 2000, respectively. While at Berkeley, his research was supported by a National Science Foundation Graduate Research Fellowship. He is also a co-founder of Digital Persona, Inc., and the principal architect of the Digital Persona fingerprint recognition algorithm. He is currently an associate professor in the Computer Science and Engineering Department at U.C. San Diego. His research interests include computer vision and pattern recognition. He is a recipient of the NSF CAREER Award and the Alfred P. Sloan Research Fellowship. In 2004 MIT Technology Review named him to the list of the 100 top young technology innovators in the world (TR100).