Visual Tracking with Online Multiple Instance Learning

Boris Babenko
University of California, San Diego
bbabenko@cs.ucsd.edu

Ming-Hsuan Yang
University of California, Merced
mhyang@ucmerced.edu

Serge Belongie
University of California, San Diego
sjb@cs.ucsd.edu

Abstract

*In this paper, we address the problem of learning an adaptive appearance model for object tracking. In particular, a class of tracking techniques called "tracking by detection" have 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 degrades the classifier and can cause further 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 present a novel online* MIL *algorithm for object tracking that achieves superior results with real-time performance.*

1. Introduction

Object tracking has many practical applications (*e.g.* surveillance, HCI) and has long been studied in computer vision. Although there has been some success with building domain specific trackers (*e.g.* faces [6], humans [16]), tracking generic objects has remained very challenging. Generally there are three components to a tracking system: image representation (*e.g.* filter banks [17], subspaces [21], *etc.*), appearance model, and motion model; although in some cases these components are merged. In this work we focus mainly on the appearance model since this is usually the most challenging to design.

Although many tracking methods employ static appearance models that are either defined manually or trained using the first frame [16, 8, 1], these methods tend to have difficulties tracking objects that exhibit significant appearance changes. It has been shown that in many scenarios an adaptive appearance model, which evolves during the tracking process as the appearance of the object changes, is the key to good performance [17, 21]. Another choice in the design of appearance models is whether to model only the object [5, 21], or both the object and the background [18, 14, 19, 4, 3, 24, 7]. 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. Because these methods have a lot in common with object detection they have been termed "tracking by detection". In particular, the recent advances in face detection [22] have inspired some successful real-time tracking algorithms [14, 19].

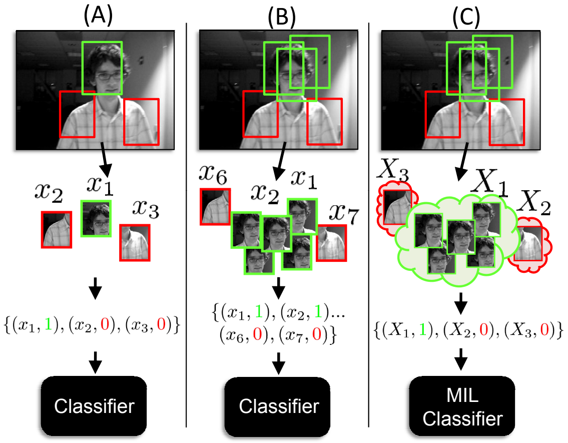


Figure 1. **Updating a discriminative appearance model:** (A) Using a single positive image patch to update a traditional discriminative classifier. The positive image patch chosen does not capture the object perfectly. (B) Using several positive image patches to update a traditional discriminative classifier. This can confuse the classifier causing poor performance. (C) Using one positive bag consisting of several image patches to update a MIL classifier. See Section 3 for empirical results of these three strategies.

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

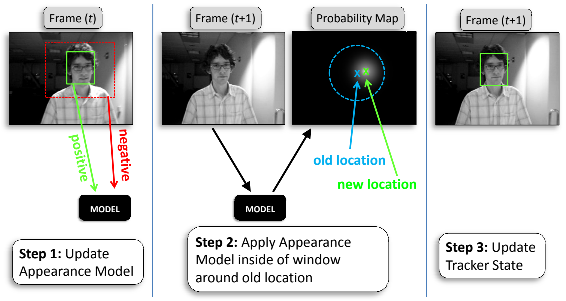

Frame (t)
Frame (t+1)
Probability Map
Frame (t+1)
positive
negative
MODEL
MODEL
old location
new location
Step 1: Update Appearance Model
Step 2: Apply Appearance Model inside of window around old location
Step 3: Update Tracker State

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

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.* [15] 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.*, when it is safe to assume small interframe motion).

Some of the above issues are encountered in object detection because it is difficult for a human labeler to be consistent with respect to how the positive examples are cropped. In other words, the *exact* object locations are unknown. In fact, Viola *et al.* [23] argue that object detection has inherent ambiguities that make it more difficult to train a classifier using traditional methods. For this reason they suggest the use of a Multiple Instance Learning (MIL) [9] approach for object detection. We give a more formal definition of MIL in Section 2.2, 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.

Algorithm 1 MILTrack

```
Input: New video frame number $k$
1: Crop out a set of image patches, $X^s = \{x | s > ||l(x) - l^*_{t-1}||\}$ and compute feature vectors.
2: Use MIL classifier to estimate $p(y = 1|x)$ for $x \in X^s$.
3: Update tracker location $l^*_t = l\Big(\mathrm{argmax}_{x \in X^s}\, p(y|x)\Big)$
4: Crop out two sets of image patches $X^r = \{x | r > ||l(x) - l^*_t||\}$ and $X^{r,\beta} = \{x | \beta > ||l(x) - l^*_t|| > r\}$.
5: 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.* [23], and propose to use a MIL based appearance model for object tracking. 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 implement such a tracker, an online MIL algorithm is required. The algorithm we propose is based on boosting and is related to the MILBoost algorithm [23] as well as the Online-AdaBoost algorithm [20] (to our knowledge no other online MIL algorithm currently exists in the literature). 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.

2. 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 [23]. We then review online boosting [20, 14] and present a novel boosting based algorithm for online MIL, which is required for real-time MIL based tracking. Finally, we review various implementation details.

2.1. 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. As we mentioned earlier, the system contains three components: image representation, appearance model and motion model. Our image representation consists of a set of Haar-like features that are computed for each image patch [22, 10]; this is discussed in more detail in Section 2.5. 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 l^*_t. Let $l(x)$ denote the location of image patch x. For each new frame we crop out a set of image patches $X^s = \{x|s > ||l(x) - l^*_{t-1}||\}$ 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:

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

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

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

This could be extended with something more sophisticated, such as a particle filter, as is done in [24, 21]; however, we again emphasize that our focus is on the appearance model. Furthermore, note that it is straightforward to track other motion information such as scale and rotation, and we chose to track only the location for simplicity and computational efficiency reasons. It is also worth noting that the Haar-like features we use are fairly invariant to moderate rotation and scale changes.

Once the tracker location is updated, we proceed to update the appearance model. We crop out a set of patches $X^r = \{x|r > ||l(x) - l^*_t||\}$, where $r < s$ is an integer radius, 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 $X^{r,\beta} = \{x|\beta > ||l(x) - l^*_t|| > r\}$, where r is same as before, and β 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[1]. Details on how these parameters were set are in Section 3, although we use the *same* parameters throughout *all* the experiments. Fig. 1 contains an illustration comparing appearance model updates using MIL and a standard learning algorithm. We continue with a more detailed review of MIL.

2.2. 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 the Multiple Instance Learning framework the 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 assumed to exist, but 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 [9, 2, 23]. The algorithm that is most closely related to our work is the MILBoost algorithm proposed by Viola *et al.* in [23]. MILBoost uses the the gradient boosting framework [13] to train a boosting classifier that maximizes the log likelihood of bags:

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

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 [23] the Noisy-OR (NOR) model is adopted for doing this:

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

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. Note that MILBoost is a batch algorithm (meaning it needs the entire training data at once) and cannot be trained in an online manner as we need in our tracking application (we refer the reader to [23] for further details on MILBoost). 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 2.4.

2.3. Related Work in Online Boosting

Our algorithm for online MIL is based on the boosting framework [11] and is related to the work on Online AdaBoost [20] and its adaptation in [14]. The goal of boosting is to combine many weak classifiers $\mathbf{h}(x)$ (usually decision stumps) into an additive strong classifier:

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

[1] Note that we could place all negative examples into a single negative bag. Our intuition is that there is no ambiguity about negative examples, so placing them into separate bags makes more sense. Furthermore the particular loss function we choose is not affected by this choice.

where α_k are scalar weights. There have been many boosting algorithms proposed to learn this model in batch mode [11, 12]; 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 particular useful in computer vision because it creates classifiers that are efficient at run time [22].

In [20], Oza develops an online variant of the popular AdaBoost algorithm [11], 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 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.* [14] 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 $\mathbf{h}$ from this pool by keeping running averages of errors for each as in [20], 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.

2.4. Online Multiple Instance Boosting

The algorithms in [20] and [14] 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 [12], we take a statistical view of boosting, where the algorithm is trying to optimzie a specific loss 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.

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

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 estimate of $p(y|x)$ to maximize the 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 α_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 of these weak classifiers in parallel, similar to [14]. Note that although examples are passed 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, using the following criteria:

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

See Algorithm 2 for the pseudo-code of Online-MILBoost.

2.4.1 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 **h** based on the bag likelihood loss function. Second, if we compare Equations 7 and 9 we see that the latter has a much more restricted choice of weak classifiers. However, this approximation does not seem to degrade the performance of the classifier in practice. 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 (see Section 2.5 for more details).

2.5. Implementation Details

2.5.1 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 γ is a learning rate parameter. The update rules for μ_0 and σ_0 are similarly defined.

2.5.2 Image Features

We represent each image patch as a vector of Haar-like features [22], which are randomly generated, similar to [10]. 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 [22].

3. 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 [14]. We plugged this learning algorithm into our system, and used the same features and motion model as for MILTrack (See Section 2.1). We acknowledge the fact that our implementation of the OAB tracker achieves worse performance than is reported in [14]; this could be because we are using simpler features, or because our parameters were not tuned per each 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 positive radius $r = 5$. This generates a total of 45 image patches comprising one positive bag. For the OAB tracker we tried two variations. In the first variation we set $r = 1$ generating only one positive example per frame; in the second variation we set $r = 5$ as we do in MILTrack (although in this case each of the 45 image patches is labeled positive). 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 (*cf*. Table 1 and Fig. 5). The scalar β for sampling negative examples was set to 50, and we randomly sample 65 negative image patches from the set $X^{r,\beta}$. The learning rate γ 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.

We also implemented the SemiBoost tracker, as described in [15]. 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 sig-

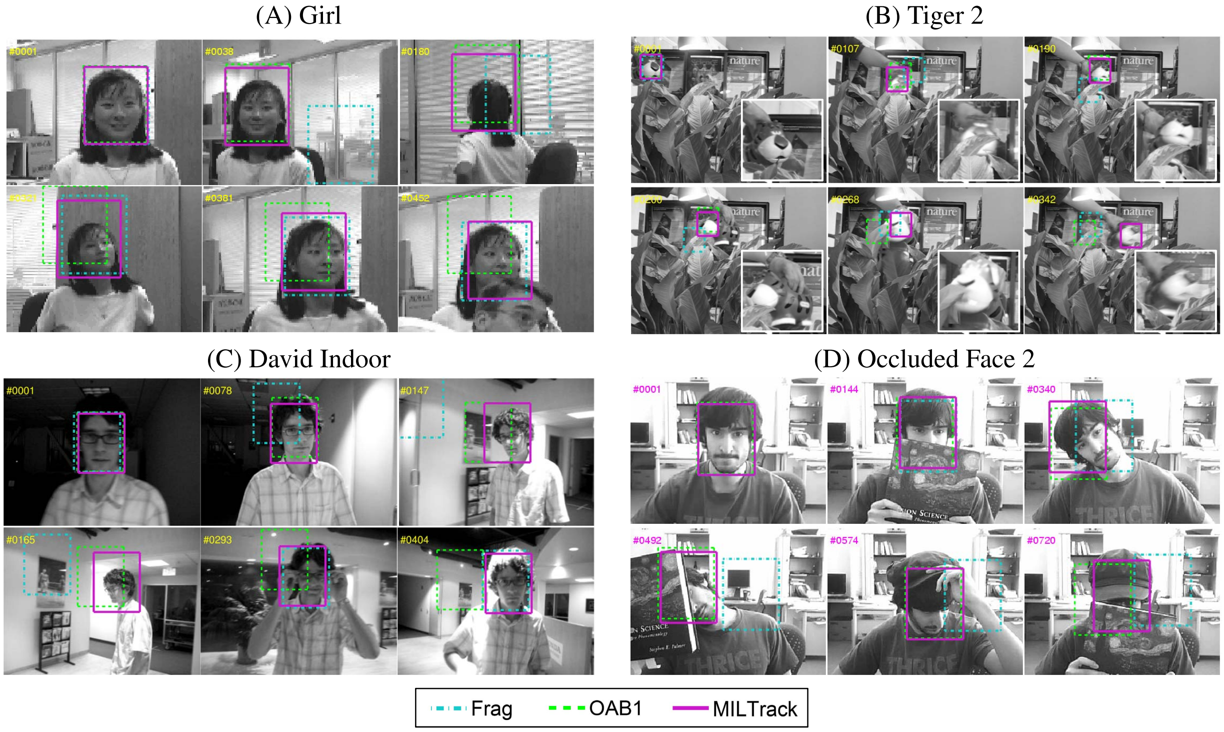


Figure 3. Screenshots of tracking results, highlighting instances of (A) out-of-plane rotation, (B) occluding clutter, (C) scale and illumination change, and (D) in-plane rotation and object occlusion. For the Tiger 2 clip we also include close up shots of the object to highlight the wide range of appearance changes. For the sake of clarity we only show MILTrack compared to OAB1 and FragTrack because these two on average got the best results next to MILTrack. Table 1 and Fig. 5 include quantitative results for all trackers we evaluated.

nificant appearance changes this algorithm did not perform well. In our implementation we use the same features and weak classifiers as our MILTrack and OAB implementations. To gather unlabeled examples we sample 200 patches from a circular region around the previous tracker location with a radius of 10 pixels.

Finally, to gauge absolute performance we also compare our results to the recently proposed FragTrack algorithm [1], the code for which is publicly available. 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. We use the same parameters as the authors used in their paper for all of our experiments. We also experimented with other trackers such as IVT [21], but found that it was difficult to compare performance since other trackers require parameter tuning per video sequence. Furthermore, as noted in [21] the IVT tracker is not expected to work well when target objects are heavily occluded.

Since the boosting based trackers involve some slight randomness, we ran them 5 times and averaged the results for each video clip.

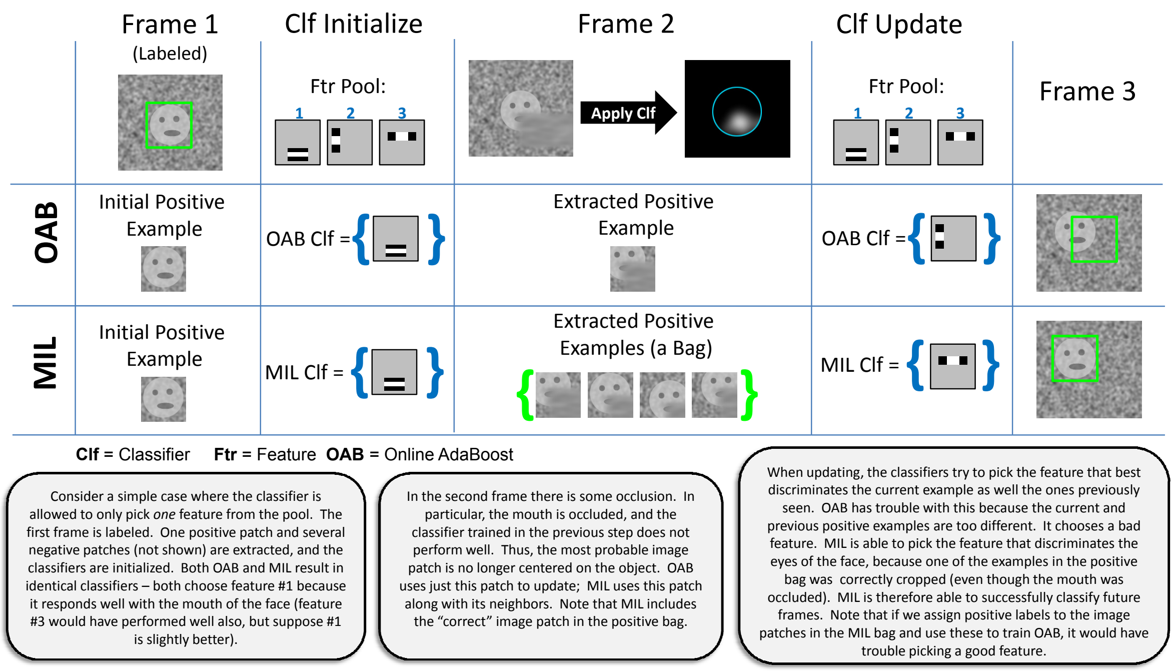
Video Clip	OAB1	OAB5	SemiBoost	Frag	MILTrack
David Indoor	49	72	59	46	23
Sylvester	25	79	22	11	11
Occluded Face	44	105	41	6	27
Occluded Face 2	21	93	43	45	20
Girl	48	68	52	27	32
Tiger 1	35	58	46	40	15
Tiger 2	34	33	53	38	17
Coke Can	25	57	85	63	21

Table 1. Average center location errors (pixels). Algorithms compared are Online-AdaBoost Tracker [14] with $r = 1$ (OAB1) and $r = 5$ (OAB5), FragTrack [1], SemiBoost Tracker [15], and MILTrack with $r = 5$. Green indicates best performance, red indicates second best. See text for details.

3.1. Video Sequences

We perform our experiments on 4 publicly available video sequences, as well as 4 of our own. For all sequences we labeled the ground truth center of the object for every 5 frames[2] (with the exception of the "Occluded Face" sequence, for which the authors of [1] provided ground truth). All video frames were gray scale, and resized to 320 × 240 pixels. The quantitative results are summarized in Table 1 and Fig. 5; Fig. 3 shows screen captures for some of the

[2] Data and code are available at `http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml`; video results available on youtube: `http://www.youtube.com/miltrack08`



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).

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.

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 have trouble picking a good feature.

Figure 4. An illustration of how using MIL for tracking can deal with occlusions.

clips. Below is a more detailed discussion of the video sequences.

Sylvester & David Indoor

These two video sequences have been used in several recent tracking papers [21, 18, 14], and they present challenging lighting, scale and pose changes. Our algorithm achieves the best performance (tying FragTrack on the "Sylvester" sequence). Note that although our implementation is single scale and orientation, the Haar-like feature we use are fairly invariant to scale and orientation changes present in these clips. The scale changes can be seen in Fig. 3(C) – the subjects' head size ranges from 88×105 pixels to 44×52 pixels.

Occluded Face, Occluded Face 2, & Girl

In the "Occluded Face" sequence, which comes from the authors of [1], 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, though it is not straightforward to incorporate such a model into FragTrack. Finally, the "Girl" sequence comes from the authors of [6]. FragTrack gets a better average error than MILTrack; however, FragTrack looses the target completely between frames 20 and 50 (*cf*. Fig. 5). Note that subject in this clip performs a 360° out of plane rotation.

Tiger 1, Tiger 2, & Coke Can

These sequences exhibit many challenges. All three video clips contains frequent occlusions and fast motion (which causes motion blur). The Tiger 1 & 2 sequences show the toy tiger in many different poses, and include out of plane rotations (*cf*. Fig. 3(B)). The Coke Can sequence contains a specular object, which adds some difficulty. Our algorithm outperforms the others, often by a large margin.

3.2. Discussion

In all cases our MILTrack algorithm outperforms both versions of the Online Adaboost and SemiBoost Trackers, and in most cases it outperforms or ties the FragTrack algorithm (*cf*. Table 1 and Fig. 5); overall, it is the most stable tracker. The reason for the superior performance is that the Online MILBoost algorithm is able to handle ambiguously labeled training examples, which are provided by the tracker itself. Rather than extracting only one positive image patch and taking the risk that that image patch is suboptimal (as is done in OAB1), or taking multiple image patches and explicitly labeling them positive (as is done in OAB5), our MIL based approach extracts a bag of potentially positive image patches and has the flexibility to pick out the best one. The SemiBoost algorithm throws away a lot of useful information by leaving all extracted image unlabeled, except for the first frame. This leads to poor performance in the presence of significant appearance changes. We notice that MILTrack is particularly good at dealing with partial occlusions (*e.g*. Tiger 2 sequence). Fig. 4 contains an illustration showing how MIL could result in better

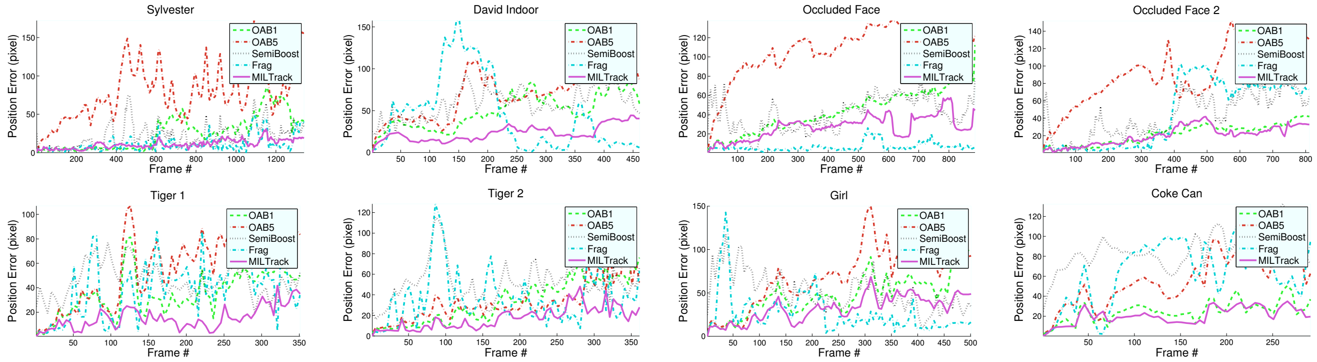


Figure 5. Error plots for eight video clips we tested on.

performance when partial occlusion is present.

4. Conclusions & Future Work

In this paper we have presented a tracking system called MILTrack that uses a novel Online Multiple Instance Learning algorithm. The MIL framework allows us to update the appearance model with a set of image patches, even though it is not known which image patch precisely captures the object of interest. This leads to more robust tracking results with fewer parameter tweaks. Our algorithm is simple to implement, and can run at real-time speeds[3].

There are many interesting ways to extend this work in the future. First, the motion model we used here is fairly simple, and could be replaced with something more sophisticated, such as a particle filter as in [21, 24]. Furthermore, it would be interesting to extend this system to be part-based like [1], which could further improve the performance with the presence of severe occlusions. A part-based model could also potentially reduce the amount of drift by better aligning the tracker location with the object. Finally we are interested in other possible applications for our online Multiple Instance Learning algorithm.

Acknowledgements

Authors would like to thank Kristin Branson, Piotr Dollár and David Ross 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. Part of this work was done while B.B. and M.H.Y. were at Honda Research Institute, USA.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, volume 1, pages 798–805, 2006.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, pages 577–584, 2003.

[3] S. Avidan. Support vector tracking. *PAMI*, 26(8):1064–1072, 2004.

[4] S. Avidan. Ensemble tracking. In *CVPR*, volume 2, pages 494–501, 2005.

[5] A. O. Balan and M. J. Black. An adaptive appearance model approach for model-based articulated object tracking. In *CVPR*, volume 1, pages 758–765, 2006.

[6] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR*, pages 232–237, 1998.

[7] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, volume 2, pages 142–149, 2000.

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

[10] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, June 2007.

[11] 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*, 55:119–139, 1997.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[13] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[14] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, pages 47–56, 2006.

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

[16] M. Isard and J. Maccormick. Bramble: a bayesian multiple-blob tracker. In *ICCV*, volume 2, pages 34–41, 2001.

[17] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *PAMI*, 25(10):1296–1311, 2003.

[18] R. Lin, D. Ross, J. Lim, and M.-H. Yang. Adaptive Discriminative Generative Model and Its Applications. In *NIPS*, pages 801–808, 2004.

[19] X. Liu and T. Yu. Gradient feature selection for online boosting. In *ICCV*, pages 1–8, 2007.

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

[21] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1):125–141, May 2008.

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages 511–518, 2001.

[23] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, pages 1417–1426, 2005.

[24] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. In *CVPR*, volume 2, pages 1037–1042, 2005.

[3] Our implementation currently runs at 25 frames per second on a Core 2 Quad desktop machine.