Semi-supervised On-Line Boosting for Robust Tracking*

Helmut Grabner[1,2], Christian Leistner[1], and Horst Bischof[1]

[1] Institute for Computer Graphics and Vision, Graz University of Technology, Austria
{hgrabner,leistner,bischof}@icg.tugraz.at
[2] Computer Vision Laboratory, ETH Zurich, Switzerland
grabner@vision.ee.ethz.ch

Abstract. Recently, on-line adaptation of binary classifiers for tracking have been investigated. On-line learning allows for simple classifiers since only the current view of the object from its surrounding background needs to be discriminiated. However, on-line adaption faces one key problem: Each update of the tracker may introduce an error which, finally, can lead to tracking failure (drifting). The contribution of this paper is a novel on-line semi-supervised boosting method which significantly alleviates the drifting problem in tracking applications. This allows to limit the drifting problem while still staying adaptive to appearance changes. The main idea is to formulate the update process in a semi-supervised fashion as combined decision of a given prior and an on-line classifier. This comes without any parameter tuning. In the experiments, we demonstrate real-time tracking of our SemiBoost tracker on several challenging test sequences where our tracker outperforms other on-line tracking methods.

1 Introduction

Designing robust tracking methods is still an open issue, especially considering various complicated variations that may occur in natural scenes, *e.g.*, shape and appearance changes of the object, illumination variations, partial occlusions of the object, cluttered scenes, *etc.* Recently tracking has been formulated as a classification problem, *i.e.*, the task of tracking is to optimally separate in each frame the object from the background (*e.g.*, Avidan [1] used support vector machines). Also, feature based tracking methods are formulated as classification tasks, *i.e.*, the work of Lepetit et al. [2] uses randomized trees and ferns based on pixel pairs [3] to discriminate key points by classifiers. In these approaches, the object to be tracked is trained a priori. The main motivation for using classifiers in these approaches is the increased speed, *i.e.*, the time is spent at the training stage and a fast classifier is available at the tracking stage. All these approaches use off-line training,

* This work has been supported by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04, the FFG project EVis (813399) under the FIT-IT program and the Austrian Science Fund (FWF) under the doctoral program Confluence of Vision and Graphics W1209.

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part I, LNCS 5302, pp. 234–247, 2008.

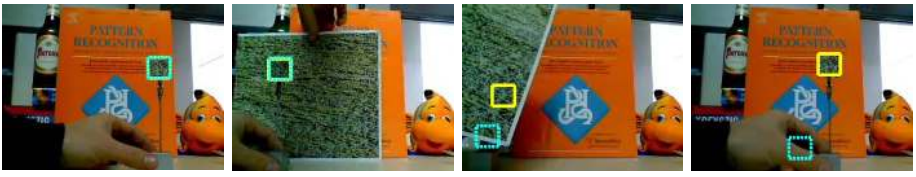

Fig. 1. Tracking of a textured patch with difficult background (same texture). As soon as the object gets occluded the original tracker from [4] (dotted cyan), drifts away. Our proposed SemiBoost tracker (yellow) successfully re-detects the object and continues tracking without drifting.

which has two important limitations. First, all appearance variations need to be covered in advance which implies that the object to be tracked needs to be known beforehand. Tracking will fail if a variation of the object is not covered in the training phase. Second, since the tracker is fixed it has to cope with all different backgrounds, therefore the classifiers are usually quite complex.

In order to cope with these problems the tracker needs to be adaptive. Collins and Liu [5] were among the first to emphasize (and exploit) this principle in a tracker. They proposed a method to adaptively select color features that best discriminate the object from the current background. There has also been considerable work along these lines, *e.g.*, Lim *et al.* [6] used incremental subspace learning for tracker updating and Avidan [7] use an adaptive ensemble of classifiers. Furthermore, Grabner *et al.* [8] have designed an on-line boosting classifier that selects features to discriminate the object from the background. This work has demonstrated that by using on-line feature selection the tracking problem can considerably be simplified and therefore the classifiers can be quite compact and fast. For instance, Fig. 1 depicts a challenging tracking sequence, where a small textured patch is tracked using on-line boosting. Since the trackers are trained to optimally handle foreground/background discrimination, they can handle also such difficult situations where the same texture is used as background. However, when we occlude the object it is lost (since it is no longer visible) and the tracker (continuously updating its representation) starts tracking something different.

Hence, using on-line adaptation we face drifting as the key problem. Each time we make an update to our tracker an error might be introduced, resulting in a tracking error, which may accumulate over time resulting in tracking failures. Matthews *et al.* [9] have pinpointed this problem and proposed a partial solution for template trackers. Looking at this problem from a classification point of view we have the necessity to introduce a "teacher" to train the classifier. Other approaches used a geometric model (*e.g.*, homography for planar objects) for verification [10] and performed updating only when the geometric model is verified. This alleviates the drifting problem but is not applicable in all situations. Co-learning (using multiple trackers operating on different features that train each other) is another strategy proposed in [11]. Combinations of generative and discriminative models are used [12]. Both approaches alleviate

Fig. 2. Detection and tracking can be viewed as the same problem, depending on how fast the classifier adapts to the current scene. On the one side a general object detector (*e.g.*, [14]) is located and on the other side a highly adaptive tracker (*e.g.*, [4]). Our approach is somewhere in between, benefiting from both approaches: (i) be sufficiently adaptive to appearance and illumination changes and (ii) limit (avoid large) drifting by keeping prior information about the object.

the drifting problem to a certain extend but cannot avoid it. Summarizing, we can either use fixed classifiers which per definition do not suffer from the drifting problem, but have limited adaptation capabilities or we can use on-line adaptation and then have to face the drifting problem[1]. In fact, this is not a binary choice as depicted in Fig. 2.

In this paper, we explore the continuum between a fixed detector and on-line learning methods as depicted in Fig. 2. Recently, this has also been investigated by Li *et al.* [15] for tracking in low-frame rates. However, in order to formulate this problem in a principled manner we use ideas from semi-supervised learning (see [16] for a recent survey). In particular, we use the recently proposed SemiBoost [17,18] for learning a classifier. Labeled data (or a previously trained model) is used as a prior and the data collected during tracking as unlabeled samples. This allows us to formulate the tracker update problem in a natural manner. Additionally, this solves the problem of how to weight the a priori information and the on-line classifier without parameter tuning. The major contribution is an on-line formulation of semi-supervised boosting which is a requirement for using this algorithm for tracking.

Back to our example shown in Fig. 1. The proposed approach performs similar to the former on-line tracker up to the third subfigure, where both get lost. Yet, in contrast to the on-line boosting, as soon as the object becomes visible again it is re-detected by the SemiBoost tracker (using the a priori knowledge) while the on-line boosted tracker meanwhile has adapted itself to a completely different region which it finally tries to track.

The reminder of the paper is organized as follows. Section 2 shortly reviews on-line boosting for feature selection and a recently published variant of semi-supervised boosting called SemiBoost[18]. In Section 3, we present our novel on-line SemiBoosting method, which is then used in a tracking application shown in Section 4. Section 5 presents some detailed experiments and results. Finally, our work concludes with Section 6.

[1] In fact, this is another instance of the stability plasticity dilemma [13].

2 Preliminaries

2.1 Off-Line Boosting for Feature Selection

Boosting is a widely [19] used technique in machine learning for improving the accuracy of any given learning algorithm. In this work, we focus on the (discrete) AdaBoost algorithm, which has been introduced by Freund and Shapire [20]. The algorithm can be summarized as follows: given a labeled training set $\mathcal{X}^L = \{\langle \mathbf{x}_1, y_1 \rangle, ..., \langle \mathbf{x}_{|\mathcal{X}^L|}, y_{|\mathcal{X}^L|} \rangle \mid \mathbf{x}_i \in \mathbb{R}^m,\ y_i \in \{-1, +1\}\}$ with a set of m-dimensional features $\mathbf{x_i}$, positive and negative labeled samples y_i and an initial uniform distribution $p(\mathbf{x_i}) = \frac{1}{|\mathcal{X}^L|}$ over the L examples. A weak classifier h is trained using $\mathcal{X}$ and $p(\mathbf{x})$. The weak classifier has to perform only slightly better than random guessing (i.e., the error rate of a classifier for a binary decision task must be less than 50%). Depending on the error e of the weak classifier, a weight $\alpha = \frac{1}{2} \ln \left(\frac{1-e}{e}\right)$ is calculated and the probability $p(\mathbf{x})$ is updated. For misclassified samples the corresponding weight is increased while for correctly classified samples the weight is decreased. Thus, the algorithm focuses on the hard examples. Boosting greedily adds a new classifier at each boosting iteration until a certain stopping criterion is met. Finally, a strong classifier $H(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n h_n(\mathbf{x})\right)$ is calculated by a linear combination of all N trained weak classifiers. As shown by Friedman et al. [21], boosting can be viewed as additive logistic regression by stage wise minimization of the exponential loss $\mathcal{L} = \sum_{\mathbf{x} \in \mathcal{X}^L} e^{-yH(\mathbf{x})}$. Thus, a confidence measure is provided by

$$P(y = 1|\mathbf{x}) = \frac{e^{H(\mathbf{x})}}{e^{H(\mathbf{x})} + e^{-H(\mathbf{x})}}. \tag{1}$$

Furthermore, boosting can be applied for feature selection [22] where each feature corresponds to a weak classifier. In each iteration n from a set of k possible features $\mathcal{F} = \{f_1, ..., f_k\}$, a weak hypothesis is built from the weighted training samples. The best f_n is selected and forms the weak hypothesis h_n. The weights of the training samples are updated with respect to the error of the chosen hypothesis.

2.2 On-Line Boosting for Feature Selection

During on-line learning, contrary to off-line methods, each training sample is only provided once to the learner and is discarded right after learning. For that purpose, the weak classifiers have to be updated on-line every time a new training sample is available. The basic idea of on-line boosting is that the importance λ of a sample can be estimated by propagating it through a fixed set of weak classifiers [23]. The importance plays the role as the weight distribution $p(\mathbf{x_i})$ in the off-line case. In fact, λ is increased proportional to the error e of the weak classifier if the sample is still misclassified and decreased, otherwise. The error of the weak classifier $\hat{e} = \frac{\lambda^w}{\lambda^w + \lambda^c}$ is estimated by the sum of correctly λ^c and incorrectly λ^w samples seen so far.

In order to perform feature selection Grabner and Bischof [8] introduced "selectors". On-line boosting is not directly performed on the weak classifiers, but on the selectors. For that purpose, a selector $h^{sel}(\mathbf{x})$ consists of a set of M weak classifiers $\{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$. When training a selector, its M weak classifiers are trained and the one with the lowest estimated error is selected $h^{sel}(\mathbf{x}) = \arg\min_m e\left(h_m(\mathbf{x})\right)$. The AdaBoost on-line training algorithm used for feature selection works as follows: A fixed number of N selectors $h_1^{sel}, .., h_N^{sel}$ are initialized with random features. The weak classifiers in each selector are updated, as soon as a new training sample $\langle \mathbf{x}, y \rangle$ is available, and the weak classifier with the smallest estimated error is selected. For updating of the weak classifier any on-line learning algorithm is applicable. Finally, the weight α_n of the n-th selector h_n^{sel} is updated and the importance λ_n is passed to the next selector h_{n+1}^{sel}. Contrary to the off-line version, the on-line classifier is available at any time of the training process as a linear combination of the N selectors.

2.3 Off-Line Semi-supervised Boosting

Unsupervised methods are looking to find an interesting (natural) structure using only unlabeled data $\mathcal{X}^U = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{X}^U|}\}$. Semi-supervised learning uses both labeled $\mathcal{X}^L$ and unlabeled $\mathcal{X}^U$ data $\mathcal{X} = \mathcal{X}^L \cup \mathcal{X}^U$. We use the recently proposed SemiBoost approach by Mallapragada *et al.* [17] which combines ideas from graph theory and clustering and outperforms other approaches on common machine learning datasets. The basic idea is to extend the loss function with unlabeled data. In order to include unlabeled samples a similarity measure $S(\mathbf{x}_i, \mathbf{x}_j)$ has to be provided to "connect" pairs of samples. The combined loss linearly combines three individual loss functions: (i) a loss for labeled examples, (ii) labeled examples and unlabeled examples and (iii) pairs of unlabeled examples. Boosting is used to minimize the combined loss.

Following the derivation of the AdaBoost algorithm the objective function is solved in a greedy manner by stage-wise selecting the best weak classifier h_n and weight α_n, which are added to the ensemble. Formally,

$$h_n = \arg\min_{h_n} \left(\frac{1}{|\mathcal{X}^L|} \sum_{\substack{\mathbf{x} \in \mathcal{X}^L \\ h_n(\mathbf{x}) \neq y}} w_n(\mathbf{x}, y) - \frac{1}{|\mathcal{X}^U|} \sum_{\mathbf{x} \in \mathcal{X}^U} (p_n(\mathbf{x}) - q_n(\mathbf{x})) \alpha_n h_n(\mathbf{x}) \right) \quad (2)$$

$$\alpha_n = \frac{1}{4} \ln \left(\frac{\frac{1}{|\mathcal{X}^U|} \left(\sum_{\substack{\mathbf{x} \in \mathcal{X}^U \\ h_n(\mathbf{x})=1}} p_n(\mathbf{x}) + \sum_{\substack{\mathbf{x} \in \mathcal{X}^U \\ h_n=-1}} q_n(\mathbf{x}) \right) + \frac{1}{|\mathcal{X}^L|} \sum_{\substack{\mathbf{x} \in \mathcal{X}^L \\ h_n(\mathbf{x})=y}} w_n(\mathbf{x}, y)}{\frac{1}{|\mathcal{X}^U|} \left(\sum_{\substack{\mathbf{x} \in \mathcal{X}^U \\ h_n(\mathbf{x})=1}} q_n(\mathbf{x}) + \sum_{\substack{\mathbf{x} \in \mathcal{X}^U \\ h_n(\mathbf{x})=-1}} p_n(\mathbf{x}) \right) + \frac{1}{|\mathcal{X}^L|} \sum_{\substack{\mathbf{x} \in \mathcal{X}^L \\ h_n(\mathbf{x}) \neq y}} w_n(\mathbf{x}, y)} \right) \quad (3)$$

where the term $w_n(\mathbf{x}, y) = e^{-2yH_{n-1}(\mathbf{x})}$, is the weight of a labeled sample. Using $\mathcal{X}^+ = \{\langle \mathbf{x}, y \rangle | \mathbf{x} \in \mathcal{X}^L, y = 1\}$ as the set of all positive samples and $\mathcal{X}^- = \{\langle \mathbf{x}, y \rangle | \mathbf{x} \in \mathcal{X}^L, y = -1\}$ as the set of all negative samples the terms

$$p_n(\mathbf{x}) = e^{-2H_{n-1}(\mathbf{x})} \frac{1}{|\mathcal{X}^L|} \sum_{\mathbf{x}_i \in \mathcal{X}^+} S(\mathbf{x}, \mathbf{x}_i) + \frac{1}{|\mathcal{X}^U|} \sum_{\mathbf{x}_i \in \mathcal{X}^U} S(\mathbf{x}, \mathbf{x}_i) e^{H_{n-1}(\mathbf{x}_i) - H_{n-1}(\mathbf{x})}, \quad (4)$$

$$q_n(\mathbf{x}) = e^{2H_{n-1}(\mathbf{x})} \frac{1}{|\mathcal{X}^L|} \sum_{\mathbf{x}_i \in \mathcal{X}^-} S(\mathbf{x}, \mathbf{x}_i) + \frac{1}{|\mathcal{X}^U|} \sum_{\mathbf{x}_i \in \mathcal{X}^U} S(\mathbf{x}, \mathbf{x}_i) e^{H_{n-1}(\mathbf{x}) - H_{n-1}(\mathbf{x}_i)} \quad (5)$$

can be interpreted as confidences of an unlabeled sample belonging to the positive ($p_n(\mathbf{x})$) and negative class ($q_n(\mathbf{x})$), respectively. The classifier is trained in order to minimize the weighted error of the samples. For a labeled sample $\mathbf{x} \in \mathcal{X}^L$ this is the same as in common boosting the weight $w_n(\mathbf{x})$. The second term considers the distance between the unlabeled sample and the labeled samples. Each unlabeled sample $\mathbf{x} \in \mathcal{X}^U$ gets the (pseudo)-label $z_n(\mathbf{x}) = \text{sign}(p_n(\mathbf{x}) - q_n(\mathbf{x}))$ and should be sampled according to the confidence weight $|p_n(\mathbf{x}) - q_n(\mathbf{x})|$.

Summarizing, the algorithm minimizes an objective function which takes distances among semi-labeled data into account using a given similarity measure between samples. When no unlabeled data is used (*i.e.*, $\mathcal{X}^U = \{\}$) Eq. 2 and Eq. 3 reduce to the well known AdaBoost formulas. After the training, we have a strong classifier similar to standard boosting.

3 Semi-supervised On-Line Boosting for Feature Selection

3.1 Approximations of the Weights

Since sample weights code the information from one weak classifier to the next, we have to determine all these weights in an on-line setting, in order to train the n-th weak classifier h_n (Eq. 2) and calculate its associated factor α_n (Eq. 3). For labeled examples we can use the on-line boosting for feature selection approach directly. The main question is how to include the unlabeled samples. Their weights and, additionally, their labels are related to $p(\mathbf{x})$ and $q(\mathbf{x})$ defined in Eq. 4 and Eq. 5, respectively. But, these terms cannot be evaluated directly, due to the sums over pairs of either labeled and unlabeled samples. Since we are in a pure on-line setting we cannot access the whole training set. Hence, we have to use approximations.

Let us assume we have a huge ($|\mathcal{X}^U| \to \infty$) amount of unlabeled data, then the second terms in Eq. 4 and Eq. 5 will be zero. Therefore, we can skip these terms without a major loss in performance. Now, following [24,18] we learn the similarity $S(\mathbf{x}_i, \mathbf{x}_j) \approx H^{sim}(\mathbf{x}_i, \mathbf{x}_j)$ by a classifier using boosting. Furthermore, we only have to sum over the similarity of the current (unlabeled) sample $\mathbf{x}$ and the set of positive or negative samples. Given the labeled samples in advance, we can train a classifier a-priori which measures the similarity, to the positive or negative class. For a positive sample this can be approximated by learning a classifier which describes the positive class $\sum_{\mathbf{x}_i \in \mathcal{X}^+} H^{sim}(\mathbf{x}, \mathbf{x}_i) \approx H^+(\mathbf{x})$, *i.e.*, provides a probability measure that $\mathbf{x}$ corresponds to the positive class. In the same manner, a classifier is built for the negative class $\sum_{\mathbf{x}_i \in \mathcal{X}^-} H^{sim}(\mathbf{x}, \mathbf{x}_i) \approx H^-(\mathbf{x})$. Instead of learning two generative classifiers we learn one discriminative classifier $H^P(\mathbf{x})$ which distinguishes the two classes, *i.e.*, $H^+(\mathbf{x}) \sim H^P(\mathbf{x})$ and $H^-(\mathbf{x}) \sim 1 - H^P(\mathbf{x})$. Since we use boosting to learn such a prior classifier, it can be translated into a probability using Eq. 1. We can now approximate Eq. 4 and Eq. 5 as

$$\tilde{p}_n(\mathbf{x}) \approx e^{-H_{n-1}(\mathbf{x})} \sum_{\mathbf{x}_i \in \mathcal{X}^+} S(\mathbf{x}, \mathbf{x}_i) \approx e^{-H_{n-1}(\mathbf{x})} H^+(\mathbf{x}) \approx \frac{e^{-H_{n-1}(\mathbf{x})} e^{H^P(\mathbf{x})}}{e^{H^P(\mathbf{x})} + e^{-H^P(\mathbf{x})}}, \quad (6)$$

$$\tilde{q}_n(\mathbf{x}) \approx e^{H_{n-1}(\mathbf{x})} \sum_{\mathbf{x}_i \in \mathcal{X}^-} S(\mathbf{x}, \mathbf{x}_i) \approx e^{H_{n-1}(\mathbf{x})} H^-(\mathbf{x}) \approx \frac{e^{H_{n-1}(\mathbf{x})} e^{-H^P(\mathbf{x})}}{e^{H^P(\mathbf{x})} + e^{-H^P(\mathbf{x})}}, \quad (7)$$

where we discard the factor 2 since we do not include pairs of unlabeled to unlabeled samples. Since we are interested in the difference, we finally get the "pseudo-soft-label"

$$\tilde{z}_n(\mathbf{x}) = \tilde{p}_n(\mathbf{x}) - \tilde{q}_n(\mathbf{x}) = \frac{\sinh(H^P(\mathbf{x}) - H_{n-1})}{\cosh(H^P(\mathbf{x}))} = \tanh(H^P(\mathbf{x})) - \tanh(H_{n-1}(\mathbf{x})). \quad (8)$$

Algorithm 1. On-line Semi-supervised Boosting for feature selection

Require: training (labeled or unlabeled) example $\langle \mathbf{x}, y \rangle$, $x \in \mathcal{X}$
Require: prior classifier H^P (can be initialized by training on $\mathcal{X}^L$)
Require: strong classifier H (initialized randomly)
Require: weights $\lambda^c_{n,m}$, $\lambda^w_{n,m}$ (initialized with 1)
for $n = 1, 2, .., N$ **do** // for all selectors
  if $x \in \mathcal{X}^L$ **then** //set weight and target of the sample
    $y_n = y$, $\lambda_n = \exp(-yH_{n-1}(\mathbf{x})$
  else
    $y_n = \text{sign}(p(\mathbf{x}) - q(\mathbf{x}))$, $\lambda_n = |p(\mathbf{x}) - q(\mathbf{x})|$ //set pseudo label
  end if
  for $m = 1, 2, .., M$ **do** // update the selector h^{sel}_n
    $h_{n,m} = \text{update}(h_{n,m}, \langle \mathbf{x}, y \rangle, \lambda)$ // update each weak classifier
    // estimate errors
    if $h^{weak}_{n,m}(\mathbf{x}) = y$ **then**
      $\lambda^c_{n,m} = \lambda^c_{n,m} + \lambda_n$
    else
      $\lambda^w_{n,m} = \lambda^w_{n,m} + \lambda_n$
    end if
    $e_{n,m} = \frac{\lambda^w_{n,m}}{\lambda^c_{n,m} + \lambda^w_{n,m}}$
  end for
  // choose weak classifier with the lowest error
  $m^+ = \arg\min_m(e_{n,m})$, $e_n = e_{n,m^+}$, $h^{sel}_n = h_{n,m^+}$
  if $e_n = 0$ or $e_n > \frac{1}{2}$ **then**
    exit
  end if
  $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$ // calculate voting weight
end for

3.2 The On-Line Algorithm

It is now straight forward to extend SemiBoost to on-line boosting [8]. For the labeled examples $\langle \mathbf{x}, y \rangle, x \in X^L$ ($y \in \{-1, +1\}$) nothing changes. For each unlabeled sample ($x \in \mathcal{X}^U$) after each selector not only the weight (the importance λ_n) is adapted, but also its estimated target y_n may change. Hence, for unlabeled samples in each selector n, we set

$$y_n = \text{sign}(\tilde{z}_n(\mathbf{x})) \text{ and } \lambda_n = |\tilde{z}_n(\mathbf{x})|, \quad (9)$$

where $\tilde{z}_n(\mathbf{x})$ is defined in Eq. 8. Summarizing, by training a prior classifier H^P from labeled samples a-priori provided, it is possible to include unlabeled data into the on-line boosting framework using pseudo-labels and pseudo-importances. Our semi-supervised boosting algorithm for feature selection is sketched in Algorithm 1. Compared to the original on-line boosting algorithm [8] only a few lines of code (highlighted lines 2-6) have to be changed in order to cope with unlabeled data.

Let us take a look at the pseudo-labels when propagating the unlabeled sample $\mathbf{x}$ through the selectors. If the prior is very confident it dictates the label. A label switch can happen, *i.e.*, $H(\mathbf{x})$ can overrule $H^P(\mathbf{x})$, if $\tilde{z}_n(\mathbf{x})$ has a differnt label as the prior $H^P(\mathbf{x})$. As can be easily seen from Eq. 8, this is the case if $|H_n| > |H^P|$. Therefore, the more confident the prior is, the longer (with respect to n) the label is not allowed to change. We do not make any statements whether this is a correct or incorrect label switch. Note, that the prior classifier can be wrong, but it has to provide a "honest" decision. Meaning, if it is highly confident it must be ensured to be a correct decision[2].

4 Robust Tracking

We first briefly review the on-line boosting tracker that is based on on-line boosting for feature selection [8,4], which is replaced by our proposed on-line SemiBoost algorithm.

The basic idea is to formulate tracking as binary classification problem between the foreground object, which has to be tracked, and the local background. Assuming the object has been detected in the first frame, an initial classifier is built by taking positive samples from the object and randomly chosen negative ones from the background. The tracking loop consists of the following steps. From time t to $t+1$ the classifier is evaluated exhaustively pixel by pixel in the local neighborhood. Since the classifier delivers a response which is equivalent to the log-likelihood ratio $H(\mathbf{x}) = \frac{1}{2}\log\left(\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})}\right)$ (see Eq. 1), the confidence distribution is analyzed and in the simplest case the local maximum is considered to be the new object position. In order to robustly find the object in the next frame and thus adapt to appearance changes of the object, different lightning conditions or background changes, the classifier gets updated. A positive update is taken for the patch where the object is most likely to be and negative updates are drawn from the local neighborhood.

4.1 Modifications of the Tracking Loop

The tracker, as reviewed above, can suffer from the drifting problem. This is due to self-learning relies on its own predictions that are always incorporated with hard labels (*i.e.*, $y \in \{-1, +1\}$), even if their confidences are very low. In

[2] There are also relations to the co-training [25] assumptions, *i.e.*, a classifier should be never "confident but wrong".

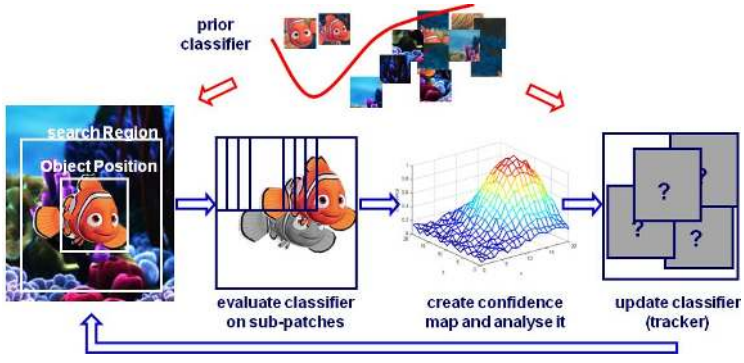

Fig. 3. Given a fixed prior and an initial position of the object in time t, the classifier is evaluated at many possible positions in a surrounding search region in frame $t + 1$. The obtained confidence map is analyzed in order to estimate the most probable position and finally the tracker (classifier) is updated in an unsupervised manner, using randomly selected patches.

contrast, incorporating our novel way of on-line semi-supervised boosting allows us to change the update strategy of the previously proposed on-line boosting tracker. The overall work flow is depicted in Figure 3, which is very similar to the one described above. The main difference is, that we do not update the classifier with fixed labels, we solely use (random) patches from the region of the estimated object position and use them as unlabeled samples to update the classifier. This is only possible because we have a prior classifier. Roughly speaking, one can think of the prior classifier as a fixed point and the on-line classifier exploring the space around it. This means that the classifier can adapt (or "drift") to new situations but has always the possibility to recover.

5 Experiments and Discussion

In this section, first, we perform experiments demonstrating the specific properties of our tracking approach. Second, we evaluated our tracker on different scenarios showing that we can cope with a large variability of different objects.

As image features which are selected by on-line SemiBoost we use Haar-like features [14] which can be calculated efficiently using integral data-structures. The performance (speed) depends on the size of the search region which we have defined by enlarging the target region by one third of the object size in each direction (for this region the integral representation is computed). In our experiments we neither use a motion model nor a scaled search window, which both however can be incorporated quite easily. The strong classifier consists of only 25 selectors each with a feature pool of 50 weak classifier. All experiments are performed on a common 2.0 GHz PC with 2 GB RAM, where we achieve 25 fps tacking speed.

5.1 Illustrations

We illustrate details of our tracker on frontal faces. As prior classifier and for initialization of the tracking process we take the default frontal face detector from OpenCV Version 1.0[3]. This demonstrates that we can use any prior in our method. The primary focus of the experiments is to compare the SemiBoost tracker with other combination methods for the prior and the on-line method[4]. As can be seen from Fig. 4, our approach (second row) significantly outperforms the on-line booster, the prior classifier and a heuristic combination of prior and on-line booster (first row). Additionally, even if the prior has very low confidence (third row), the tracker is still able to correctly follow the (side) face. This shows that we can adapt to appearance changes.

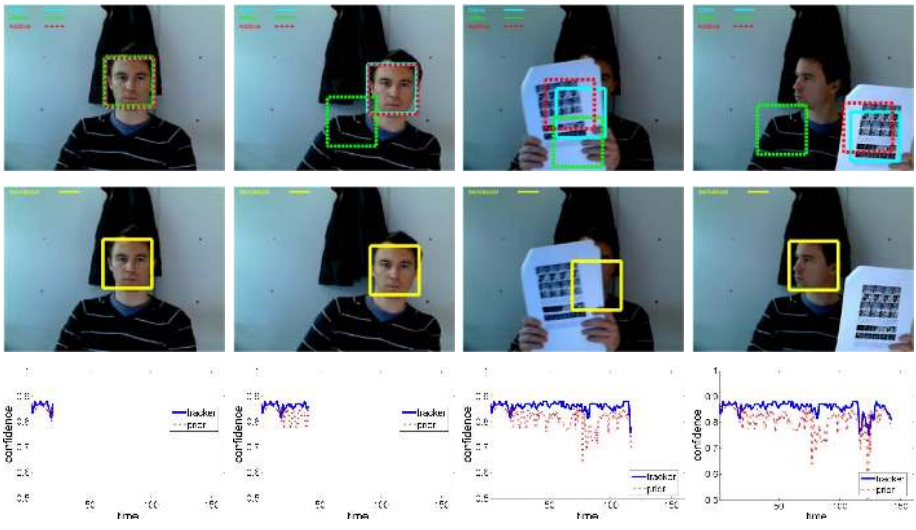

Fig. 4. Tracking a face in an image sequence under various appearance changes. The first row illustrates three different types of update strategies for the tracker, *i.e.*, (i) on-line boosting (cyan), (ii) prior classifier (red) and (iii) a heuristic combination of (i) and (ii) using the sum-rule, *i.e.*, $0.5(H^P(\mathbf{x}) + H(\mathbf{x}))$ (green). The second row shows the SemiBoost tracker using the same off-line prior. The last row depicts confidence values of the tracked patch over time for the prior and the SemiBoost tracker, respectively.

Fig. 5 depicts some illustrative samples taken for updates for both the on-line tracker and our tracker. As can be observed, while both approaches track the same object, they incorporate totally different updates. After some time the on-line booster performs wrong updates with still high confidence. This is the main reason for drifting. Furthermore, in the SemiBoost method both sample labels

[3] `http://sourceforge.net/projects/opencvlibrary/`, 2008/03/16.
[4] The OpenCV detector fails on side looking faces.

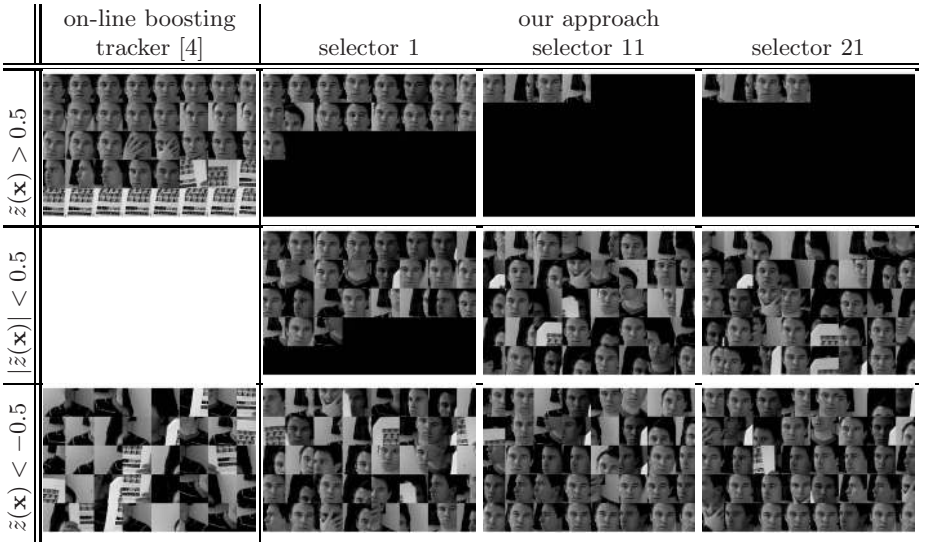

Fig. 5. Typical updates used for the former on-line boosting tracker (first column). If the tracker loses the object and due to the self-learning update strategy which delivers hard updates, it focuses on another image region. The remaining columns show how samples are incorporated by the SemiBoost tracker. While they are propagated through the selectors, their importance and label can change, with respect to the prior.

and sample weights change while propagating through the selectors of the SemiBoost tracker while being constant for the former approach. Only such samples are incorporated which are necessary to augment the prior knowledge or invert it in order to be adaptive. Positive samples are inherently treated with caution, *i.e.*, only few positive examples are considered.

5.2 One-Shot Prior Learning

For these experiments, the prior is learned from the first frame only. In fact, we build a trainingset $\mathcal{X}_P = \langle \mathbf{x}_o, +1 \rangle \cup \{ \langle \mathbf{x}_i, -1 \rangle | \mathbf{x}_i \neq \mathbf{x}_o \}$ where $\mathbf{x}_o$ corresponds to the marked image region and negative samples are generated from the local neighborhood[5]. Since this trainingset is quite small the time needed to train the prior classifier H^P is negligible. After this one-shot training, the prior classifier is kept constant.

In Fig. 6, we compare our new method to the on-line boosting approach on various tracking scenarios. First, as can be seen in row 1, we are still able to handle challenging appearance changes of the object. Row 2 of Fig. 6 depicts tracking during a fast movement. Since some incorrect updates and the self-learning strategy

[5] Also some invariance can be included in the training set, *e.g.*, by adding "virtual" samples [26] in order to train a more robust classifier.

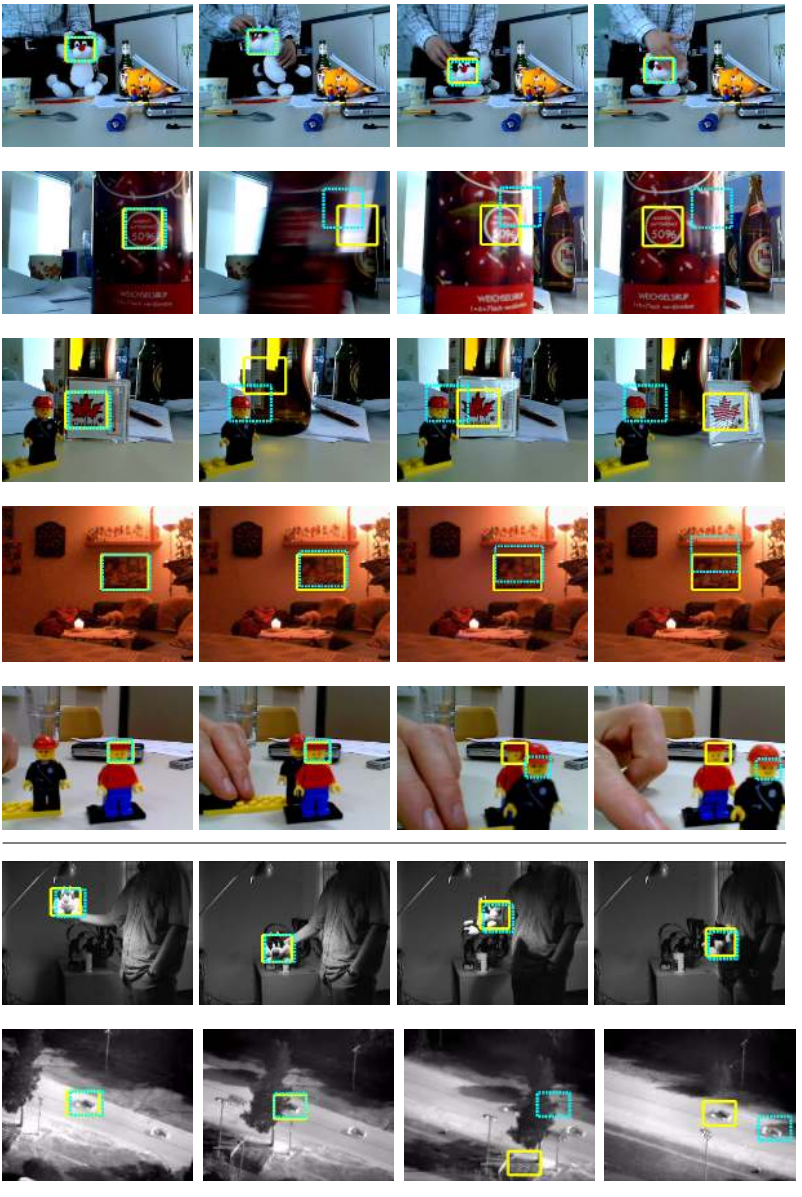

Fig. 6. Comparisons of our proposed SemiBoost tracker (yellow) and the previously proposed on-line tracker (dotted cyan). Our approach is still able to adapt to appearance changes while limiting the drinfing. Additionally, results on two public sequences are shown (last two rows). The first sequence have been provided by Lim and Ross ([6]) and the second sequence is taken from the VIVID-PETS 2005 dataset[6].

[6] `http://www.vividevaluation.ri.cmu.edu/datasets/datasets.html`, 2007/06/12

of the on-line boosting tracker loses the target and focuses on another part while the semi-supervised tracker is able to re-detect the object. An extremal case is shown in row 3, where we remove the object from the scene. If the object is present again and thanks to the fixed prior our proposed approach has not forgotten the appearance as it is the case for the other tracker and snap to the object again. The next experiment (row 4) focuses on the long term behavior. We chose to track a non-moving object in a static scene for about 1 hour. In order to emphasize the effect we use rather dark illumination conditions. While our proposed tracker stays at the object, the on-line booster starts to drift away. The reason is the accumulation of errors. The final experiment shows a special case of drifting as depicted in the last row of Fig. 6. Two very similar objects are put together in the scene. Since the pure on-line tracker has not the additional prior information, it is very likely that it is unstable and may switch to another object. Additionally, we choose two public available tracking sequences which have been already used in other publications as can be seen in the last two rows. Our approach performs comparable to the previous on-line tracker on appearance changes (sixth row). After the object was totally occluded (last row), our approach is able to recover the correct object while the former on-line tracker gets confused and starts tracking the second (wrong) car. Additional tracking videos are included as supplementary material.

6 Conclusions

In this paper, we have presented a tracker which limits the drifting problem while still being adaptive to various appearance changes which arise in typical real world scenarios. We have employed ideas from semi-supervised learning and on-line boosting for feature selection. The so trained on-line classifier is used in a tracking framework in order to discriminate the object from the background. The knowledge from labeled data can be used to build a fixed prior for the on-line classifier. In order to still be adaptive, during tracking unlabeled data is explored in a principled manner. Furthermore, our approach does not need parameter tuning and is easy to implement. We have demonstrated successful tracking of different objects in real-time on various challenging sequences.

References

1. Avidan, S.: Support vector tracking. IEEE Trans. PAMI 26, 1064–1072 (2004)
2. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: Proc. CVPR, vol. 2, pp. 775–781 (2005)
3. Özuysal, M., Fua, P., Lepetit, V.: Fast keypoint recognition in ten lines of code. In: CVPR (2007)
4. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: Proc. BMVC, vol. 1, pp. 47–56 (2006)
5. Collins, R., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Trans. PAMI 27(10), 1631–1643 (2005)

6. Lim, J., Ross, D., Lin, R., Yang, M.: Incremental learning for visual tracking. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) NIPS, vol. 17, pp. 793–800. MIT Press, Cambridge (2005)
7. Avidan, S.: Ensemble tracking. In: Proc. CVPR, vol. 2, pp. 494–501 (2005)
8. Grabner, H., Bischof, H.: On-line boosting and vision. In: Proc. CVPR, vol. 1, pp. 260–267 (2006)
9. Matthews, I., Ishikawa, T., Baker, S.: The template update problem. IEEE Trans. PAMI 26, 810–815 (2004)
10. Grabner, M., Grabner, H., Bischof, H.: Learning features for tracking. In: Proc. CVPR (2007)
11. Tang, F., Brennan, S., Zhao, Q., Tao, H.: Co-tracking using semi-supervised support vector machines. In: Proc. ICCV, pp. 1–8 (2007)
12. Woodley, T., Stenger, B., Cipolla, R.: Tracking using online feature selection and a local generative model. In: Proc. BMVC (2007)
13. Grossberg, S.: Competitive learning: From interactive activation to adaptive resonance. Neural networks and natural intelligence, 213–250 (1998)
14. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proc. CVPR, vol. I, pp. 511–518 (2001)
15. Li, Y., Ai, H., Yamashita, T., Lao, S., Kawade, M.: Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In: Proc. CVPR, pp. 1–8 (2007)
16. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
17. Mallapragada, P.K., Jin, R., Jain, A.K., Liu, Y.: Semiboost: Boosting for semi-supervised learning. Technical report, Department of Computer Science and Engineering, Michigan State University (2007)
18. Leistner, C., Grabner, H., Bischof, H.: Semi-supervised boosting using visual similarity learning. In: Proc. CVPR (to appear, 2008)
19. Schapire, R.: The boosting approach to machine learning: An overview. In: Proceedings MSRI Workshop on Nonlinear Estimation and Classification (2001)
20. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
21. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28(2), 337–407 (2000)
22. Tieu, K., Viola, P.: Boosting image retrieval. In: Proc. CVPR, pp. 228–235 (2000)
23. Oza, N., Russell, S.: Online bagging and boosting. In: Proceedings Artificial Intelligence and Statistics, pp. 105–112 (2001)
24. Hertz, T., Bar-Hillel, A., Weinshall, D.: Learning distance functions for image retrieval. In: Proc. CVPR, vol. 2, pp. 570–577 (2004)
25. Balcan, M.F., Blum, A., Yang, K.: Co-training and expansion: Towards bridging theory and practice. In: NIPS. MIT Press, Cambridge (2004)
26. Girosi, F., Chan, N.: Prior knowledge and the creation of virtual examples for rbf networks. In: IEEE Workshop on Neural Networks for Signal Processing (1995)