Speeding Up Semi-Supervised On-line Boosting for Tracking*

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

[1] Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{*godec,leistner,bischof*}*@icg.tugraz.at*

[2] Computer Vision Laboratory,
ETH Zurich, Switzerland
grabner@vision.ee.ethz.ch

Abstract
*Recently, object tracking by detection using adaptive on-line classifiers has been investigated. In this case, the tracking problem is reduced to the discrimination of the current object view from the local background. However, on-line learning may introduce errors, which causes drifting and let the tracker fail. This can be avoided by using semi-supervised on-line learning (*i.e.*, the use of labeled and unlabeled training samples), which allows to limit the drifting problem while still staying adaptive to appearance changes, in order to stabilize tracking. In particular, this paper extends semi-supervised on-line boosting by a particle filter to achieve a higher frame-rate. Furthermore, a more sophisticated search-space sampling, and an improved update sample selection have been added. In addition, a review of the semi-supervised on-line boosting algorithm is given and further experiments have been accomplished.*

1 Introduction

Within the last decades, object tracking has obtained much attention in computer vision. The design of robust tracking methods is still an open issue, especially considering the complicated variations that may occur in natural scenes (*e.g.*, shape and appearance changes of the object, illumination variations, partial occlusions, cluttered scenes, *etc.*). Recently, tracking by detection has been investigated, where the object has to be discriminated from the background in each frame. The advantage of such an approach is the increased speed since the time-consuming training is done before tracking. However, using off-line training limits the classifier to variations of the object that are already included within the training set, which requires the object to be known before.

To cope with variations of the object that are not known a priori, the tracker needs to be adaptive. Hence, different methods using on-line adaption for visual tracking have been proposed [2, 17, 4], with variation in the used object representations and learning methods. Grabner *et al.* [8] proposed an on-line boosting algorihm for feature selection, that demonstrated excellent real-time performance on the tracking task.

*This work was supported by the FFG project EVis (813399) under the FIT-IT program and the EU-project SCOVIS (216465), and the Austrian Science Fund (FWF) under the doctoral program Confluence of Vision and Graphics W1209.

However, using un-supervised on-line training each time the classifier is updated an error may be introduced into the learned object model. Matthews *et al.* [19] have pinpointed this problem as the *template update problem* and proposed a partial solution to the problem by dropping update samples that differ too much from the actual object model. Other approaches use additional knowledge to limit drifting (*e.g.*, geometric model for homography verification [11], combination of generative and discriminative models [24], or co-learning of trackers using different types of features [23]). But even if these methods can alleviate this problem, it cannot be avoided. The balance between a static detector (to avoid drifting) and an adaptive classifier (implicitly accepting that drifting may occur) is directly related to the stability plasticity dilemma [13].

While tracking is an un-supervised process, previous approaches [2, 8] used supervised learning methods for training the classifier continuously. To overcome this discrepancy, semi-supervised learning [3, 25, 21] can be used. This allows the integration of manually labeled training samples as well as unlabeled samples that are retrieved during the tracking process. This problem has recently been addressed by Grabner *et al.* [9] using semi-supervised learning ideas recently proposed by Mallapragada *et al.* [18] and Leistner *et al.* [14]. The integration of prior knowledge into the tracking process is done without parameter-tuning within their approach.

The major motivation of this paper is to first give a review of the recently proposed semi-supervised on-line boosting approach of Grabner *et al.* [9] and, second, to propose several improvements of the method in terms of both speed and accuracy. Therefore, we propose to use particle filtering for smarter search-space sampling leading to increased speed and improved update patch selection resulting in higher accuracy.

The reminder of this paper is organized as follows. Section 2 gives an introduction to the basic techniques and a detailed review and discussion of the semi-supervised on-line boosting algorithm. Section 3 presents a novel search-space sampling method within this tracking framework based on particle filtering. Section 4 shows results and evaluations done with the implemented tracking framework, and finally Section 5 gives a conclusion and an outlook to further research.

2 Review of Semi-Supervised On-line Boosting for Tracking

Within this section, we give an introduction and review to the used methods and the application of on-line semi-supervised boosting to tracking.

2.1 Semi-Supervised On-line Boosting

Boosting is a popular machine learning technique for improving the accuracy of any given learning algorithm widely used in computer vision [22]. In this paper, we mainly focus on the discrete Adaboost algorithm [5]. Several weak classifiers $h_n(\mathbf{x})$ are trained sequentially using a weight distribution on labeled samples and additively combined to a strong classifier $H(\mathbf{x})$ in the form

$$H(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n h_n(\mathbf{x}), \tag{1}$$

with weights α_n corresponding to the weak classifiers error. The sample weights are adapted after each iteration according to the error of the added weak classifier in order to focus on samples that are hard to learn. $H(\mathbf{x})$ is a large margin classifier yielding confidence-rated predictions.

Semi-Supervised Boosting

In contrast to supervised or un-supervised learning methods, semi-supervised learning uses both labeled $\mathcal{X}^L$ and unlabeled $\mathcal{X}^U$ samples, where $\mathcal{X} = \mathcal{X}^L \cup \mathcal{X}^U$. Mallapragada *et al.* [18] recently proposed a semi-supervised approach for boosting. In particular their method is inspired by previous graph-based approaches [25] where the similarity $S(\mathbf{x}_a, \mathbf{x}_b)$ between samples is measured, resulting in a loss for labeled examples, labeled and unlabeled examples, and pairs of unlabeled examples.

Boosting now solves the objective function in a greedy manner by stage-wise selecting the best weak classifier h_n with weight α_n and adding them to the ensemble H (see also [9]). Formally,

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

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

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

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

and

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

can be interpreted as confidences of an unlabeled sample belonging to the positive (Eq. 4) and negative class (Eq. 5), 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 with 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 (3) reduce to the well known AdaBoost formulation. After the training, we have a strong classifier similar to standard boosting.

Approximations for On-line Processing

Contrary to off-line methods, during on-line learning each training sample is provided only once to the learner. Oza and Russell [20] proposed an on-line version for boosting. They model the sequential incoming of the samples using a Poisson distribution and compute the importance λ of

a sample by propagating it through the set of weak classifiers. Later, Grabner *et al.* [8] introduced *selectors* in order to allow for on-line feature selection. On-line boosting is performed on a set of N selectors and not directly on the hypotheses space. A selector $h_n^{sel}(\mathbf{x})$ holds a set of M weak classifiers $\{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$ that are related to a subset of features $\mathcal{F}_n = \{f_1, \ldots, f_M\} \in \mathcal{F}$, where $\mathcal{F}$ is the full feature pool. At each time the selector $h_n^{sel}(\mathbf{x})$ selects the best weak hypothesis $h^{sel}(\mathbf{x}) = \arg\min_m e\left(h_m(\mathbf{x})\right)$ according to the estimated training error $\hat{e} = \frac{\lambda_{wrong}}{\lambda_{wrong}+\lambda_{corr}}$ where λ_{corr} and λ_{wrong} are the importance weights of the samples seen so far that were classified correctly and incorrectly, respectively.

For the purpose of an adaptive on-line object tracker, we have to modify the semi-supervised boosting approach to fit into the on-line feature selection mechanism (see [9] for full derivation). Especially terms that measure the similarity between all samples are not applicable within on-line processing; $p_n(\mathbf{x})$ and $q_n(\mathbf{x})$ include terms that make use of the whole data-set; thus they have to be approximated. But since the number of unlabeled examples grows toward infinity $\left|\mathcal{X}^U\right| \rightarrow \infty$, the second term in $p_n(\mathbf{x})$ and $q_n(\mathbf{x})$ (Eq. (4) and (5)) tends toward zero, assuming that most samples will have rather small similarity. Thus, the similarity measurement among unlabeled samples has been eliminated. By this, also the regularization term $\frac{1}{|\mathcal{X}^L|}$ can be eliminated, since no weighting between labeled and unlabeled samples is needed. The measurment between labeled and unlabeled samples, the first terms of $p_n(\mathbf{x})$ and $q_n(\mathbf{x})$, are approximated using a statically learned classifier. Instead of learning two separate classifiers $H^+(\mathbf{x})$ and $H^-(\mathbf{x})$, this classifier $H^P(\mathbf{x})$ directly distinguishes between the positive and negative class X^+ and X^- with $H^+(\mathbf{x}) \approx H^P(\mathbf{x})$ and $H^-(\mathbf{x}) \approx 1 - H^P(\mathbf{x})$. This classifier description is plugged into the original equations instead of $\sum_{\mathbf{x}_i \in X^+} S(\mathbf{x}, \mathbf{x}_i)$ and $\sum_{\mathbf{x}_i \in X^-} S(\mathbf{x}, \mathbf{x}_i)$.

Now p_i and q_i can be simplified to

$$\tilde{p}_n(\mathbf{x}) \approx e^{-2H_{n-1}(\mathbf{x})} \sum_{\mathbf{x}_i \in X^+} S(\mathbf{x}, \mathbf{x}_i) \approx e^{-H_{n-1}(\mathbf{x})} H^+(\mathbf{x}) \approx \frac{e^{-H_{n-1}(\mathbf{x})} e^{H^P(\mathbf{x})}}{e^{H^P(\mathbf{x})} + e^{-H^P(\mathbf{x})}} \tag{6}$$

$$\tilde{q}_n(\mathbf{x}) \approx e^{2H_{n-1}(\mathbf{x})} \sum_{\mathbf{x}_i \in X^-} S(\mathbf{x}, \mathbf{x}_i) \approx e^{H_{n-1}(\mathbf{x})} H^-(\mathbf{x}) \approx \frac{e^{H_{n-1}(\mathbf{x})} e^{-H^P(\mathbf{x})}}{e^{H^P(\mathbf{x})} + e^{-H^P(\mathbf{x})}}. \tag{7}$$

The difference between this two terms can be simplified to a so called pseudo-soft label

$$\tilde{z}_n(\mathbf{x}) = \tilde{p}_n(\mathbf{x}) - \tilde{q}_n(\mathbf{x}) = \tanh(H^P(\mathbf{x})) - \tanh(H_{n-1}(\mathbf{x})) \tag{8}$$

which is used for setting the label of the training examples in the different classifier stages and can easily be integrated into the on-line supervised boosting for feature selection algorithm [8].

Discussion

In contrast to the supervised on-line algorithm, the label and the weight of a training example is determined by combination of the decision of the prior classifier H^P and the decision of the classifier until the previous stage H_{n-1} in the semi-supervised case. Thus, the focus of the classifier stages is moved from samples that have been misclassified by the previous stages to *samples where the on-line classifier and the prior classifier disagree*.

The algorithm eliminates the main discrepancy within the pure on-line tracking framework, namely the unsupervised updates for a supervised learning algorithm. Due to the semi-supervised nature of

the algorithm, it can handle labeled and unlabeled data, which makes it much easier to incorporate training samples, both labeled and unlabeled ones. The power of the update mechanism can be traced back to the adaption of the weight and label of the training sample. This impedes updates that are not well-aligned or incorrect to be learned with a high weight. Another benefit of the algorithm is the incorporation of any kind of prior knowledge that provides a confidence measure for a presented training sample without parameter-tuning.

The boosting optimization in the semi-supervised case tries to reach a minimal distance between the confidence of the prior classifier and the confidence of the semi-supervised on-line classifier. This constrains the adaptivity of the semi-supervised classifier by limiting the confidence of the semi-supervised classifier to the confidence of the prior knowledge as a maximum. The upper limit for the confidence prevents the on-line classifier from over-fitting to samples that are not ideal positive or negative samples from the view-point of the prior classifier. On the other hand, the dynamics of the updated classifier is extended, since weighting is extended to negative values. This is also needed to un-learn over-fitting samples as mentioned quite before. In comparison to the supervised sample weighting, where most samples gain very low weight after a few selector stages, label switching and high dynamics of weights frequently occurs within semi-supervised on-line boosting. The mechanism can also be interpreted as *co-training* [16], where one classifier is kept fixed to limit drifting and to constrain the adaptivity of the other classifier.

In comparison to the on-line variant, off-line semi-supervised boosting [18] suffers from quadratic runtime caused by the similarity measure between each pair of samples. This term has been eliminated within the proposed on-line algorithm [14], which enables processing of large datasets. Though the off-line algorithm calculates the optimal solution for the given training data and feature set, the on-line algorithm converges to the solution of the off-line algorithm with $n \rightarrow \infty$.

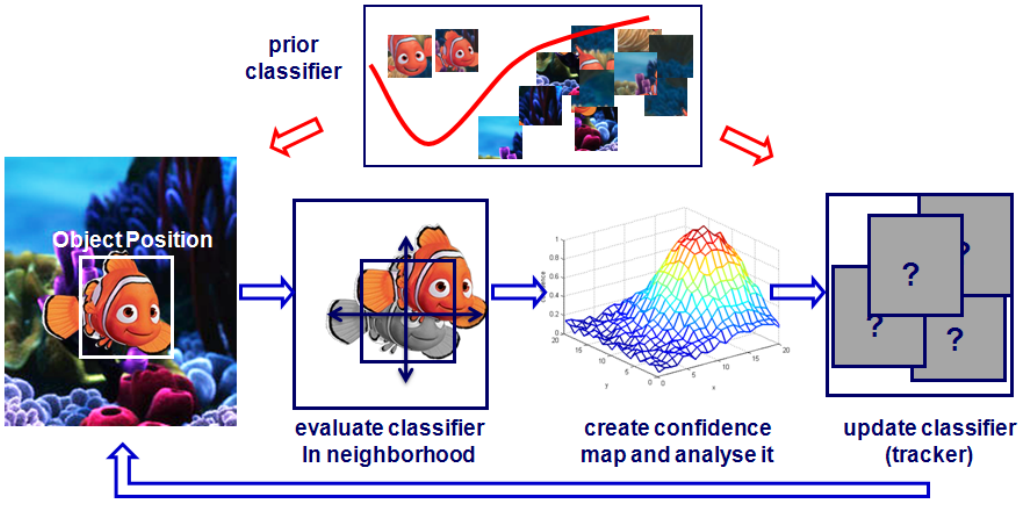


Figure 1: Tracking Loop: Semi-supervised learning allows use of unlabeled data within the updating process (taken from [9]).

2.2 Application to Tracking

A major problem of *tracking-by-detection* with on-line adaption is the accumulation of errors introduced within the adaption phase. This *label jitter*, *i.e.*, wrong sample alignment, and *label noise* is the main reason for drifting trackers. As we can now use labeled and unlabeled samples for updates,

we can use assign a label to initial samples and use samples that are learned incrementally as unlabeled ones, letting the algorithm inherently select an adequate weight and label. Figure 1 shows the differences in the update phase of the tracking loop, regarding to a supervised or a semi-supervised update scheme. This change in the learning strategy improves the insensitivity of the tracker to label noise and jitter with the cost of limited adaptability to large changes in object appearance.

3 Speeding up with Particle Filtering

Real-time performance is a critical requirement for tracking approaches. The most time-consuming parts of the tracking loop are related to the learning method (speed up with, *e.g.*, [10]) and the evaluation phase, where the search-space is sampled and evaluated. For tracking, typically several degrees of freedom of the target object, *e.g.*, rotation and scaling or affine transformations have to be used. This makes exhaustive search computational infeasible and arises the demand for a better search-space sampling method. Within tracking, particle filtering can be used to predict the whole state of the object including location, rotation, and scaling.

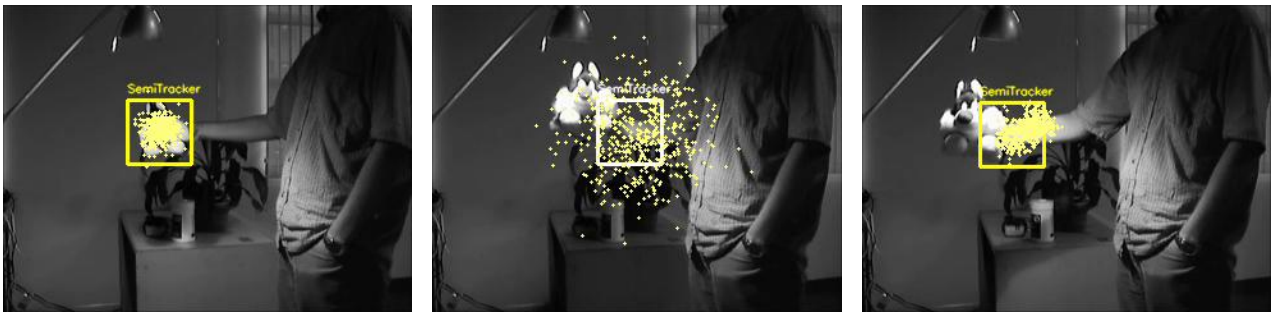
Particle filtering [1] is a method, which can be used to effectively estimate the state of a system using a sequence of noisy measurements $\mathbf{z}$ according to a set of N_P weighted particles $\{\mathbf{x}^i_{1:k}, \omega^i_{1:k}\}$ with $\sum_{i=1}^{N_P} \omega^i_k = 1$ and time $k \in \{1, \ldots, t\}$ represented as $1 : k$ within the following equations. The posterior density $p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right)$ can be estimated using the observation model $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$, the state transition model $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)$ and the proposal density function $q\left(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k\right)$ using Equation (9) and (10):

$$\omega^i_k \propto \omega^i_{k-1} \frac{p\left(\mathbf{z}_k|\mathbf{x}^i_k\right) p\left(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}\right)}{q\left(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k\right)} \tag{9}$$

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_P} \omega^i_k \delta\left(\mathbf{x}_k - \mathbf{x}^i_k\right). \tag{10}$$

Choosing the importance density to be the prior $q\left(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k\right) = p\left(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}\right)$ reduces Equation 9 to $\omega^i_k \propto \omega^i_{k-1} p\left(\mathbf{z}_k|\mathbf{x}^i_k\right)$, where the particle weights are directly proportional to the observation model. These formulations allow to iteratively estimate the posterior distribution using only the actual measurements and the last object state density, which is given by the finite set of particles, where each particle simulates the object behavior using Monte-Carlo simulations and a motion model. To avoid the degeneracy of the particle set, resampling of the weights is done regularly.

The state space of the particle filter has been limited to simple translation. A simple auto-regressive model is applied to the particles, which thereby predicts the object's motion. The decreased number of samples that have to be evaluated by the classifier in each frame is directly related to decreased runtime per frame. With a sufficient number of particles (*i.e.*, 500), the execution time is reduced by a factor of $\frac{1}{2}$ (see Section 4.1 for detailed results), which frees resources that can be used for further improvements. However, the main advantage of the particle filter is the smarter way of sampling the search space. The particle movement is based on motion information captured during the runtime of the application, regarding on the complexity of the used motion model.



(a) Tracking a moving object (b) Object lost, particles spread (c) False detection

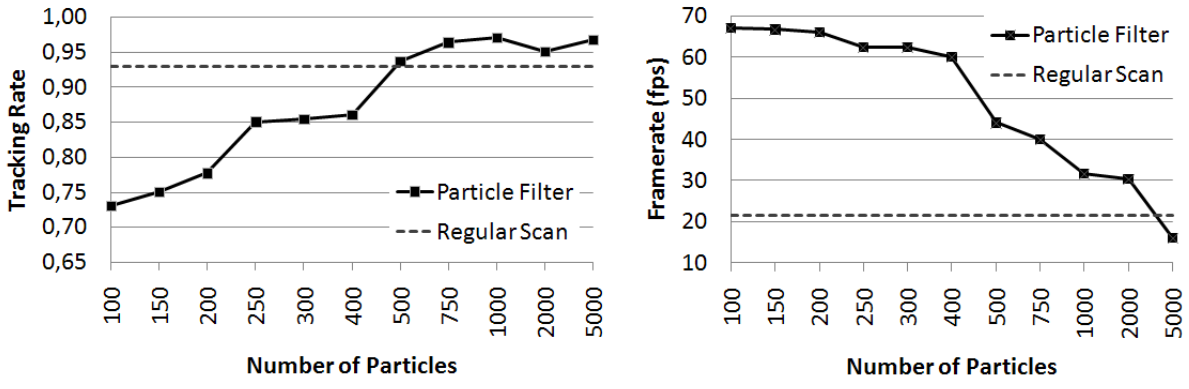
Figure 2: Tracking with particle filtering.

4 Evaluation and Results

This section presents different evaluations of the semi-supervised on-line boosting algorithm. Since results of the functionality and the improvements have already been shown in [9] and within the according sections, the main focus lies on experiments evaluating the improvements we proposed within this paper.

4.1 Speedup of Particle Filtering

Since the main focus of the paper is the speed improvement, an evaluation of speedup versus performance loss of the particle filter depending on the number of particles is given. To measure the overall tracking success, the ratio of successfully tracked frames to the sequence length is given as tracking rate. Figure 3 (a) shows that a number of 500 particles is sufficient to reach the same tracking performance as using exhaustive sampling in the object neighborhood. For this evaluation, this means a *sample reduction of* 90% and a *doubled up frame-rate* without decreased tracking performance.



(a) Tracking Rate (b) Frame-rate

Figure 3: Tracking rate and frame-rate at various numbers of particles N_P.

Due to the position estimate of the particle filter, additional label jitter is introduced into the tracking result. Using this result for training a classifier may lead to problems if the used learning algorithm is sensitive to label jitter. Thus, combining a particle filter with the original on-line boosting approach [8]

without any post-processing or refinement would result in even faster drifting of the tracker. This knowledge suggests that the higher runtime effort of more robust learning algorithms pays off with an easier and more stable integration of runtime-saving estimation methods.

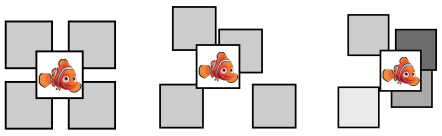
4.2 Improved Update Patch Selection

Since the particle filter provides a more sophisticated sampling of the search-space, also the used training sample selection method should make advantage of this. Therefore, different update schemes have been implemented and evaluated against each other. The update strategies can be divided into static, ranked, and random schemes. Static patch selection uses a fixed selection pattern, which is centered on the actual object location. Ranked patches use different criteria for sorting the patches $\mathbf{P}$ within the search area. The ranked patches are then selected randomly with higher probability for high-ranked patches[1]. As a ranking criterion we chose the confidence of the current on-line classifier. Random samples are selected randomly from the set of patches $\mathbf{P}$.

Figure 4 shows the average results of different implemented update rules for several test sequences. As can clearly be seen, the geometry-based patch selection yields better results than other selection methods. Since the circular selection of the update patches gains equal results as the computationally more expensive distance-rated method, circular selection is preferred. The chosen scheme randomly selects a fixed number of patches out of $\mathbf{P}$, where the distance to the object position lies in a fixed interval. This limits the selectable patches to a circular ring enclosing the actual object location. Figure 5 shows two differently updated trackers evaluated on a testing sequence.

	Simple	**Geometric**	Random	Distance	Confidence
Tracking Rate	0.96	**0.98**	0.87	0.98	0.92
Precision	0.96	**0.98**	0.88	0.99	0.93

(a) Comparison of different update methods



(b) Simple (c) Random (d) Rated

Figure 4: Update Methods.

(a) Initial position (b) Frame 350 (c) Frame 750

Figure 5: Circular updates (yellow rectangle) gain better result than simple updates (blue rectangle).

4.3 One-shot Prior Learning

A special characteristic of the proposed algorithm, the need of prior knowledge, can be interpreted both as an advantage and as a drawback. If the specific object is known a-priori, it is reasonable to

[1]Patches with lower index have higher selection probability: $p_{update} = P_{sorted}\left[\left\lfloor \frac{rand(|P|)^2}{|P|} \right\rfloor\right]$.

include all existent knowledge into the tracking process, whereas full knowledge of the object would make an adaptive classifier unnecessary.

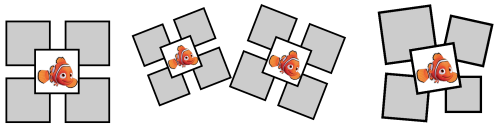
Taking only the first view of the object into account, as done in [9], would constrain the adaptive classifier very tight to this view of the object, since every update with a variation of the object would be weighted low. To overcome the problem of the missing training data for the prior classifier, virtual examples [7] are created during the initialization phase to simulate natural object behavior like rotation and scaling. Illumination changes have not been considered, since the used Haar-like features are inherently resistant to them.

Based on the simple original update scheme, different combinations of transformations have been implemented. For ease of implementation, only the input image was cloned and transformed; the training samples have then been chosen circularly from the created images. Schemes with fixed settings for placement and transformation have shown the best results, since they guarantee a fixed portion of transformed data within the training set.

For evaluation, the different initialization schemes have been used to train different classifiers that are kept fixed while tracking objects in various videos. The average results of this comparison can be seen in Figure 6. The combination of rotation and scale gains the best results, which can be interpreted quite naturally, since these are transformations an object undergoes naturally during movement. This initialization can beside initialization fo the prior classifier also be used for initial training of the semi-supervised classifier. Figure 7 shows two differently initialized trackers evaluated on a testing sequence.

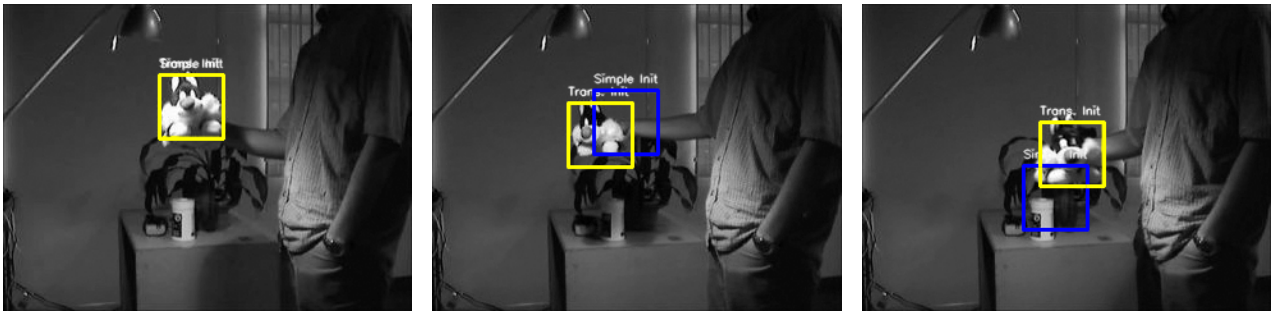
	Simple	Scale	Rotation	**Rot. & Scale**	Random
Tracking Rate	0.96	0.98	0.87	**0.98**	0.92
Precision	0.96	0.98	0.88	**0.99**	0.93

(a) Comparison of different one-shot training methods



(b) Simple (c) Rot. & Scale (d) Random

Figure 6: One-shot-Training.



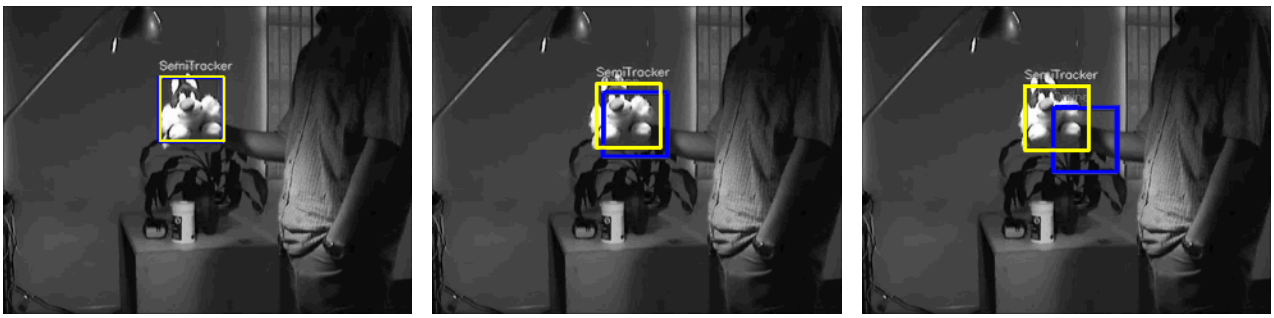
(a) Initial position (b) Frame 220 (c) Frame 500

Figure 7: Better initialization with virtual examples (yellow rectangle) lead to better alignment during tracking than without (blue rectangle).

4.4 The Influence of Label Noise

One of the major motivations for semi-supervised learning was the assumed higher resistance of the updating process to label noise. Thus, the influence of misaligned update samples to the semi-supervised classifier should be less than to the supervised classifier. Figure 8 shows the tracking result

of the semi-supervised on-line boosting algorithm [9] in comparison to the pure on-line boosting algorithm [8], being updated with manually misaligned samples (2 pixels to the right and 2 pixels down).



(a) Initial position (b) After 16k updates (c) After 40k updates

Figure 8: Being updated with misaligned samples, the supervised classifier (blue) drifts away while the semi-supervised classifier (yellow) is robust.

The tracking result (Fig. 8) clearly shows one of the main advantages of the semi-supervised update process in comparison to the unsupervised approach. Since the semi-supervised classifier uses the prior knowledge, which was extracted from the first frame, the misaligned samples are not learned with full weight $\lambda = 1$ as in the on-line case. Due to the used object representation based on simple Haar-like features, that are positioned within the object patch, such a misalignment can have a huge impact on the learned statistics.

4.5 Using Class Knowledge as a Prior

If tracking an object from an a priori known class, a general detector for this class could be used as prior knowledge. Since the prior in this case cannot distinguish between different instances of the tracked object class, the semi-supervised classifier is allowed to adapt to other co-occuring instances of this class. For instance, as can be seen in Figure 9, if we use a prior face detector (*i.e.*, taken from OpenCV[2]) the tracker may adapt to a second face appearing in the sequence that should not be tracked.

(a) Tracking a face, using a class-prior (b) Two faces within search space (c) Classifier switches to other face and starts adapting (d) Finally tracking the other face

Figure 9: Class-Prior Experiment.

[2]Open Source Computer Vision Library, http://opencv.willowgarage.com (02.04.2009)

(a) Initial Object (b) Occlusion, Tracker lost (c) Appearance change (d) Background change

Figure 10: Tracking the Dudek-Sequence (from http://www.cs.toronto.edu/∼dross/ivt/ (02.04.2009)).

5 Conclusion

The work of Grabner *et al.* [9] has shown the benefit of using semi-supervised on-line boosting for tracking. The fixed prior classifier serves as an adaptivity-limiting factor that reduces the drifting problem. In this work, we gave a detailed review on the algorithm and discussed its benefits for tracking. Moreover, we addressed open problems of the approach, such as the improved search-space sampling using particle filtering, an enhanced training sample selection scheme, and a method for creating an useful prior classifier out of the first frame by the creation of virtual samples. The thereby reduced runtime enables further improvements of the tracking system.

Since up to now Haar-like features are used for object representation, an extension of the search-space to rotation would preclude real-time processing. Therefore, other types of features should be explored [8, 12]. In this case, the particle filter should then be extended to estimate the whole state space of the tracked object. Alternative learning algorithms that are more robust to noise (*e.g.* on-line versions of [6, 15]) may be used.

References

[1] S. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 2002.

[2] S. Avidan. Ensemble tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

[3] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[4] R.T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2005.

[5] 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(1):119–139, 1997.

[6] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 2000.

[7] F. Girosi and N. Chan. Prior knowledge and the creation of virtual examples for rbf networks. In *IEEE Workshop on Neural Networks for Signal Processing*, 1995.

[8] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. British Machine Vision Conf.*, 2006.

[9] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proc. European Conf. on Computer Vision*, 2008.

[10] H. Grabner, J. Sochman, H. Bischof, and J.G. Matas. Training sequential on-line boosting classifier for visual tracking. In *Proc. Intern. Conf. on Pattern Recognition*, 2008.

[11] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[12] M. Grabner, C. Zach, and H. Bischof. Efficient tracking as linear program on weak binary classifiers. In *Proc. DAGM Symposium*, 2008.

[13] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Neural networks and natural intelligence*, 1998.

[14] C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[15] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

[16] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proc. IEEE Intern. Conf. on Computer Vision*, 2003.

[17] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. In *Advances in Neural Information Processing Systems*. 2005.

[18] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu. Semiboost: Boosting for semi-supervised learning. Technical report, Department of Comp. Science and Engineering, Michigan State University, 2007.

[19] L. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2004.

[20] N. Oza and S. Russell. Online bagging and boosting. In *Proc. Artificial Intelligence and Statistics*, 2001.

[21] A. Rahimi, B. Recht, and T. Darrell. Learning to transform time series with a few examples. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2007.

[22] R. Schapire. The boosting approach to machine learning: An overview. In *Proc. MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[23] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *Proc. IEEE Intern. Conf. on Computer Vision*, 2007.

[24] T. Woodley, B. Stenger, and R. Cipolla. Tracking using online feature selection and a local generative model. In *Proc. British Machine Vision Conf.*, 2007.

[25] X. Zhu. Semi-supervised learning literature survey. Technical report, Comp. Sciences, University of Wisconsin-Madison, 2005.