# On-line Boosting and Vision [*]

Helmut Grabner and Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology
{hgrabner, bischof}@icg.tu-graz.ac.at

## Abstract

*Boosting has become very popular in computer vision, showing impressive performance in detection and recognition tasks. Mainly off-line training methods have been used, which implies that all training data has to be a priori given; training and usage of the classifier are separate steps. Training the classifier on-line and incrementally as new data becomes available has several advantages and opens new areas of application for boosting in computer vision. In this paper we propose a novel on-line AdaBoost feature selection method. In conjunction with efficient feature extraction methods the method is real time capable. We demonstrate the multifariousness of the method on such diverse tasks as learning complex background models, visual tracking and object detection. All approaches benefit significantly by the on-line training.*

## 1. Introduction

Boosting has been successfully used in a wide variety of machine learning task and applied to computer vision problems. In this paper we focus on a specific boosting algorithm, discrete AdaBoost for classification. AdaBoost has been analyzed carefully (e.g. [27]) and tested empirically by many researchers. For instance, following the overview given in [26], boosting has been used for text recognition, text filtering, routing, "ranking" problems, learning problems in natural language processing, medical diagnostic and customer monitoring and segmentation. Various variants of Boosting have been developed (e.g. Real-Boost [9], LP-Boost [7]). Boosting is strongly related to support-vector machines (SVM) [23]. For SVM good results for vision problems have been reported (e.g. [2]), though on-line versions exist [28] their main drawback is the computational complexity.

In this paper we use boosting for feature selection. This has been initially introduced by Tieu and Viola [29] in the context of image retrieval. Feature selection and feature combination is of high interest to many researchers in machine learning [4]. Compared to other approaches, boosting has many advantages, therefore it is quite popular [24]. The seminal work of Viola and Jones [31] which applied boosting to robust and fast object detection tasks, paved the way of boosting in the area of computer vision, e.g. [18, 15, 30, 34].

All of the above mentioned approaches work off-line or in a "pseudo-on-line" batch processing mode (i.e. collecting a set of training examples and then run the off-line boosting algorithm). However, off-line boosting limits the usage for many applications. For example, tracking requires adaptive techniques for adjusting to possible variations of the target object over time. This can only be handled with adaptive methods which are able to incrementally update their representations. Thus there is an essential need for on-line algorithms that are able to learn continuously.

In this paper we introduce a novel on-line boosting based feature selection framework. Javed et al. [12] proposes on-line boosting in a co-training framework for object detection. In their work a classifier is first trained off-line on a generic scenario, which is later adapted and refined by on-line boosting. Our work is much more general which allows to perform on-line feature selection using boosting. Therefore we can tackle a large variety of real-time tasks. Second, we can manage very large feature pools at reduced computational costs. This gives the feasibility to apply our algorithm to many different real-time applications.

The remainder of the paper is organized as follows. In Section 2 we introduce the on-line algorithm, provide some analysis and draw relations to the off-line case. In Section 3 multiple applications (namely object detection, object tracking and background modeling), using this novel on-line method, are demonstrated. Finally, we present some

[*]The project results have been developed in the MISTRAL Project which is financed by the Austrian Research Promotion Agency (www.ffg.at). This work has been sponsored in by the Austrian Federal Ministry of Transport, Innovation and Technology under P-Nr. I2-2-26p VITUS2 and by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04, the EC funded NOE MUSCLE IST 507572.

conclusion and further work.

## 2. On-line feature selection

First, we briefly review the off-line boosting approach and how it is used for feature selection.We proceed by introducing on-line boosting. Finally, we present our novel on-line boosting for feature selection algorithm.

### 2.1. Off-line boosting

Boosting is a general method for improving the accuracy of any given learning algorithm. This is done by combining (a weighted voting) $N$ hypotheses which have been generated by repeating training with different subsets of training data. Boosting transforms a weak learning algorithm into a strong one. Let us first define some terms:

**Weak classifier:** A weak classifier has only to perform slightly better than random guessing (i.e., for a binary decision task, the error rate must be less than 50%). The hypothesis $h^{weak}$ generated by a weak classifier is obtained by applying a learning algorithm.

**Strong classifier:** Given a set of $N$ weak classifiers, a strong classifier is computed as linear combination of the weak classifiers. The value $conf(\cdot)$ (which is related to the margin) can be interpreted as a confidence measure.

$$h^{strong}(\mathbf{x}) = \text{sign}(conf(\mathbf{x})) \quad (1)$$

$$conf(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x}) \quad (2)$$

We focus on the discrete AdaBoost (adaptive Boosting) algorithm [9] introduced by Freund and Shapire, that reweights the training samples instead of re-sampling them. The basic algorithm works as follows: Given a training set $\mathcal{X} = \{\langle \mathbf{x_1}, y_1\rangle, ..., \langle \mathbf{x_L}, y_L\rangle \mid \mathbf{x_i} \in \mathbb{R}^m,\ y_i \in \{-1, +1\}\}$ with positive and negative labeled samples and an initial uniform distribution $p(\mathbf{x_i}) = \frac{1}{L}$ over the examples. A weak classifier $h^{weak}$ is trained based on $\mathcal{X}$ and $p(\mathbf{x})$. Based on the error $e_n$ the weak classifier $h_n^{weak}$ gets assigned a weight $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$. $p(\mathbf{x})$ is updated such that the probability increase for the samples that are misclassified. If the sample is classified correctly the corresponding weight is decreased. Therefore, the algorithm focuses on the difficult examples. The process is repeated, and at each boosting iteration a new weak hypothesis is added, until a certain stopping condition is met (e.g. a given number of weak classifiers are trained).

Freund and Schapire [9] proved strong bounds on the training and generalization error of AdaBoost. For the case of binary classification the training error drops exponentially fast with respect to the number of boosting rounds $N$ (i.e. number of weak classifiers). Schapire et al. [27] showed that boosting maximizes the margin and proved that larger margins for the training set are translated to superior upper bounds on the generalization error.

### 2.2. Off-line boosting for feature selection

Tieu and Viola [29] introduced boosting methods for feature selection. The idea is that each feature corresponds to a single weak classifier and boosting selects from the features. A pool of possible features $\mathcal{F}$ is given. Since this feature pool can be very large, for computational reasons the algorithm focuses on a subset $\mathcal{F}_{sub} = \{f_1, ..., f_k\} \subseteq \mathcal{F}$.

Training proceeds in a similar manner to standard boosting. In each iteration $n$ the algorithm selects one new feature and adds it (with the corresponding voting factor) to the ensemble. All features are evaluated and the best one is selected which forms the weak hypothesis $h_n^{weak}$, the weight $\alpha_n$ is set according to the error of $h_n^{weak}$. Finally, a strong classifier $h^{strong}$ is computed as weighted linear combination of the weak classifiers. As already stated the training error drops exponentially fast over the boosting iterations, which are now equivalent to the number of selected features.

### 2.3. On-line boosting

To obtain an on-line boosting algorithm (i.e. an algorithm that operates on a single example and discards it after updating), each of the boosting steps described above has to be done on-line. On-line updating the weak classifiers is not the problem because many on-line learning algorithms for generating a hypothesis can be used. Also the estimation of the corresponding voting-weights is straight forward, if we know the estimated error of the weak classifiers (which we do). The crucial step is the computation of the weight distribution for the samples, because we do not know a priori the difficulty of a sample (i.e., we do not know if we have seen the sample before).

We use ideas proposed by Oza et al. [21] and the experimental comparisons he did [20] (also some other approaches exist, e.g. for the *Arc-x4* algorithm [8]). The basic idea is that the importance (difficulty) of a sample can be estimated by propagating it through the set of weak classifiers. One can think of this, as modeling the information gain with respect to the first $n$ classifier and code it by the importance weight $\lambda$ for doing the update of the $n+1$-th weak classifier.

Oza has proved, if off-line and on-line boosting are given the same training set, then the weak classifiers (Naive Bayes classifiers) returned by on-line boosting converges statistically to the one obtained by off-line boosting as the number of iterations $N \rightarrow \infty$. Therefore, for repeated presentation of the training set on-line boosting and off-line boosting deliver the same result. For details see the PhD-Thesis of Oza [19].

The on-line algorithm requires that the number of weak classifiers is fixed at the beginning. Note the interchange of roles. In the off-line case all samples are used to update (and select) one weak classifier, whereas in the on-line case one sample is used to update all weak classifiers and the corresponding voting weight.

## 2.4. On-line boosting for feature selection

The approach of Oza is not directly applicable to feature selection. The essential novelty of our approach is, that we propose an on-line boosting algorithm for solving the feature selection task. For this purpose we need a further concept.

**Selector:** Given a set of $M$ weak classifier with hypothesis $\mathcal{H}^{weak} = \{h_1^{weak}, ..., h_M^{weak}\}$, a selector selects exactly one of those.

$$h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x}) \tag{3}$$

where $m$ is chosen according to a optimisation criterion. In fact we use the estimated error $e_i$ of each weak classifier $h_i^{weak} \in \mathcal{H}^{weak}$ such that $m = \arg\min_i e_i$.

Note, that the selector can interpreted as a classifier (he switches between the weak classifiers). Training a selector means that each weak classifier is trained (updated) and the best one (with the lowest estimated error) is selected. Similar to the off-line case, the weak classifiers $\mathcal{H}^{weak}$ correspond to features, i.e. the hypotheses generated by the weak classifier is based on the response of the feature. The selectors can therefore select from a subset of $M$ features $\mathcal{F}_{sub} = \{f_1, ..., f_M \mid f_i \in \mathcal{F}\}$ of the global feature pool.

In summary: *The main idea is to apply on-line boosting not directly to the weak classifiers but to the selectors.*

The overall principle is depicted in Fig. 1 and in Algorithm 2.1. In particular, the new on-line AdaBoost training for feature selection works as follows: First, a fixed set of $N$ selectors $h_1^{sel}, .., h_N^{sel}$ is initialized randomly, each with its own feature pool $\mathcal{F}_n$. When a new training sample $\langle \mathbf{x}, y \rangle$ arrives the selectors are updated. This update is done with respect to the importance weight $\lambda$ of the current sample[1]. For updating the weak classifiers, any on-line learning algorithm can be used, but we employ a standard EM technique to estimate the probability distributions of positive and negative samples and generate a hypothesis (see Section 3 for more details). The weak classifier with the smallest error is selected by the selector

$$\arg\min_m(e_{n,m}),\ e_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}} \tag{4}$$

[1]Either $\lambda$ is used as a learning rate in the learning algorithm or by $k$-times repeated updating $k \sim Poisson(\lambda)$ as proposed by Oza.

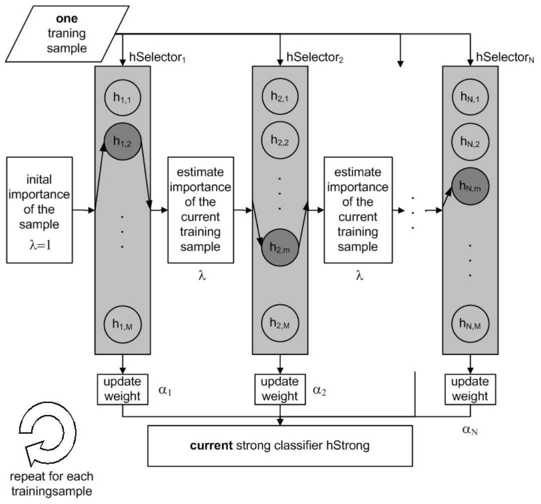


Figure 1. Novel on-line boosting for feature selection.

$e_{n,m}$ is the error of the $m$-th weak classifier $h_{n,m}^{weak}$ in the in the $n$-th selector, estimated from the weights of correctly $\lambda_{n,m}^{corr}$ and wrongly $\lambda_{n,m}^{wrong}$ classified examples seen so far. Finally, the corresponding voting weight $\alpha_n$ and the importance weight $\lambda$ of the sample are updated and passed to the next selector $h_{n+1}^{sel}$.

In order to increase the diversity of the classifier pool $F_n$ for the selector $h_n^{sel}$ and to adapt to changes in the environment the worst weak classifier is replaced by one randomly chosen from the feature pool $\mathcal{F}$.

This procedure is repeated for all selectors. The number of selectors is constant similar to the number of weak classifiers in Oza's on-line algorithm. A strong classifier is obtained by linear combination of selectors.

$$h^{strong}(\mathbf{x}) = \text{sign}\Big(\sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(\mathbf{x})\Big) \tag{5}$$

In contrast to the off-line version a classifier is available at any time.

## 2.5. Discussion

We point out the differences and relations between the off-line and the on-line algorithm and take a look at the advantages and disadvantages. The main difference is that in the on-line case only the information from one training example is used and in the off-line case the training set is fixed.

From the algorithm's point of view, in the off-line case at each boosting iteration a new weak classifier is created and thus a feature from the feature pool $\mathcal{F}$ is selected and added it to the ensemble. A drawback for the on-line algorithm, is that the discriminative complexity of the classifier is limited, because the number of weak classifiers is fixed (since also the number of selectors is fixed). However, the features selected within a selector and also the voting-weight

**Algorithm 2.1** On-line AdaBoost for feature selection

```
Require: training example $\langle \mathbf{x}, y \rangle$, $y \in \{-1, +1\}$
Require: strong classifier $h^{strong}$ (initialized randomly)
Require: weights $\lambda_{n,m}^{corr}$, $\lambda_{n,m}^{wrong}$ (initialized with 1)

  initialize the importance weight $\lambda = 1$
  // for all selectors
  for $n = 1, 2, .., N$ do

      // update the selector $h_n^{sel}$
      for $m = 1, 2, .., M$ do

          // update each weak classifier
          $h_{n,m}^{weak} = \text{update}(h_{n,m}^{weak}, \langle \mathbf{x}, y \rangle, \lambda)$

          // estimate errors
          if $h_{n,m}^{weak}(\mathbf{x}) = y$ then
              $\lambda_{n,m}^{corr} = \lambda_{n,m}^{corr} + \lambda$
          else
              $\lambda_{n,m}^{wrong} = \lambda_{n,m}^{wrong} + \lambda$
          end if
          $e_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}}$
      end for

      // choose weak classifier with the lowest error
      $m^+ = \arg\min_m(e_{n,m})$
      $e_n = e_{n,m^+}$; $h_n^{sel} = h_{n,m^+}^{weak}$
      if $e_n = 0$ or $e_n > \frac{1}{2}$ then
          exit
      end if

      // calculate voting weight
      $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$

      // update importance weight
      if $h_n^{sel}(\mathbf{x}) = y$ then
          $\lambda = \lambda \cdot \frac{1}{2 \cdot (1-e_n)}$
      else
          $\lambda = \lambda \cdot \frac{1}{2 \cdot e_n}$
      end if

      // replace worst weak classifier with a new one
      $m^- = \arg\max_m(e_{n,m})$
      $\lambda_{n,m^-}^{corr} = 1$; $\lambda_{n,m^-}^{wrong} = 1$;
      get new $h_{n,m^-}^{weak}$

  end for
```

can change over time. In addition, the feature pool $\mathcal{F}_n \subseteq \mathcal{F}$ of each selector is adapted by replacing the worst feature (weak classifier). If the process is running for a long time, a lot of features are processed and evaluated but still only a small number of features is sufficient for updating the selector. Since in the on-line case learning continues, the model will continuously improve by exploring more features and training data.

The aspect of real-time is important for on-line learning. Our proposed algorithm is easy to implement and runs very fast. Both speed and memory are $O(M \cdot N)$ assuming that all $N$ selectors include the same number of $M$ weak classifiers. The main computational effort is spend for updating the weak classifiers, which depends on the learning algorithm and the time to calculate the feature value. The time consumed by our framework is negligible. In order to decrease computation time we use a method similar to Wu et al. [33]. Assuming all feature pools in the selectors are the same $\mathcal{F}_1 = \mathcal{F}_2 = ... = \mathcal{F}_M$, then we can update all corresponding weak classifiers only once and so the selectors choose only the best feature according to $\lambda$. This speeds up the process considerably while only slightly decreasing the performance.

## 3. Applications and Experiments

In this paper we use the standard Haar-like features from Viola and Jones [31] and in addition orientation histograms (with 16 bins) similar to [13, 6] and a simple version (4-th neighborhood) of local binary patterns (LBP) [17] as features. Note that we use integral images and integral histograms [22] as efficient data structures, which allow a very fast calculation of all these feature types. All our experiments shown bellow run in real time on a standard PC (Intel Pentium 1.6 GHz with 512 MB RAM).

On-line learning for obtaining a weak classifier $h_j^{weak}$ for a feature $j$, where $f_j(\mathbf{x})$ evaluates this feature on the image $\mathbf{x}$ can be implemented straight forward for these feature types. We build a model by estimate the probability $P(1|f_j(\mathbf{x}))$ via a Gauss distribution with mean $\mu^+$ and standard deviation $\sigma^+$ for positive labeled samples and $P(-1|f_j(\mathbf{x}))$ by $\mathcal{N}(\mu^-, \sigma^-)$ for negative labeled samples. For this purpose we incrementally estimate the mean and variance by a Kalman filtering approach [32]. We build a simple state space model for estimation the (constant) mean and achieve $\mu_t = \mu_{t-1} + v_t$ and $\sigma_t^2 = \sigma_{t-1}^2 + v_t$ for the variance. $v_t \sim \mathcal{N}(0, R)$ is a random noise processes with variance $R$. In our experiments we set $R = 0.01$ and furthermore the initial state $P_0 = 1000$, $\mu_0 = 0$ and $\sigma_0^2 = 0$. The following update equations for the adaptive estimation can be derived:

$$
\begin{aligned}
K_t &= P_{t-1}/(P_{t-1} + R) && (6)\\
\mu_t &= K_t \cdot f_j(\mathbf{x}) + (1 - K_t) \cdot \mu_{t-1} && (7)\\
\sigma_t^2 &= K_t \cdot (f_j(\mathbf{x}) - \mu_t)^2 + (1 - K_t) \cdot \sigma_{t-1}^2 && (8)\\
P_t &= (1 - K_t) \cdot P_{t-1} && (9)
\end{aligned}
$$

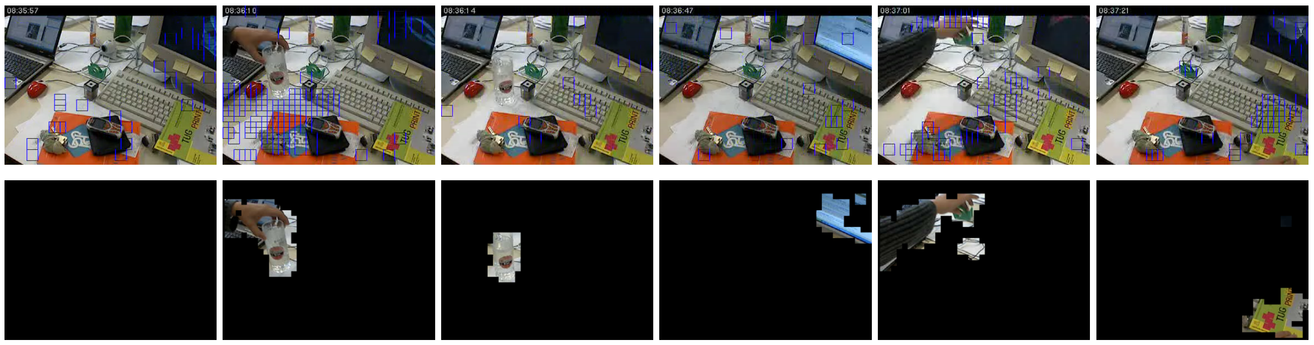


Figure 2. Evaluation of the background model. Each thin blue rectangle in the input image (first row) corresponds to classifiers which are updated in order to learn the "allowed" changes. The regions which can not be modeled by the learned classifiers are related to the foreground (second row).

As hypotheses for the classical Haar Wavelets we use either a simple threshold

$$h_j^{weak}(\mathbf{x}) = p_j \cdot \text{sign}(f_j(\mathbf{x}) - \theta_j) \tag{10}$$

$$\theta_j = |\mu^+ + \mu^-|/2, \; p_j = \text{sign}(\mu^+ - \mu^-) \tag{11}$$

or a Bayesian decision criterion, based on the estimated Gaussian probability density function $g(x|\mu,\sigma)$

$$h_j^{weak}(\mathbf{x}) = \text{sign}(P(1|f_j(\mathbf{x})) - P(-1|f_j(\mathbf{x}))) \tag{12}$$

$$\approx \text{sign}(g(f_j(\mathbf{x}|\mu^+,\sigma^+) - g(f_j(\mathbf{x})|\mu^-,\sigma^-)) \tag{13}$$

For the histogram based features (orientation histograms and LBP), we use nearest neighbor learning with a distance function $D$. The cluster centers for positive $\mathbf{p_j}$ and negative $\mathbf{n_j}$ samples are learned (estimated) by applying the Kalman filter technique to each bin.

$$h_j^{weak}(\mathbf{x}) = \text{sign}(D(f_j(\mathbf{x}), \mathbf{p_j}) - D(f_j(\mathbf{x}), \mathbf{n_j})) \tag{14}$$

The whole feature pool $\mathcal{F}$ contains an enormous amount of possible features because of the highly over complete representation (each feature prototype can appear at different position and scale). In our experiments the local feature pool $\mathcal{F}_n$ used by the $n$-th selector is very small, in particular we use $|\mathcal{F}_n| = 250$ and $N = 50$ selectors. In the following we show how the novel on-line boosting algorithm can be used for such diverse tasks as background modeling, tracking and object detection. In all three cases the on-line learning capability is required.

## 3.1. Background model

A basic task in surveillance applications is background subtraction. One needs a robust and flexible background model. Based on the idea of a block based background model [10] (which contains also a good overview of related work), we propose a *classifier based* background model. The basic idea is to partition the image in small (overlapping) blocks, each block contains a classifier, which classifies the region as foreground or background.

Where we define as background everything that is statistical predictable in the image. Therefore everything, that can not be predicted is foreground. This definition allows us to describe dynamic (multi modal) background (e.g. flashing light, moving leaves in the wind, flag waves, etc.). A robust background model has to adapt to dynamic backgrounds and must be sensitive to corresponding foreground objects. Therefore on-line algorithms are required.

First, in the initial period a separate classifier is build for each patch by considering all frames as positive examples. Later on, the algorithm analyzes the image and does positive updates according to a given policy. In our experiment we use following very simple policy: We update a classifier if the normalized confidence is positive but less than one half.

$$0 < \frac{conf(\mathbf{x})}{\sum_{n=1}^{N} \alpha_n} \leq \frac{1}{2} \tag{15}$$

Since we do not have negative examples we treat the problem as a one-class classification problem, therefore only positive samples for updating are used. The classifier models only the background (including dynamic changes), everything else belongs to a foreground object. We can calculate the negative distribution for each feature directly without learning. We model the gray value of each pixel as a uniformly distributed with mean 128 and variance $\frac{256^2}{12}$ (for an 8 bit image). Applying standard statistic operations the parameters of the negative distribution $\mu^-$ and $\sigma^-$ of Haar features can be computed easily. In the case of an orientation histogram feature, the negative cluster $\mathbf{n_j}$ consists of equally distributed orientations. The characteristic negative cluster for a 16 bin LBP-feature is given by $\mathbf{n_j} = \frac{1}{50} \cdot [6,4,4,1,4,1,1,4,4,1,1,4,1,4,4,6]$ for the binary patterns $[0000_2, 0001_2, 00010_2 ..., 1111_2]$[2].

An experimental example is shown in Figure 2 for a cluttered desk scenario. The first row shows the input sequence

[2]These numbers are obtained by a lengthly calculation assuming equal probability of all patches.

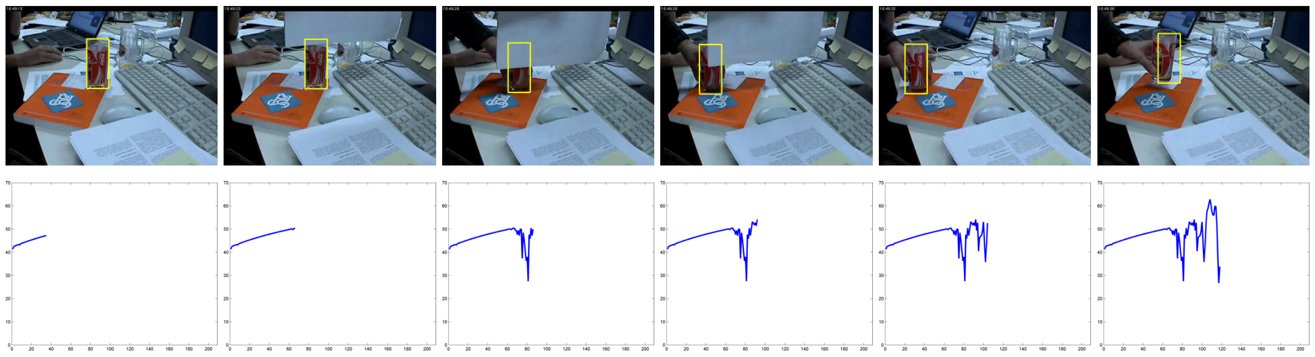


Figure 3. Experimental results when applying our on-line feature selection method to tracking. The initial marked glass is robustly tracked (occlusions and changes in appearance) over the whole image sequence (first row). The second row shows the maxima of the confidence map over time.

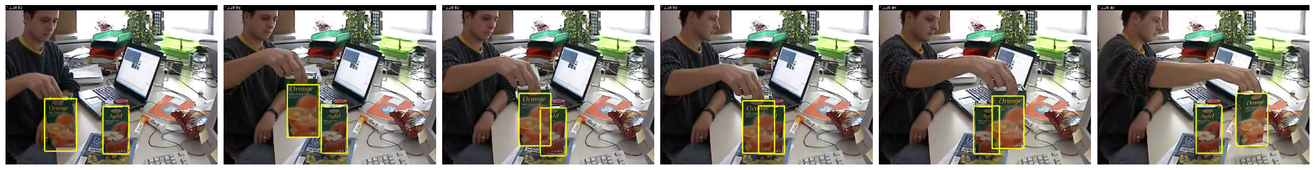
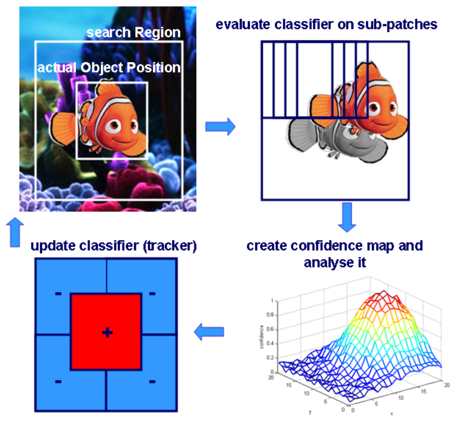


Figure 4. Robust tracking of two similar objects which are severely occluded.

overlapped with blue rectangles indicating classifiers which are currently updated. Those classifiers that give a negative response, and by definition this are foreground objects, are depicted in the second row. Added, removed or shifted objects in the scene are detected very well. Note, that during learning of this sequence the screen saver was active, therefore this dynamic background has been correctly modeled as background by the classifiers. But, when we weak up the computer (4-th column) the change is detected.

On a standard quarter PAL ($384 \times 288$) resolution a framerate of about 15-20 fps is achieved. Each sub-classifier contains 30 selectors (each can choose from 250 weak classifier) and analyzes a $20 \times 20$ image patch with 50% overlapp.


search Region
actual Object Position
evaluate classifier on sub-patches
create confidence map and analyse it
update classifier (tracker)
+
-
Figure 5. Principle of tracking with a classifier

## 3.2. Tracking

This work was inspired by Avidan [3]. The main idea is to formulate the tracking task as a classification problem and to continuously update the current classifier which represents the object to optimally discriminate it from the current background [5].

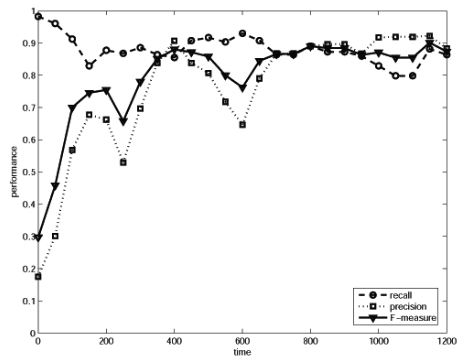
The principle is depicted in Figure 5. We assume that we have already detected the object in the gray-scale image and therefore have an initial tracking region. We start, building an initial classifier using the marked region as positive samples and patches in the local neighborhood as negative samples. At time $t + 1$ we evaluate the current classifier in a region of interest around the previous detection (in fact we could also use a motion model to predict the new position). For each classified patch we receive a confidence value which is entered in a confidence map. This confidence map is analyzed and the tracking window is shifted to the best possible position[3]. We update the classifier and continue the process. This update has several advantages, first we learn to optimally discriminate the object from the background, and second the object can change its appearance which will be learned by the boosting approach. In addition depending on the update parameters the algorithm is robust to occlusions. The on-line update capability is the big advantage over support vector or relevance vector tracking methods [2].

[3]At the moment we simply shift it to the maximum, but also a more powerful method, like a mean shift approach, can be used. But even our simple strategy shows pretty good results that are adequate for demonstrating the on-line boosting method.

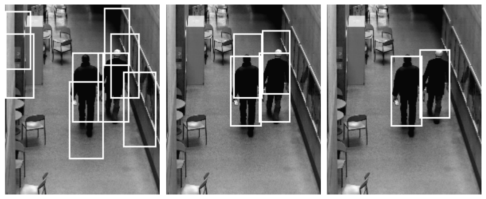
Figure 3 demonstrates the tracker, the first row shows the tracking of the object over time. The second row depicts maximal confidence over time. At the beginning nothing changes and the confidence increases, because on-line boosting finds better and better features for describing the object. Later, when we occlude the object, the confidence decreases, however the object is correctly tracked as it moves.

Again, this is only possible since we can perform a very fast on-line update of the strong classifier modeling the object. Tracking different kinds of objects is feasible because we learn a specific model based tracker. Furthermore also the tracking of multiple objects is possible. For example, a face detector can initialize a tracker for each face. The tracker will learn the best representation for each instance (a more specific model). See Figure 4 for an illustration with boxes which are similar and occlude each other, but nevertheless the tracker never gets confused.

The performance depends on the size of the search region. We simply search on a region twice as large as the last known position. We achieve an update rate of more than 20 fps. The strong classifier contains 50 selectors (each can choose from 250 weak classifiers).



(a)



(b) (c) (d)

Figure 6. Performance of the detector trained with conservative learning framework (a) and particular results at beginning (b), after 300 learning updates (c) to the final version after about 1200 updates (d). Taken from [25]

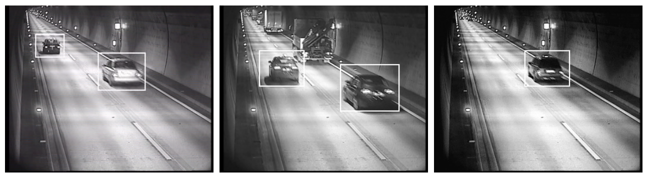


Figure 7. Results obtained by the final car classifier using conservative learning.

## 3.3. Object Detection

Based on the classical work from Viola and Jones [31] many authors perform object detection by off-line boosting. The trained classifier (detector) is scanned over the whole gray-scale image at multiple locations and scales. This has to be done for each object.

In this application there is no need for on-line learning. But on-line learning becomes very interesting (and moreover essential) when one intends to continuously learn a model. This is the case for active learning and image retrieval applications [11, 1] where human input is used to update the classifier (e.g. a support vector machine).

Another example where the on-line algorithm is useful is learning detections by either co-training [14], or the methods proposed by Nair and Clark [16] or Roth et al. [25]. There the basic idea is that other algorithms provide labels that can be used for updating the classifier. By a combination of different methods one can train detectors without hand labeling. Figure 6 depicts various stages of a person detector learning process. A motion model and a robust PCA algorithm is used to train the on-line AdaBoost classifier. One can see that the person detector is significantly improving as the training continues. Figure 7 demonstrates the same algorithm applied to cars.

## 4. Conclusion

In this paper we have introduced a novel on-line algorithm for feature selection based on boosting. As we have demonstrated on-line learning and on-line feature selection is essential for a wide variety of computer vision problems. Three fundamental vision applications, namely background modeling, tracking and active learning for object detections, have been presented. Due to the efficient computation of the features based on integral images, all presented applications run in real-time. In addition, we have already achieved impressive results using very simply strategies, which can be further improved by using more sophisticated ones. The experiments show that various inevitable variations that occur in natural scenes (e.g., changes in illumination, changes in viewpoint, reflectance) can only be adequately handled by on-line learning methods adjusting to these changes.

The paper presents only a subset of possible applications where the proposed on-line feature selection algorithm can be used. Moreover we have started to combine these tasks more closely, i.e. updating the detector while tracking and training a recognizer module, all requiring on-line capabilities.

## References

[1] Y. Abramson and Y. Freund. SEmi-automatic VIsual LEarning (SEVILLE): Tutorial on active learning for visual object recognition. In *Proc. CVPR*, 2005.

[2] S. Avidan. Support vector tracking. *PAMI*, 26:1064–1072, 2004.

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

[4] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

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

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, volume 1, pages 886–893, 2005.

[7] A. Demiriz, K. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[8] A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53(1-2):71–109, 2003.

[9] Y. Freund and R. 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.

[10] M. Heikkil, M. Pietikinen, and J. Heikkil. A texture-based method for detecting moving objects. In *Proc. 15th BMVC*, pages 187–196, 2004.

[11] S. Hoi and M. Lyu. A semi-supervised active learning framework for image retrieval. In *Proc. CVPR*, volume 2, pages 302–309, 2005.

[12] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proc. CVPR*, pages 695–700, 2005.

[13] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *Proc. CVPR*, pages 53–60, 2004.

[14] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proc. 9th ICCV*, pages 626–633, 2003.

[15] Y. Li and W. Ito. Shape parameter optimization for adaboosted active shape model. In *Proc. 10th ICCV*, 2005.

[16] V. Nair and J. Clark. An unsupervised, online learning framework for moving object detection. In *Proc. CVPR*, 2004.

[17] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.

[18] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proc. 8th ECCV*, volume 2, pages 71–84, 2004.

[19] N. Oza. *Online Ensemble Learning*. PhD thesis, University of California, Berkeley, 2001.

[20] N. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[21] N. Oza and S. Russell. Online bagging and boosting. In *Proc. Artificial Intelligence and Statistics*, pages 105–112, 2001.

[22] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proc. CVPR*, volume 1, pages 829–836, 2005.

[23] G. Rätsch, B. Schökopf, S. Mika, and K. Müller. Svm and boosting: One class. 2000.

[24] D. Redpath and K. Lebart. Observations on boosting feature selection. In *Proc. Multiple Classifier Systems*, pages 32–41, 2005.

[25] P. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proc. Workshop on VS-PETS*, 2005.

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

[27] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proc. International Conference on Machine Learning*, pages 322–330, 1997.

[28] D. Tax and P. Laskov. Online SVM learning: From classification to data description and back. In *Proc. Neural Network and Signal Processing*, pages 499–508, 2003.

[29] K. Tieu and P. Viola. Boosting image retrieval. In *Proc. CVPR*, pages 228–235, 2000.

[30] A. Torralba, K. Murphy, and W. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *Proc. CVPR*, volume 2, pages 762–769, 2005.

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

[32] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, UNC-CH Computer Science Technical Report 95041, 1995.

[33] J. Wu, J. Rehg, and M. Mullin. Learning a rare event detection cascade by direct feature selection. In *Proc. NIPS*, 2003.

[34] P. Yang, S. Shan, W. Gao, S. Li, and D.Zhang. Face recognition using Ada-boosted Gabor features. In *Proc. Conference on Automatic Face and Gesture Recognition*, pages 356–361, 2004.