Active Sampling via Tracking *

Peter M. Roth and Horst Bischof
Graz University of Technology
Institute for Computer Graphics and Vision
{pmroth, bischof}@icg.tugraz.at

Abstract

To learn an object detector labeled training data is required. Since unlabeled training data is often given as an image sequence we propose a tracking-based approach to minimize the manual effort when learning an object detector. The main idea is to apply a tracker within an active on-line learning framework for selecting and labeling unlabeled samples. For that purpose the current classifier is evaluated on a test image and the obtained detection result is verified by the tracker. In this way the most valuable samples can be estimated and used for updating the classifier. Thus, the number of needed samples can be reduced and an incrementally better detector is obtained. To enable efficient learning (i.e., to have real-time performance) and to assure robust tracking results, we apply on-line boosting for both, learning and tracking. If the tracker can be initialized automatically no user interaction is needed and we have an autonomous learning/labeling system. In the experiments the approach is evaluated in detail for learning a face detector. In addition, to show the generality, also results for completely different objects are presented.

1. Introduction

A lot of research has been focused on efficient training of detectors but only little attention has been paid to efficiently labeling and acquiring suitable training data. Thus, training data, i.e., positive and negative samples are usually obtained by hand labeling a large number of images, which is a time consuming and tedious task.

Negative examples (i.e., examples of images not containing the object) are usually obtained by a bootstrap approach [21]. Starting with a few negative examples a classifier is trained. The obtained classifier is applied to images, that do not even contain the object-of-interest. Those subimages, where a (wrong) detection occurs are added to the set of negative examples and the classifier is retrained. This process can be repeated several times. Therefore, obtaining negative examples is usually not much of a problem.

Automatically obtaining a set of positive examples is a more difficult task. Existing approaches to minimize the labeling effort (e.g., [10, 15]) are based on two stages. In the first stage, an initial classifier is trained using a smaller number of examples. In the second stage, the classifier is applied on a training sequence and the detected patches are added to the set of positive examples. Levin et al. [10] start with a small set of hand labeled data and generate additional labeled examples by applying co-training of two classifiers. To completely avoid hand labeling Nair and Clark [15] propose to use motion detection to obtain the initial training set. New examples are acquired by applying a detector obtained by on-line learning (Winnow).

As a disadvantage for these approaches an initial classifier has to be trained and the classifiers are biased by the examples used to train the initial model. Thus, a great number of potential new examples, even different views of the same object, would not be added because they do not fit to the learned model! Having a sequence of images this can be avoided by using a tracker. Once initialized a robust tracker is able to follow the object-of-interest through a sequence of images. If tracking works on a longer sequence different views of an object can be obtained and used for learning.

Thus, especially in field of face recognition, various tracking-based methods were proposed to to gather training samples (e.g., [9, 19]). Lee et al. [9] proposed to couple face recognition and face tracking to improve the recognition rate. Both modules share the same appearance model, a collection of linear subspaces, which approximates a nonlinear appearance manifold (e.g., [14]). Tracking is applied in both, the training and recognition stage. In the training stage, the training samples are obtained by cropping the results obtained by a variant of eigentracking [2] on a training sequence. In the recognition stage, the final result is obtained by using a Bayesian framework, that considers both, the results of the detection in the current frame and the (tracked) results from the previous frames.

*This work was supported by the FFG project AUTOVISTA (813395) under the FIT-IT program and the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04.

Similarly, Sivic et al. [19] apply tracking to obtain training samples. They first run a frontal face detector [13], that is trained by boosting orientation-based features. To achieve a low false positive rate a conservative detection threshold is used. Hence, a lot of faces are not detected and the false negative rate is increased. The thus obtained detections are then used to initialize an affine covariant region tracker [20]. To compute the face representation from the tracked patches, first, localized features such as left and right eyes, tip of the nose, and center of the mouth are searched. Then, the object representation is built from five overlapping SIFT descriptors [12] at the detected features. As a disadvantage, to learn the model for the feature position and appearance a great amount (i.e., 5000 images) of hand-labeled face images is needed!

In contrast, Hewitt and Belongie [7] proposed a method for learning a face representation, where a tracker serves for verification. The tracker locates the face correctly whereas the initial classifier may fail. To finally compute a classifier large-scale and low-scale features are used. But as a disadvantage the user needs to manually select the face in the first input frame to build the initial face model.

But these approaches have several limitations. First, a manual initialization [9] or a pre-trained classifier [9, 19] is needed to initialize the learning process. In addition, since a simple tracker is prone to errors and the selected patches must be inspected manually before they can be used for learning [9] . Thus, still some human effort is necessary. Second, even though the tracking information provides the labels on the fly, which would allow on-line learning, the models are trained off-line. Finally, the usage of very complex object models limits the approaches to a particular object class (i.e., faces).

To overcome these problems in this paper we introduce an active learning framework, that is based on discriminative on-line learning. Thus, no off-line training is required and an existing classifier can directly be updated. The main idea is that a tracker is applied for selecting the most valuable positive and negative samples. Thus, in contrast to a passive random sampling strategy a faster convergence is assured. In particular we apply on-line boosting for both, for learning and tracking, which allows to learn a detector in real-time (e.g., processing images from a life camera). But since the approach is quite general other methods may be applied as well. To have an autonomous learning system the tracker is initialized automatically. Thus, for learning an object representation no user interaction is needed.

The outline of the paper is as follows: First, in Section 2 we discuss the applied on-line learning method, the used tracker, and possible autonomous initializations for tracking. Next, in Section 3 we introduce the tracking-based active sampling strategy. Experimental evaluations are given in Section 4. Finally, we conclude the paper in Section 5.

2. Preliminaries

2.1. On-line Learning

For learning in this paper we apply On-line Boosting for Feature Selection [5]. The main idea of boosting for feature selection, in general, is that each feature f_j corresponds to a single weak classifier h_j and that boosting selects an informative subset of N features, where a 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%). In fact, various different feature types may be applied but similar to the seminal work of Viola and Jones [23] in this work we use Haar-like features, which can be calculated efficiently using integral data-structures.

Thus, in the off-line case boosting for feature selection can be summarized as follows: given a training set of positive and negative samples $\mathcal{S} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_L, y_L)\}$, where $\mathbf{x}_l \in \mathbb{R}^m$ is a sample and $y_l \in \{-1, +1\}$ is the corresponding label, a set of possible features $\mathcal{F} = \{f_1, ..., f_M\}$, a learning algorithm $\mathcal{L}$, and a weight distribution D, that is initialized uniformly by $D(l) = \frac{1}{L}$. In each iteration n, $n = 1, ..., N$, all features f_j, $j = 1, \ldots, M$ are evaluated on all samples $(\mathbf{x}_l, y_l)$, $l = 1, \ldots, L$ and hypotheses are generated by applying the learning algorithm $\mathcal{L}$ with respect to the weight distribution D over the training samples. The best hypothesis is selected and forms the weak classifier h_n and the weight distribution D is updated according to the error of the selected weak classifier. The process is repeated until N features are selected (i.e., N weak classifiers are trained). Finally, a strong classifier H is computed as a weighted linear combination of all weak classifiers h_n.

Contrary, during on-line learning each training sample is provided only once to the learner. Thus, all steps described above have to be on-line and the weak classifiers have to be updated whenever a new training sample is available. On-line updating the weak classifiers is not a problem since various on-line learning methods exist, that may be used for generating hypotheses. The same applies for the voting weights α_n, that can easily be computed if the errors of the weak classifiers are known. The crucial step is the computation of the weight distribution since the difficulty of a sample is not known a priori. Thus, the basic idea is to estimate the importance λ of a sample by propagating it through the set of weak classifiers [16]. In fact, λ is increased proportional to the error e of the weak classifier if the sample is misclassified and decreased otherwise.

Thus, the work-flow for on-line boosting for feature selections selection can be described as follows: a fixed number of N selectors $s_1, .., s_N$ is initialized with random features. A selector s_n can be considered a set of M weak classifiers $\{h_1, \ldots, h_M\}$, 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

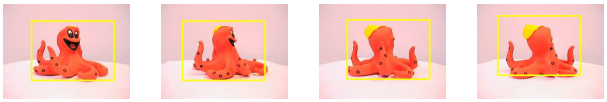
pool. The selectors are updated whenever a new training sample $\langle \mathbf{x}, y \rangle$ is available and the selector $s_n(\mathbf{x})$ selects the best weak hypothesis according to the estimated training error from the importance weights of the correctly and incorrectly classified samples seen so far. Finally, the weight α_n of the n-th selector s_n is updated, the importance λ_n is passed to the next selector s_{n+1}, and a strong classifier is computed by a linear combination of N selectors:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n s_n(\mathbf{x})\right). \quad (1)$$

Thus, contrary to the off-line version, an on-line classifier is available at any time of the training process.

2.2. Tracking

In addition to speed (real-time!) the most important quality criterion for a tracker is the stability. The stability can be considered a measure how exactly a tracker determines the object's position over time. Since for the learning framework proposed in this paper the tracked patches are used to select the patches for updating the current classifier the patches must be cut very accurately. Thus, a "very stable" tracker is needed, that is able to follow an object without drifting. The importance of a stable tracker is demonstrated in Figure 1. We tracked a toy animal, that was put onto a turn table. Since only the appearance of the object is changing but not the position this is a very simple tracking task. As can be seen from Figure 1(a), where the object was tracked using a simple color tracker, the object is tracked but there are small drifts. Thus, the object is not optimally captured. In contrast, for the results shown in Figure 1(b) we used a more stable tracker and we get highly accurate results.



(a)

(b)

Figure 1. Importance of a stable tracker for learning: (a) tracking results obtained by a simple color tracker – the bounding box is drifting and the tracked patches can not be used for learning; (b) tracking results obtained by Tracking via On-line Boosting – highly accurate patches are obtained, that can be used to learn an object model.

In fact, the results shown in Figure 1(b) were obtained by Tracking via On-line Boosting [5], where the main idea is to formulate tracking as a binary classification problem. For that purpose by using on-line boosting, that was discussed in detail in Section 2.1, a classifier is estimated, that discriminates between the object and the local background. In contrast to related (boosting-based) tracking approaches (e.g., [1, 3]) applying an on-line learning method allows to immediately model changes in the object's appearance. Moreover, varying the number of selectors provides a steerable environment in terms of speed and accuracy.

The tracking step is based on simple template tracking [6]. Assuming that the object was detected at time t at time $t+1$ a search region is defined around the detection in previous frame. By evaluating the current classifier on each subpatch in the region of interest a confidence value is obtained for each patch. The thus obtained confidence map is analyzed and the target window is shifted to the best position, i.e., the window where the confidence value is maximized. Alternatively, a mean shift approach [4] may be applied. Finally, the classifier is updated and the process is continued.

To initialize the tracker, a selected image region is assumed to be a positive sample. At the same time negative examples are extracted by taking regions of the same size as the target window from the surrounding background. Using these samples several iterations of the on-line boosting algorithm are carried out. Thus, the classifier adapts to the specific target object and at the same time it is discriminative against its surrounding background.

2.3. Initialization of the Tracker

The simplest method is to manually initialize the tracker. But to have an automatic learning system this step should be automatic as well. The main idea is to find a suitable segmentation, that separates the object-of-interest from the background. In the following we give a not exhaustive summary of approaches, that we successfully applied to automatically initialize a tracker:

Change Detection: To initialize a tracker via change detection first a background model **B** (see, e.g., [18]) is estimated Then, given the current image **I** the changes caused by foreground objects can be detected by background subtraction and pixel-wise thresholding:

$$\mathbf{FG}(m,n) = |\mathbf{B}(m,n) - \mathbf{I}(m,n)| > \theta\,. \quad (2)$$

From the thus obtained binary regions bounding boxes are estimated, that can be used to initialize the tracker.

Color Model: Since most image sources (cameras, video streams, etc.) provide color images it might useful to use this color information; at least to initialize the tracker. In particular, we can apply the ideas of skin-color modeling to initialize a tracker. Thus, a skin-color model can be estimated as follows: First, the variance caused by the intensity

is removed. In fact, this is achieved by normalizing the data or by transforming the original pixel values into a different color space (e.g., *rg*-color-space or *HSV*-color-space). Second, based on this more suitable representation a color histogram is computed, which is used to estimate an initial mixture model by *K-means clustering*. Finally, a Gaussian mixture model is estimated, which can efficiently be done by applying the iterative EM-algorithm. To initialize a tracker the probability map of an image is thresholded and the thus obtained binary mask is used to define the object-of-interest.

Detector: Similar to the approach of Sivic et al. [19] a weak detector may be used to initialize the process for learning a stronger detector. For instance, an object detector, that captures only partially the appearance of an object (e.g., a frontal person detector), can be used to initialize a tracker. Then through tracking new views of the object are collected and finally a detector can be trained, that models the whole appearance.

3. Active Learning Framework

When learning a classifier the samples are usually drawn randomly from a fixed set. The set represents the underlying distributions of positive and negative samples. Hence, a great number of samples is needed. Such a sampling strategy, which is often referred to as *passive learning* [11], would result in a slow convergence.

To overcome these problems an adaptive learning algorithm taking advantage of the ideas if *active learning* (e.g., [11,22,24]) can be applied. In general, an active learner can be considered a quintuple $(C, Q, S, \mathbf{L}, \mathbf{U})$ [11], where C is a classifier, Q is a query function, S is a supervisor (teacher), and $\mathbf{L}$ and $\mathbf{U}$ are a set of labeled and unlabeled data, respectively. First, an initial classifier C_0 is trained from the labeled set $\mathbf{L}$. Given a classifier C_{t-1}, then the query function Q selects the most informative unlabeled samples from $\mathbf{U}$ and the supervisor S is requested to label them. Using the thus labeled samples the current classifier is re-trained obtaining a new classifier C_t. This procedure is summarized in Algorithm 1.

Algorithm 1 Active Learning

Input: unlabeled samples $\mathbf{U}$, classifier C_{t-1}
Output: classifier C_t

1: **while** teacher can label samples u_j **do**
2: Apply C_{t-1} to all samples u_j
3: Let Q find the m most informative samples u_q
4: Let teacher S assign labels y_q to samples u_q
5: Re-train classifier: C_t
6: **end while**

When considering an adaptive system we can start from a small set of labeled data $\mathbf{L}$. But usually the unlabeled data $\mathbf{U}$ is not available in advance. Assuming we have already trained a classifier C_{t-1} the first crucial point at time t is to define a set of unlabeled data $\mathbf{U}_t$ and a suitable query function Q.

Thus, the first crucial question is "What are the most valuable samples?". It has been shown [17] that it is more effective to sample the current estimate of the decision boundary than the unknown true boundary. In fact, the most valuable samples are exactly those, that were misclassified by the current classifier, which is illustrated in Figure 2. The red and green points indicate the samples, that were wrongly classified by the current classifier. Obviously, these points are much better samples than randomly selected points from the class! Hence, the algorithm is focused on the hard samples and the current decision boundary (dashed gray line) can be moved such that those samples are correctly classified (solid gray line). For a detailed theoretical discussion see [17]. In this way the number of required training samples can considerably be reduced! Considering the task of object detection the misclassified samples are the detected false positives and the missed true positives.

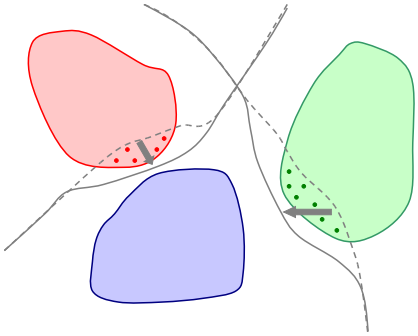
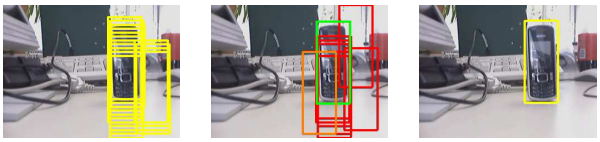


Figure 2. Sampling at the current decision boundary is more efficient than random sampling: updating using only a small number of wrongly classified samples (red and green points) allows to move the current decision boundary (dashed gray line) in the right direction (solid gray line).

The second crucial point is the definition of the supervisor S. In fact, new unlabeled data has to be robustly included into the already built model. More formally, at time t given a classifier C_{t-1} and an unlabeled example $\mathbf{x}_t \in \mathbb{R}^m$. Then, a reliable supervisor is needed, that robustly estimates a label $y_t \in \{+1, -1\}$ for $\mathbf{x}_t$.

Assuming that we have a robust tracker, that is able to follow an object over a longer period of time without small drifts. The key idea in this paper is that, when learning a discriminative classifier C, a tracker T undertakes both, the task of the query function Q and of the teacher S. This is illustrated in Figure 3.



(a) (b) (c)

Figure 3. Active learning - suitable updates for learning a detector: (a) classifier evaluated at time $t-1$, (b) positive (green bounding box) and negative (red bounding boxes) updates selected at time t, and (c) classifier evaluated at time $t+1$.

For that purpose, at time t the classifier C_{t-1} is applied on the current image $\mathbf{I}_t$ (see Figure 3(a)). The thus obtained detection result is verified by the tracking result T_t (based on size and position of the corresponding bounding boxes), that robustly represents the object-of interest. Based on this verification the valuable samples (see Figure 3(b)), i.e., the reported false positives (red bounding boxes) and the (missing) true positives (green bounding boxes), are identified (tracker T is used as query function Q). In addition, such the selected samples are labeled (tracker T is used as teacher S). These samples are fed back into the discriminative classifier as positive and negative examples, respectively, and we finally get a better classifier C_t (see Figure 3(c)). Exploiting the huge amount of video data (i.e., we have video stream) this process can be iterated to produce a stable and robust classifier. The update strategy is summarized more formally in Algorithm 2.

Algorithm 2 Active Sampling Strategy

Input: discriminative classifier D_{t-1}, tracking result T_{t-1}, unknown image $\mathbf{I}_t$
Output: classifier D_t

1: Evaluate D_{t-1} on $\mathbf{I}_t$ obtaining J detections x_j
2: **for** $j = 1, \ldots, J$ **do**
  if $T_{t-1} \approx x_j$ **then**
    $update\,(D_{t-1}, x_j, +)$
  else if $T_{t-1} \not\approx x_j$ **then**
    $update\,(D_{t-1}, x_j, -)$
  end if
3: **end for**

In praxis a two stage approach is applied. In the first stage, to generate an initial classifier, tracked patches are used as positives samples whereas the negative samples are extracted randomly from the background, where the tracker has not detected an object. In the second stage, this classifier is re-trained by using the Active Sampling strategy described in Algorithm 2. The whole learning approach is summarized more formally in Algorithm 3.

Algorithm 3 Active Sampling via Tracking

1: Initialize Tracker T_0
2: Collect pos./neg. training samples
3: Train initial classifier D_0
4: **while** t **do**
5: Evaluate Tracker: T_t
6: update D_t (Algorithm 2)
7: **end while**

To actually learn the object representation we apply and On-line Boosting for Feature Selection (Section 2.1), but any other on-line learning method may be applied. Moreover, we apply Tracking via On-line Boosting (see Section 2.2), that is based on the same representation. By using a robust configuration (i.e., by using 50 selectors) we can assure that the tracking results have the required accuracy. To initialize the learning process the object is either selected by hand once or, if possible, automatically by any of methods discussed in Section 2.3.

4. Experimental Results

In the following demonstrate the presented approach mainly based on a face detection task. Thus, the learning process was started using a weak pre-trained face detector. Alternatively, a probabilistic skin-color model can be applied for the same purpose.

First, we show the importance of a good tracker. Therefore, we created several challenging sequences each consisting of 250 frames and trained different face classifiers as described above. The only difference between these classifiers is that in the active learning framework trackers of varying accuracy were applied to select new training samples. In fact, Tracking via On-line Boosting provides a steerable environment, i.e., we can vary the number of applied selectors. Illustrative detection results for two special cases (using 10 and 50 applied selectors) are shown in Figure 4.

For both cases the detector mainly captures the faces. But it clearly can be seen that the detection results in Figure 4(a), that were obtained for the detector trained using the samples selected by the more accurate tracker, are more accurate. Depending on the ground-truth and the used evaluation criterion the detections shown in Figure 4(b) could even be counted as false positive.

More formally, this is summarized in Table 1 and Table 2, where we analyzed the detection characteristics. For that purpose, we varied the level of significance. In fact, to enable a fair comparison we applied the overlap criterion. From Table 1 it can be seen that for the detector trained using the reliable samples whether the number of true positives is decreased nor the number of false positives is in-

(a)

(b)

Figure 4. Quality of detection depends on the accuracy of the tracker: (a) tracker using 50 applied selectors and (b) tracker using 10 applied selectors.

creased if we increase the required overlap from originally 40% to 60%. In contrast, if we perform the same experiment for the detector trained by using the less reliable data the recall and the precision are dramatically reduced. This can be explained by the small drifts of the tracker, which return patches, that do not describe the faces very well. Hence, we get a less significant detector.

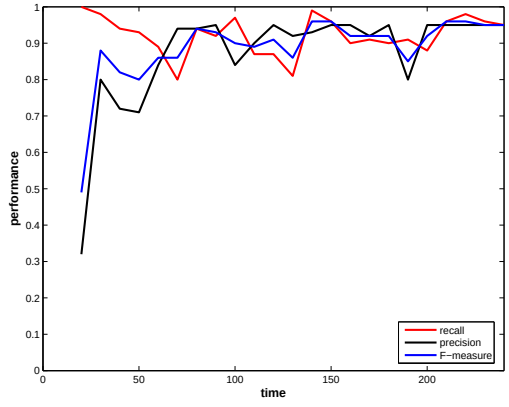
overlap	40%	50%	60%
recall	0.92	0.92	0.92
prec.	0.95	0.95	0.94
F-m.	0.95	0.94	0.93

Table 1. Detection characteristics for face detector trained from "good" samples if required overlap is varied.

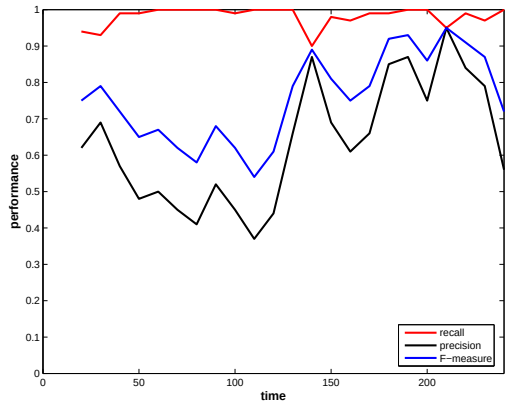
overlap	40%	50%	60%
recall	0.91	0.77	0.37
prec.	0.86	0.74	0.36
F-m.	0.89	0.75	0.36

Table 2. Detection characteristics for face detector trained from "bad" samples if required overlap is varied.

In the following we show that by using the proposed update strategy the number of required samples can be reduced and that a better classifier can be estimated. Thus, we compare the performance of face detectors, that were trained on-line by using different sampling strategies. The first detector (Active Sampling) was trained as described in Section 3. The second detector was trained using patches, that were obtained by the tracker whereas the negative samples were randomly selected from the background (Random Sampling). For both methods during learning after a predefined number of processed frames classifiers were saved, which were evaluated on an independent test sequence; the obtained performance curves are shown in Figure 5.



(a)



(b)

Figure 5. (a) Active Sampling vs. (b) Random Sampling.

From Figure 5(a) it can be seen that for Active Sampling less than 100 training frames are required to obtain a precision, a recall, and hence an F-measure of more than 0.90. In contrast, Figure 5(b) shows that Random Sampling requires a longer settling phase. Even though a trend of increasing accuracy can be observed after approximative 150 training frames we the precision is insufficient and the precision curve still shows a stochastic trend. But, in fact, compared to Active Sampling we have a higher recall.

The same can be seen from Table 3, where we compare the final face detectors obtained by Active Sampling and Random Sampling. In addition, a state-of-the-art face detector, i.e., the approach of Viola & Jones [23] is included in this comparison. To evaluate this detector we used the pre-trained classifier included in OpenCV[1], that is based on the work of Krupa et al. [8].

	Active Samp.	Rand. Samp.	V & J
recall	0.96	0.97	0.64
prec.	0.95	0.79	0.76
F-m.	0.95	0.87	0.69

Table 3. Face detection: Active Sampling vs. Random Sampling vs. Viola & Jones.

Similar to the learning curves shown in Figure 5 it can be seen that Active Sampling and Random Sampling have a comparable recall whereas the precision for Active Sampling is much better. Moreover, the recall as well as the precision for the Viola & Jones approach is much smaller compared to the two other methods. This is not a surprising result since this detector was trained for frontal faces only. But as can be seen in Figure 4 the evaluation sequences also contain semi-profile faces and persons looking down or to the ceiling, that can not be captured by the detector.

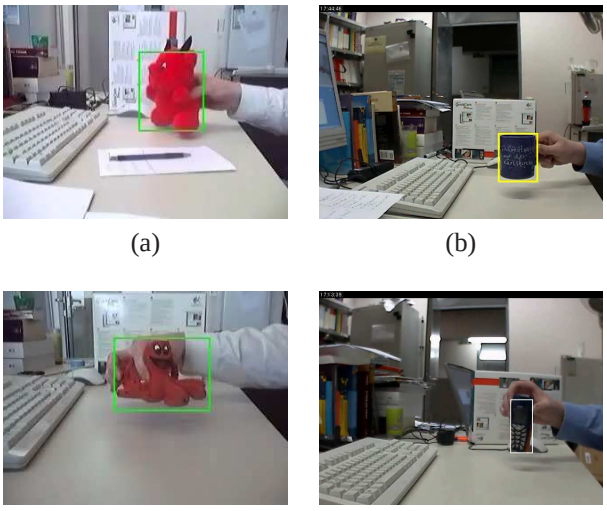
Finally, to demonstrate that the proposed approach is not limited to face detection we applied the method to learn a detector for different hand held objects (i.e., a soft-toy (devil), a bathing-toy (octopus), a coffee cup, and a cell phone). To start the learning process for these experiments change detection was applied to automatically initialize the tracker. Alternatively, for objects having a predominant color such as the devil (see Figure 6(a)) or the cup (see Figure 6(b)) also a probabilistic color model may be applied for the same purpose. The thus obtained detection results are summarized in Table 4.

	devil	cup	octopus	cell phone
recall	0.95	0.98	0.99	0.94
prec.	0.98	1.00	1.00	1.00
F-m.	0.96	0.99	0.99	0.97

Table 4. Detection results obtained for hand held objects.

[1] http://sourceforge.net/projects/opencvlibrary/ (March 20, 2008)

Even only a small number of training frames (i.e., approximative 250) were used for learning competitive detection results were obtained. In addition, representative detection results are shown in Figure 6. It can be seen that there is a high variance in size, shape, and appearance. Hence, the presented approach is quite general and is not restricted to specific visual properties.



(a) (b) (c) (d)

Figure 6. Illustrative detection results for different hand held objects: (a) devil, (b) cup, (c) octopus, and (d) cell phone.

5. Conclusion

We have presented a tracking-based active learning approach for learning a discriminative object representation. Having an image sequence, the main idea is to track the object-of-interest and to use the tracking results to select and label new training samples. For that purpose the current discriminative classifier is evaluated on an input image and the detection results are verified by the tracker. In this way only valuable samples are used for updating, which in fact, reduces the training time to finally get a powerful classifier. Since a stable tracker and an efficient on-line learning method are required we apply on-line boosting for feature selection for both, for learning the object representation and for tracking. That is, the presented approach can be applied for real-time learning a detector (e.g., processing images from a video camera). In contrast to similar approaches, the presented method is not limited to a particular objects and the human effort is minimized. In fact, the experimental results show that competitive detectors for faces and hand held objects can be estimated. Moreover, if the tracker can be initialized automatically by an independent cue such as change detection or a weak detector no user interaction is needed; otherwise the human effort is reduced to manually select the object-of-interest once in the beginning.

References

[1] S. Avidan. Ensemble tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 494–501, 2005.

[2] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. European Conf. on Computer Vision*, pages 329–342, 1996.

[3] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005.

[4] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume II, pages 1197–1203, 1999.

[5] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 260–267, 2006.

[6] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[7] R. Hewitt and S. Belongie. Active learning in face recognition: Using tracking to build a face model. In *Proc. IEEE Workshop on Vision for Human-Computer Interaction*, 2006.

[8] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Proc. IEEE Intern. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 157–164, 2003.

[9] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Visual tracking and recognition using probabilistic appearance manifolds. *Computer Vision and Image Understanding*, 99(3):303–331, 2005.

[10] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume I, pages 626–633, 2003.

[11] M. Li and I. K. Sethi. Confidence-based active learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(8):1251–1261, 2006.

[12] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intern. Journal of Computer Vision*, 60(2):91–110, 2004.

[13] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust detectors. In *Proc. European Conf. on Computer Vision*, volume I, pages 69–82, 2004.

[14] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *Intern. Journal of Computer Vision*, 14(1):5–24, 1995.

[15] V. Nair and J. J. Clark. An unsupervised, online learning framework for moving object detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 317–324, 2004.

[16] N. C. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proc. ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining*, 2001.

[17] J.-H. Park and Y.-K. Choi. On-line learning for active pattern recognition. *IEEE Signal Processing Letters*, 3(11):301–303, 1996.

[18] M. Piccardi. Background subtraction techniques: a review. In *Proc. IEEE Intern. Conf. on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.

[19] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: Video shot retrieval for face sets. In *Intern. Conf. on Image and Video Retrieval*, pages 226–236, 2005.

[20] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *Proc. European Conf. on Computer Vision*, volume I, pages 85–98, 2004.

[21] K.-K. Sung and T. Poggio. Example-based learning for view-based face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[22] M. Turtinen and M. Pietikänien. Labeling of textured data with co-training and active learning. In *Proc. Workshop on Texture Analysis and Synthesis*, pages 137–142, 2005.

[23] P. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 511–518, 2001.

[24] R. Yan, J. Yang, and A. Hauptmann. Automatically labeling video data using multi-class active learning. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume I, pages 516–523, 2003.