Conservative learning for learning object detectors [⋆]

Peter M. Roth and Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{pmroth,bischof}@icg.tu-graz.ac.at

Abstract. We will introduce a new effective framework for learning an object detector. The main idea is to minimize the manual effort when learning a classifier and to combine the power of a discriminative classifier with the robustness of a generative model. Starting with motion detection an initial set of positive examples is obtained by analyzing the geometry (aspect ratio) of the motion blobs. Using these samples a discriminative classifier is trained using an on-line version of AdaBoost. In fact, applying this classifier nearly all objects are detected but there is a great number of false positives. Thus, we apply a generative classifier to verify the obtained detections and to decide if a detected patch represents the object of interest or not. As we have a huge amount of data (video stream) we can be very conservative and use only patches for (positive or negative) updates if we are very confident about our decision. Applying this update rules an incrementally better classifier is obtained without any user interaction. Moreover, an already trained classifier can be re-trained on-line and can therefore easily be adapted to a completely different scene. We demonstrate the framework on different scenarios including pedestrian and car detection.

1 Introduction

In the recent years, the demand for automatic methods analyzing large amounts of visual data has been increasing. Starting with face detection [32,38] there has been a considerable interest in visual object detection, e.g., pedestrians [39], cars [1], bikes [23], etc. All these trends have been encouraging research in the area of automatic detection, recognition, categorization, and interpretation of objects, scenes and events.

These visual systems have to encompass a certain level of knowledge about the world they are observing and analyzing. The most convenient way of knowledge acquisition is its accumulation through learning. Thus, the core of most systems is usually a classifier, e.g., AdaBoost [8], Winnow [18], neural network [32] or support vector machine [37] that is obtained by learning.

These approaches have achieved considerable success in the above mentioned applications but a requirement of all these methods is a training set which in some cases needs to be quite large. The problem of obtaining enough training data increases even

[⋆] This work has been supported by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04, the S9104-N04EU FP6-507752 NoE MUSCLE IST project, and by the Austrian Federal Ministry of Transport, Innovation and Technology under P-Nr. I2-2-26p VITUS2.

further because the methods are view based, i.e., if the view-point of the camera changes the classifier needs to encompass this variability (e.g., car from the side and car from the back). Training data is usually obtained by hand labeling a large number of images which is a time consuming and tedious task. For illustration consider the pictures shown in Fig.1. To get a considerable training set the task would be to mark all persons that occur within this scene. Clearly, this is not practicable for applications requiring a large number of different view-points (e.g., video surveillance by large camera networks). Thus, acquiring training samples and as a consequence learning should be as automatic as possible (especially if the amount of data required for learning is huge).

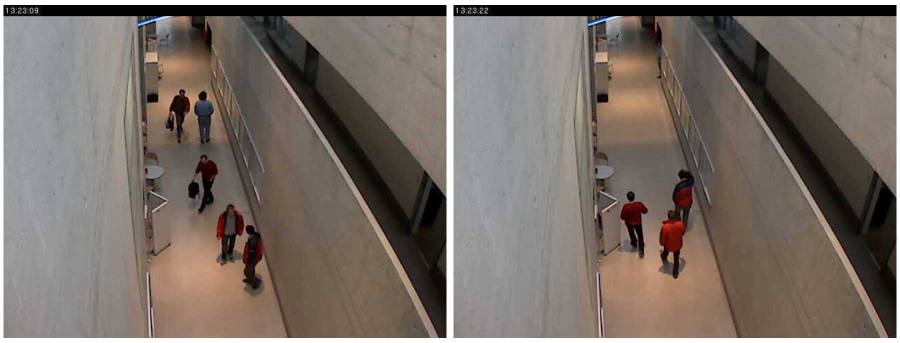


Fig. 1. Hand labeling: labeling all occurring persons as positive examples is a time consuming task if a great amount of training data is required.

Negative examples (i.e., examples of images not containing the object) are usually obtained by a bootstrap approach [35]. One starts with a few negative examples and trains a classifier. The obtained classifier is applied to images not containing the object. Those sub-images 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.

Obtaining reliable positive examples is, however, a more difficult problem, since discriminant classifiers are very sensitive to false training data. In addition, if the visual system's environment is constantly changing, the system has to keep adapting to these changes. It has, therefore, to keep continuously learning and updating its knowledge. When implementing continuous learning mechanisms, two main issues have to be addressed. First, the representation, which is used for modeling the observed world, has to allow updating with newly acquired information. This update step should be efficient and should not require access to the previously observed data while still preserving the previously acquired knowledge. And secondly, a crucial issue is the quality of updating, which highly depends on the correctness of the interpretation of the current visual input. In this context, several learning strategies can be used, ranging from a completely supervised learning approach (when the correct interpretation of the current visual input is given by a tutor) to a completely unsupervised approach (when the visual system has to interpret the current input without any additional assistance). Obviously, the latter approach is preferable, especially when the amount of data to be processed is large.

Having these two issues in mind, one has to carefully select the type of representations of the objects (or subjects, or scenes) that can be used. Discriminative representations are compact, task dependent, efficient, and effective, but usually not very robust. On the other hand, reconstructive representations are usually less efficient and less effective, but more general and robust. Thus, it would be desirable to combine these two representations to achieve best of both worlds which would lead to efficient and effective, while still general and robust continuous learning techniques.

In fact, all of these requirements are met by the *on-line conservative learning framework* [29,31]. A preliminary version of this approach [30] was based on batch methods. Thus, it was not suitable for on-line learning which we show is a beneficial extension for unsupervised learning. To avoid hand labeling of input data, we want that the visual system labels data automatically. During the learning process a sufficient knowledge is required for reliably evaluating the visual input. Thus, the process of labeling should strongly be intertwined with the process of continuous learning, which could provide enough redundant information to determine statistically consistent data. Only the sufficiently consistent data would then be used to build the representations, enabling robust learning (and updating of the representations) under non-ideal real-world conditions. We refer to this approach as conservative learning. Also importantly, such a conservative approach assures, that non-relevant (corrupted, inexact, false) data is not included into the model and that the model is not degraded.

The proposed framework is depicted in Fig. 2. The basic idea is to use a huge amount of unlabeled data that is readily available for most detection task (i.e., just mount a video camera and observe the scene) to avoid hand labeling of training data for object detection tasks.

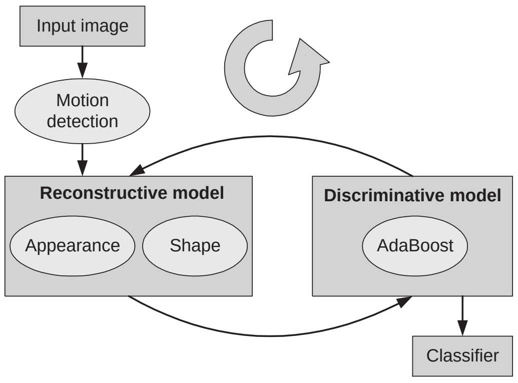


Fig. 2. The proposed on-line learning framework.

We use two types of models, a reconstructive one which assures robustness and serves for verification, and a discriminative one, which actually performs the detection.

To get the whole process started we use a simple motion detector to detect potential objects of interest. In fact, we miss a considerable amount of objects (which can be compensated by just using longer sequences) and we will also get miss-detections (which will be reduced in the subsequent steps). The output from the motion detector can be used to robustly build a first initial reconstructive representation. To further increase the robustness we are combining multiple cues – we use one representation on shape and the other on appearance. In particular, we use the robust incremental PCA [34] at this stage such that most of the miss-detections (background, false detections, over-segmentations, etc.) are not incorporated in the reconstructive model. This is very crucial as the discriminative classifier needs to be trained with "clean" images to produce good classification results. The discriminative model, the incremental Ada-Boost [9], is then used to detect new objects in new images. The output of the discriminative classifier is then verified by the reconstructive model, and detected false positives are fed back into the discriminative classifier as negative examples and true positives as positive examples to further improve the discriminative model. In fact, it has been shown in the active learning community [26] that it is more effective to sample the current estimate of the decision boundary than the unknown true boundary. This is exactly achieved by our combination of reconstructive and discriminative classifiers. Exploiting the huge amount of video data this process can be iterated to produce a stable and robust classifier. Since all methods (PCA, AdaBoost, Background) operate in an incremental manner, every image can be discarded immediately after it is captured and used for updating the model.

The outlined approach is similar to the recent work of Nair and Clark [21] and Levin et al. [15] who also proposed methods for automatic labeling new training data. Nair and Clark propose to use motion detection for obtaining the initial training set and then Winnow as a final classifier. Their approach does not include reconstructive classifiers, nor does it iterate the process to obtain more accurate results. In that sense our framework is more general. Levin et al. use the so called co-training framework to start with a small training set and to increase it by using a co-training of two classifiers operating on different features. We show that using a combination of reconstructive and discriminative classifiers helps to increase the performance of the discriminative one.

2 On-line Conservative Learning

2.1 Motion Detection

To get the process started we use motion detection (or more generally change detection). Having a stationary camera a common approach to detect foreground objects is to pixel-wise threshold the difference image between the currently processed image and a background image. Let $\mathbf{B}_t$ be the current estimated background image, $\mathbf{I}_t$ the current input image and θ a threshold, then a pixel is classified as foreground if

$$|\mathbf{B}_t(x, y) - \mathbf{I}_t(x, y)| > \theta. \tag{1}$$

For realistic applications different environmental conditions (e.g., changing lightening conditions or foreground models moving to background and vica versa) have to be handled. Therefore, several adaptive methods for estimating a background model $\mathbf{B}_t$ have been proposed that update the existing model with respect to the current input frame $\mathbf{I}_t$ (e.g., running average [12], temporal median filter [19], approximated median filter [20]). A more detailed discussion of these different background models can be found in [4, 10, 27].

To minimize the computational costs and to have an adaptive on-line method we apply the approximated median method [20] to estimate the background model. The approximated median filter computes an approximation of the temporal median by incrementing the current estimate by one if the input pixel value is larger than the estimate and by decreasing it by one if smaller:

$$\mathbf{B}_{t+1}(x,y) = \begin{cases} \mathbf{B}_t(x,y) + 1 & \mathbf{B}_t(x,y) < \mathbf{I}_t(x,y) \\ \mathbf{B}_t(x,y) - 1 & \mathbf{B}_t(x,y) > \mathbf{I}_t(x,y)\ . \end{cases} \tag{2}$$

This estimate (only one reference image has to be stored!) eventually converges to the real median. To avoid that non-moving foreground objects are accumulated into the background model and to preserve a robust median estimation spatial and temporal weights may be assigned for the updates.

The obtained motion blobs can be labeled as persons if the aspect ratio of their bounding box is within the pre-specified limits. Examples of accepted patches are shown in Fig. 3 where the corresponding blob regions are marked by a green bounding box. As can be seen from Fig. 3(a) motion blobs that do not fit to the given constraints are not included into the set of positive trainings samples.

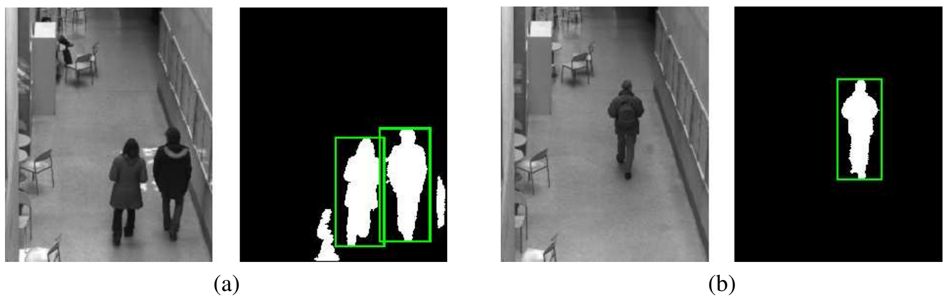


Fig. 3. Motion blobs obtained from background subtraction. Blobs that fulfill the constrains are accepted are positive samples (green bounding boxes).

2.2 Reconstructive Model

As reconstructive model we use a PCA-based subspace representation. This low-dimensional representation captures the essential reconstructive characteristics by exploiting the redundancy in the visual data. As such, it enables "hallucinations" and comparison of the visual input with the stored model [13]. In this way the inconsistent data can be rejected and the discriminative model can be trained from clear data only.

During the learning process, however, a sufficient knowledge is required for a reliable evaluation of the visual input that is still to be acquired. Nevertheless, by considering the reconstruction error, the robust learning procedure can discard inconsistencies in the input data and train the model from consistent data only [6,33]. Furthermore, this can be done also in an incremental way. A bunch of methods for incremental building of eigenspaces have been proposed [3, 11, 16]. Some of them also address the problem of robust incremental learning [17,34]. We use a simplified version of [34] and by checking the consistency of the input images (patches) we keep continuously accepting or rejecting potential patches as positive or negative training examples for the discriminative learner (and updating the reconstructive model). In the following this algorithm is summarized.

For batch PCA all training images are processed simultaneously. A fixed set of input images $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ is given, where $\mathbf{x}_i \in \mathbb{R}^m$ is an individual image represented as a vector. It is assumed that $\mathbf{X}$ is mean normalized. Let $\mathbf{Q} \in \mathbb{R}^{m \times m}$ be the covariance matrix of $\mathbf{X}$, then the subspace $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{m \times n}$ can be computed by solving the eigenproblem for $\mathbf{Q}$ or more efficiently by solving SVD of $\mathbf{X}$.

In contrast, for incremental learning, the training images are given sequentially. Assuming that an eigenspace was already built from n images, at step $n+1$ the current eigenspace can be updated in the following way [34]: First, the new image $\mathbf{x}$ is projected in the current eigenspace $\mathbf{U}^{(n)}$ and the image is reconstructed: $\tilde{\mathbf{x}}$. The residual vector $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$ is orthogonal to the current basis $\mathbf{U}^{(n)}$. Thus, a new basis $\mathbf{U}'$ is obtained by enlarging $\mathbf{U}^{(n)}$ with $\mathbf{r}$. $\mathbf{U}'$ represents the current images as well as the new sample. Next, batch PCA is performed on the corresponding low-dimensional space $\mathbf{A}'$ and the eigenvectors $\mathbf{U}''$, the eigenvalues $\boldsymbol{\lambda}''$ and the mean $\boldsymbol{\mu}''$ are obtained. To update the subspace the coefficients are projected in the new basis $\mathbf{A}^{(n+1)} = \mathbf{U}''^T \left(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}\right)$ and the subspace is rotated: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$. Finally, the mean $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ and the eigenvalues $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ are updated. In each step the dimension of the subspace is increased by one. To preserve the dimension of the subspace the least significant principal vector may be discarded [11]. The method is summarized more detailed in Algorithm 1.

To obtain an initial model, the batch method may be applied to a smaller set of training images. Alternatively, to have a fully incremental algorithm, the eigenspace may be initialized using the first training image $\mathbf{x}$: $\boldsymbol{\mu}^{(1)} = \mathbf{x}$, $\mathbf{U}^{(1)} = \mathbf{0}$ and $\mathbf{A}^{(1)} = \mathbf{0}$.

This method can easily be extended in a robust manner, i.e., corrupted input images may be used for incrementally updating the current model. To achieve this, outliers in the current image are detected and replaced by more confident values: First, an image is projected to the current eigenspace using the robust approach [13] and the image is reconstructed. Second, outliers are detected by pixel-wise thresholding (based on the expected reconstruction error) the original image and its robust reconstruction. Finally, the outlying pixel values are replaced by the robustly reconstructed values.

To further increase the robustness of the reconstructive model, we build two subspace representations in parallel: appearance-based and shape-based representation. The former is created from the cropped and resized appearance patches, which are detected by the motion detector. Since the output of this detector is also a binary segmentation mask, this mask is used to calculate the shape images based on the Euclidean

Algorithm 1 Incremental PCA

Input: mean vector $\boldsymbol{\mu}^{(n)}$, eigenvectors $\mathbf{U}^{(n)}$, coefficients $\mathbf{A}^{(n)}$, input image $\mathbf{x}$
Output: mean vector $\boldsymbol{\mu}^{(n+1)}$, eigenvectors $\mathbf{U}^{(n+1)}$, coefficients $\mathbf{A}^{(n+1)}$, eigenvalues $\boldsymbol{\lambda}^{(n+1)}$

```
 1: Project image x to current eigenspace:
        $\mathbf{a} = \mathbf{U}^{(n)T}\left(\mathbf{x} - \boldsymbol{\mu}^{(n)}\right)$
 2: Reconstruct image:
        $\tilde{\mathbf{x}} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$
 3: Compute residual vector:
        $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$
 4: Append r as new basis vector to U:
        $\mathbf{U}' = \begin{pmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{||\mathbf{r}||} \end{pmatrix}$
 5: Determine the coefficients in the new basis:
        $\mathbf{A}' = \begin{pmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & ||\mathbf{r}|| \end{pmatrix}$
 6: Perform PCA on $\mathbf{A}'$ and obtain the mean $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{U}''$ and the eigenvalues $\boldsymbol{\lambda}''$.
 7: Project coefficients to new basis:
        $\mathbf{A}^{(n+1)} = \mathbf{U}''^{T}\left(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}\right)$
 8: Rotate subspace:
        $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$
 9: Update mean:
        $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$
10: Update eigenvalues:
        $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$
```

distance transform [2]. The first five eigenvectors of the thus obtained appearance-based and shape-based model are shown in Figs. 4(a,b). In addition, an example of such binary masks and the corresponding shapes obtained by distance transform is shown in Fig. 5.

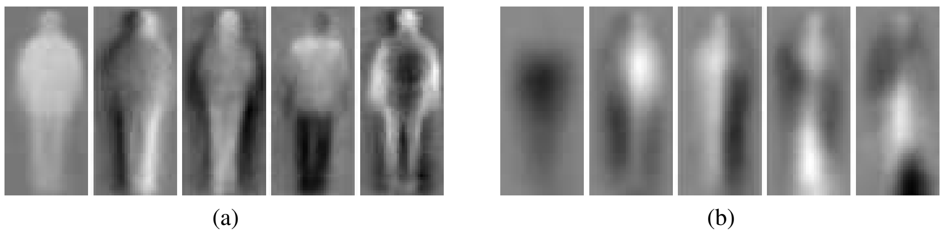


Fig. 4. Reconstructive model - first five principal vectors: (a) appearance-based model and (b) shape model.

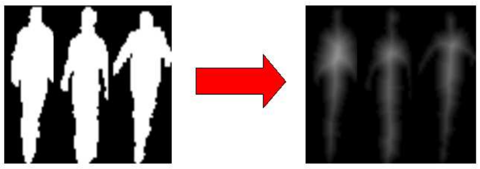


Fig. 5. Shape model obtained by Euclidean distance transform of binary shape images.

2.3 On-line AdaBoost for Feature Selection

In general, boosting converts (boosts) a weak learning algorithm into a strong one. Various variants of Boosting have been developed (e.g., Real-Boost [8], LP-Boost [7]) but here we focus on the discrete AdaBoost (adaptive boosting) algorithm introduced by Freund and Shapire [8]. It adaptively re-weights the training samples instead of re-sampling them and trains a weak classifier with respect to these weight distribution. Finally, a strong classifier is build from a linear combination of all the trained weak classifiers.

Boosting for feature selection was first introduced by Tieu and Viola [36] and later used by Viola and Jones [38] for face detection. The main idea is that each feature corresponds to a single weak classifier and that boosting selects an informative subset of N features from all possible features $\mathcal{F} = \{f_1, ..., f_M\}$. In each iteration step $n = 1, ..., N$ all features f_j are evaluated on all positive and negative training samples $\mathcal{X} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_L, y_L)\}$, where $\mathbf{x}_i \in \mathbb{R}^m$ is a sample and $y_i \in \{-1, +1\}$ is the corresponding label. A hypothesis is generated by applying the learning algorithm $\mathcal{L}$ with respect to the weight distribution $p(\mathbf{x}_i)$ over the training samples which is initialized uniformly. The best hypothesis is selected and forms the weak classifier h_n^{weak}. The weight distribution $p(\mathbf{x}_i)$ is updated according to the error of the current weak classifier. The process is repeated until N features are selected (i.e., N weak classifiers are trained). Finally, a strong classifier h^{strong} is computed as a weighted linear combination of all weak classifiers where the weights α_n are calculated according to the errors of h_n^{weak}.

In our work we use three different types of features, which are Haar-like features [38], Orientation histograms [5,14], and a simple version of local binary patterns (LBP) [22]. Note, that the computation of all feature types can be done very efficiently using integral images [38] and integral histograms [28] as data structures. This allows to do exhaustive template matching when scanning the whole image.

The method as described above works off-line; all training samples must be given in advance which is not the case for many applications. In order to get an on-line algorithm, all steps have to be on-line. Therefore, we use the on-line AdaBoost algorithm proposed by Oza [24, 25]. In contrast to the off-line version, the weak classifiers are updated whenever a new training sample arrives. Since we can not calculate the weight distribution the basic idea of on-line boosting is that the importance λ of a sample can be estimated by propagating it through the set of weak classifiers. This allows to adaptively train a detector and to efficiently generate the training set. In addition, a classifier is available at any time. Moreover, Oza [25] has proved, if off-line and on-line boosting are given the same training set the weak classifiers returned by on-line boosting converges statistically to the one obtained by off-line boosting as the number of iterations $N \rightarrow \infty$.

In the following we will shortly discuss on-line boosting for feature selection that was proposed by Grabner and Bischof [9]. They introduce "selectors" and perform on-line boosting on these selectors and not directly on the weak classifiers. Each selector $h^{sel}(\mathbf{x})$ holds a set of M weak classifiers $\{h_1^{weak}(\mathbf{x}), \ldots, h_M^{weak}(\mathbf{x})\}$ and selects one of them

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

according to an optimization criterion (using the estimated error e_i of each weak classifier h_i^{weak} such that $m = \arg\min_i e_i$).

Thus, on-line boosting for feature selection can be summarized as follows: first, a fixed set of N selectors $h_1^{sel}, .., h_N^{sel}$ is initialized randomly with weak classifier (i.e., features). 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 λ of the current sample. For updating the weak classifiers any on-line learning algorithm can be used. To obtain the weak classifiers h_j^{weak} the feature f_j is evaluated on the sample image **x**. The weak classifier with the smallest estimated error $e_n^\star$ is selected by the selector and the corresponding voting weight $\alpha_n = \frac{1}{2} \ln \left(\frac{1-e_n^\star}{e_n^\star} \right)$ and the importance weight λ of the sample are updated and passed to the next selector h_{n+1}^{sel}. The weight is increased if the example is misclassified by the current selector or decreased otherwise. Finally, a strong classifier is obtained by linear combination of N selectors:

$$h^{strong}(\mathbf{x}) = \text{sign}\Big(\sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(\mathbf{x}) \Big) . \tag{4}$$

2.4 Conservative Update Rules

Having the reconstructive models described in Section 2.2 each image patch can be checked whether it is consistent with them or not. Figs. 6(a,b) depict an image and its appearance and shape reconstructions in the case of a correct and a false detection. In the latter case, the reconstruction error for both, the appearance-based and the shape model are significantly larger (i.e., the original image and its reconstruction differ significantly) whereas we get small reconstruction errors for the correct example.

Thus, the reconstruction error can be considered a meaningful criterion for our update decisions. In the conservative learning framework we perform updates only if we are very confident, in particular we used following update rule: the current discriminative classifier is applied to a training image and all patches that were labeled as object are verified by the reconstructive model. If the reconstruction error for both, appearance and shape, is very low there is a positive update of the classifier; if the reconstruction errors are big there is a negative update. Otherwise there is no update. As this decisions are based on very conservative thresholds most patches are not considered at all. This update rule is illustrated in Fig. 7.

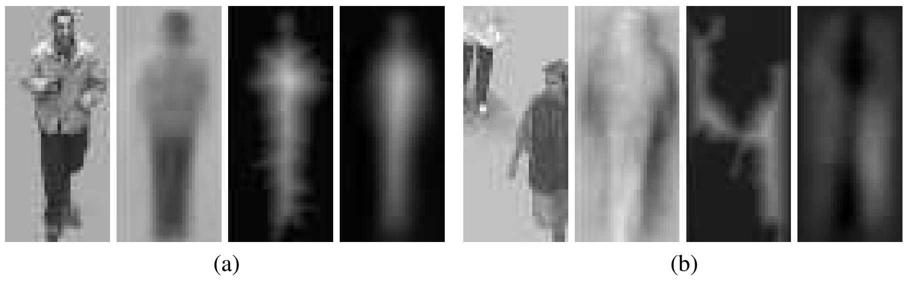


Fig. 6. Appearance image, its reconstruction, shape image, its reconstruction: (a) in case of correct detection, (b) in case of false detection.

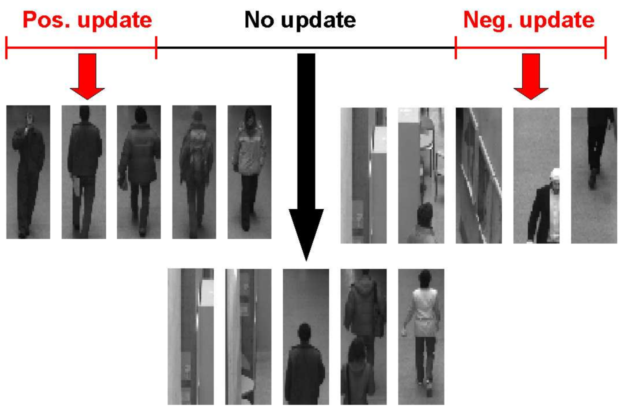


Fig. 7. Conservative updating the discriminant classifier.

Finally, in Fig. 8 we illustrate the incremental learning process by visualizing the updates. Therefore, the first, the third, the 34th and the 64th frame of a sequence of total 300 frames are shown. A red bounding box indicates a negative update, a green bounding box a positive update and a white bounding box indicates a detection that is not used for updating the existing classifier. It can be seen that there is a great number of negative updates within the very first frames (see Figs. 8(a,b)). If the incremental learning process is running for a longer time we finally get a stable classifier and only a small number of updates are necessary (see Figs. 8(c,d)).

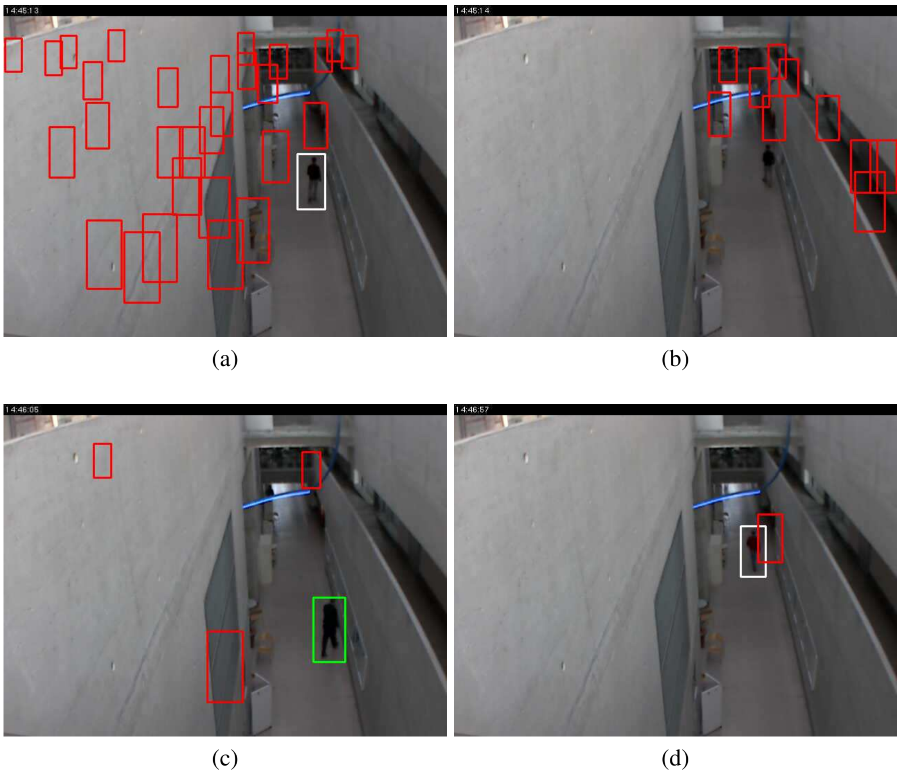


Fig. 8. On-line Conservative Learning: (a) updates for 1st frame, (b) updates for 3rd frame, (c) updates for 34th frame, and (d) update for 64th frame.

3 Experimental Results

3.1 Description of Experiments

For testing our framework we used two different surveillance video sequences. For the first one (*CoffeeCam*), showing a corridor in a public building near to a coffee dispenser, we have recorded images over several days. A simple motion detector triggers the camera and then each second one image is recorded. In total we have recorded over 35000 images. To train the classifiers a sequence containing 1200 frames has been used. For evaluation purposes we have generated a challenging independent test set of 300 frames (containing groups of persons, persons partially occluding each other and persons walking in different directions) and a corresponding ground truth.

The second sequence (*Caviar*), showing a corridor in a shopping center, was taken by the CAVIAR project and is publicly available[1]. There is a great number of short sequences that have been joined to a single one. To avoid redundancy the frame rate was reduced to approx. 1 fps (only every 25th frame was stored). For evaluation purposes an

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1; January 29, 2007

independent test set of 144 frames was created. Note that CAVIAR provides a ground truth in XML format which we used for our evaluation. This ground truth annotates also persons which are only partially visible (e.g., only a hand or head) which, in fact, can not be detected by our system.

To monitor the progress during on-line training after several training images the classifier obtained up to now is evaluated on the same independent test sequence. Therefore, the experiments are split into two main parts: First, we trained and evaluated classifiers on the *CoffeeCam* sequence. Second, to demonstrate the on-line adaptation capability, a classifier trained on the *CoffeeCam* sequence was applied to the *Caviar* test set. In addition, we show detection results obtained from classifiers that was trained using conservative learning. Therefore, we evaluated a person detector on a very complex sequence (a crowd in a lobby of a public building) and a car detector on a sequence showing a tunnel.

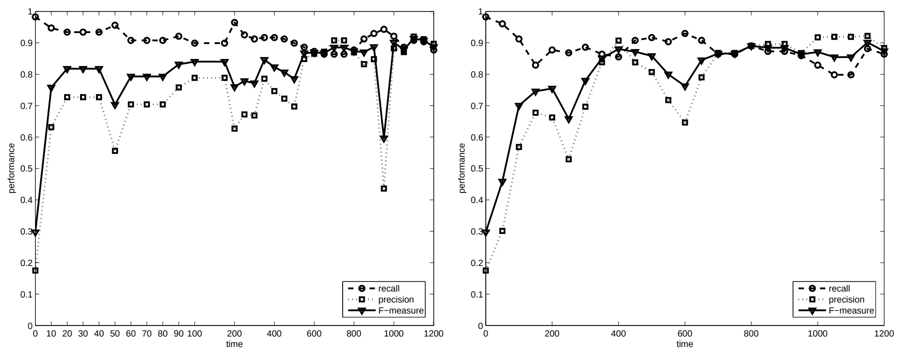
3.2 CoffeeCam

First, we need an initialization stage to collect positive and negative samples applying a motion based classifier. All patches where motion detection has detected an object are selected as positive examples. The negative examples are obtained by randomly sampling regions where no motion was detected (*AdaBoost1*). Since the motion detector returns approx. 10% false positives a robust reconstructive representation (PCA on appearance and shape) is computed from the output of the motion detector. Thus, the false positives can be filtered out and may be used as negative examples (*AdaBoost2*). Next, we can use the thus obtained data sets to train an initial AdaBoost classifier and PCA models for appearance and shape and start the on-line training. The PCA models were updated using the incremental PCA later on.

Let us have a look at the results obtained by on-line learning. As an evaluation criterion we used similar to [1], precision, recall and the F-measure that can be considered as trade-off between recall and precision. Fig. 9 depicts the performance curves if we start on-line training from *AdaBoost2*. One can see a clear improvement (especially in the first 100 steps) where a lot of false positives can be eliminated. The sudden decrease in performance around frame no. 900 is caused by a single background patch that is detected as a false positive over the whole test sequence; after the next update the curves get back to the previous level.

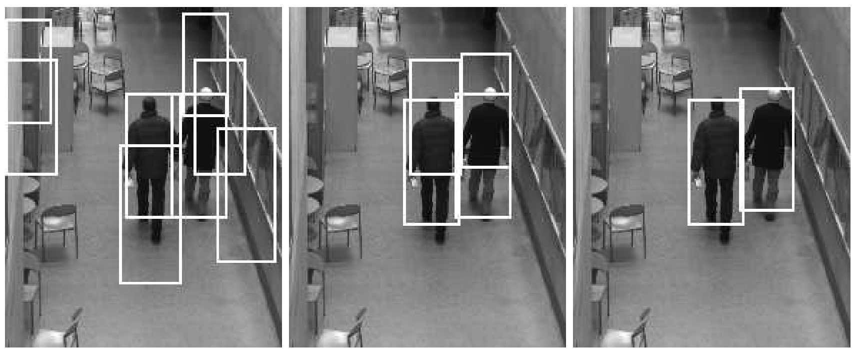
Fig. 9(b) depicts the performance curves if we start on-line training from *AdaBoost1*. Since this initial classifier is worse there is a greater number of false positives in the beginning. Therefore more than 300 frames are necessary to obtain comparable results to the previous classifier but finally we get the same performance as for the first test case! This example demonstrates that it is beneficial (1) first to perform a few steps of off-line learning at the beginning and then switch to an on-line version and (2) to use clean data for training.

Fig. 10 shows the detections by three different on-line classifiers (initial, after 300 and after 1200 training frames) on the test sequence. There are many false positives in (a) that can be completely removed by on-line training, as can be seen in (b) and (c). Fig. 11 depicts some more examples of persons correctly detected by the final classifier that was trained with 1200 frames.



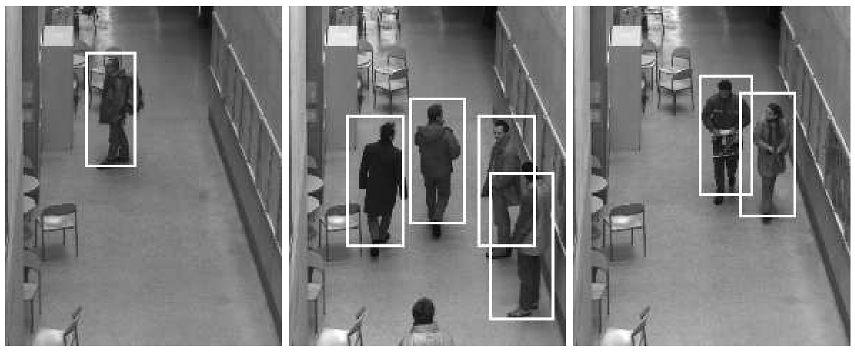
(a) (b)

Fig. 9. (a) On-line learning started from *AdaBoost2*, (b) On-line learning started from *AdaBoost1*.



(a) (b) (c)

Fig. 10. Improvement of on-line classifier: (a) initial classifier, (b) after 300 frames and (c) after 1200 frames.



(a) (b) (c)

Fig. 11. Detections by the final classifier trained with 1200 frames.

Finally, we want to show that the on-line algorithms are comparable to the off-line versions of the methods. Therefore we have trained classifiers of different stages using the off-line framework we have proposed in [30] and evaluated them on the test sequence. The initialization stage (collect patches and build an initial classifier) is the same as described above (*Off-line AdaBoost1* and *Off-line AdaBoost2*). To train a new classifier the current classifier is evaluated on another sequence. Thus, new positive and negative examples are added to the current training set; a new classifier is trained (*Off-line AdaBoost3*). Table 1 depicts these increasingly better results compared to the final classifiers obtained by the on-line framework.

method	recall	prec.	F-m.
Off-line AdaBoost1	97.4 %	27.5 %	42.9 %
Off-line AdaBoost2	91.9 %	57.4 %	70.7 %
Off-line AdaBoost3	93.6 %	94.8 %	94.2 %
AdaBoost1	86.4 %	88.3 %	87.4 %
AdaBoost2	87.7 %	89.7 %	88.7 %

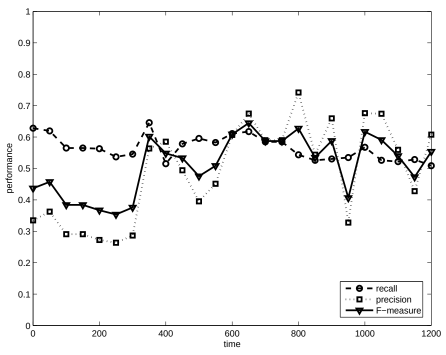
Table 1. Experimental results of the off-line and of the on-line framework.

3.3 Switch to Caviar

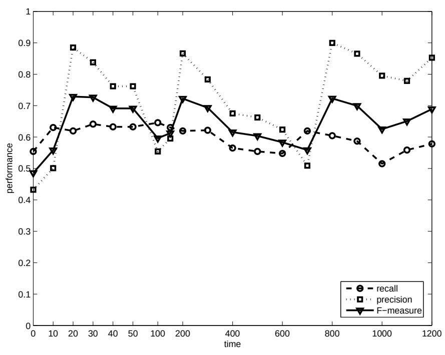
After we have shown that the on-line learning framework is working on the *CoffeeCam* data we want to demonstrate two interesting aspects: First, we show that the classifiers trained on the *CoffeeCam* data describe a generalized person model. Therefore Fig. 12 depicts the performance of the classifiers while training them on *CoffeeCam* training data and evaluating them on the *Caviar* test sequence. One can clearly see that the precision (and therefore the F-measure) is improved by on-line training while the recall is roughly constant. This shows that a person model is learned that generalizes over specific setups.

Second, we demonstrate that the on-line learning framework is able to adapt to a completely different scene. Therefore we perform on-line training on the *Caviar* data set starting with a classifier obtained by the *CoffeeCam* training. Due to the compression noise a new model for appearance is required. Furthermore motion detection can not be applied to the *Caviar* data set, because the quality of the motion blobs is too bad for classifying based on size and aspect ratio restrictions only. Since the shape model is more robust (holes etc.), the shape model estimated from the *CoffeeCam* data can be used to collect patches for generating an appearance based PCA model. The PCA models (appearance and shape) were updated using the incremental PCA later on.

Fig. 12(b) depicts the evaluation results of this experiment. The main improvement is achieved within the first 100 frames (false positives in the background). The precision (and therefore the F-measure) can be clearly increased. Please note that the obtained performance of approx. 60% detection rate might look quite low, but this is actually



(a)



(b)

Fig. 12. (a) Performance of *CoffeeCam* classifier evaluated on *Caviar* test set, (b) Performance of Caviar *on-line* learning.

quite good considering the given ground truth. If we exclude all persons that are not at least 50% visible we get a detection rate of approx. 79%.

The noisy behavior of the curves can be explained by the nature of on-line learning and the way we perform the evaluation. We have a fixed test set, therefore it may happen that particular cases occurring in the test set are not occurring in the training sequence for some time. Thus, the classifier will not perform well on this particular data, i.e., if this happens to parts of the background that are visible most of the time. Examples of the finally obtained classifier that was trained with 1200 frames are shown in Fig. 13.

3.4 Further Detection Results

To demonstrate that the proposed framework even works on more complex scenarios we show detection results on a *crowdy scene*. Therefore, a classifier was trained on-line from 1000 frames. This classifier was then evaluated on an independent test sequence of 600 frames. Examples of detections are shown in Fig.14.

Finally, we show that the proposed framework is quite general and is not limited to train a person model. Therefore, Fig. 15 shows results of a car detector applied in a tunnel that was trained using on-line conservative learning.

4 Summary and Conclusions

We have presented an on-line learning framework that avoids hand labeling of training data. This framework has been used on challenging person detection tasks. We have demonstrated that on-line learning obtains comparable results to off-line learning. We have also shown that we can adapt an already trained person detector to a quite different set-up (i.e, by reducing the number of false positives). While the *CoffeeCam* sequence was obtained with a high quality camera looking down a corridor, the *Caviar* sequence was obtained with a consumer video camera mounted almost horizontally.

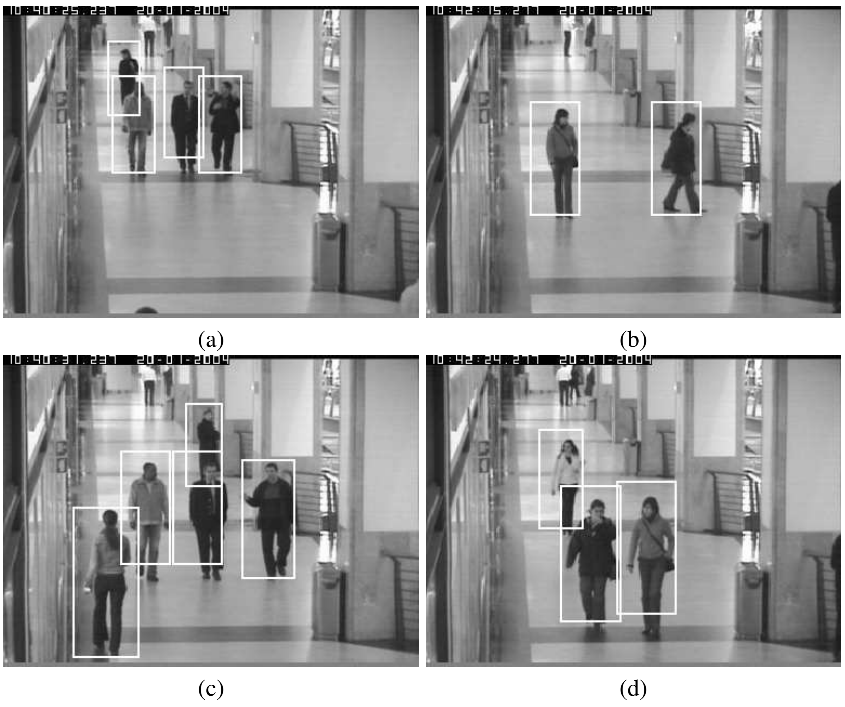


Fig. 13. Detections by the final classifier trained with 1200 frames.

Moreover, the sequences contain a considerable amount of compression noise. Nevertheless, the on-line framework was able to adapt to this quite different scenario. The proposed framework is quite general. It can be used to learn completely different objects (e.g., cars) and can be extended in several ways. More modules (generative and discriminative classifiers) operating on different modalities will further increase the robustness and generality of the system. In particular, we will add a tracking algorithm to obtain a wider variety of positive samples (which in turn should increase the detection rate).

References

1. S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.
2. H. Breu, J. Gil, D. Kirkpatrick, and M. Werman. Linear time euclidean distance transform algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(5):529–533, 1995.
3. S. Chandrasekaran, B. S. Manjunath, Y.-F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, 1997.

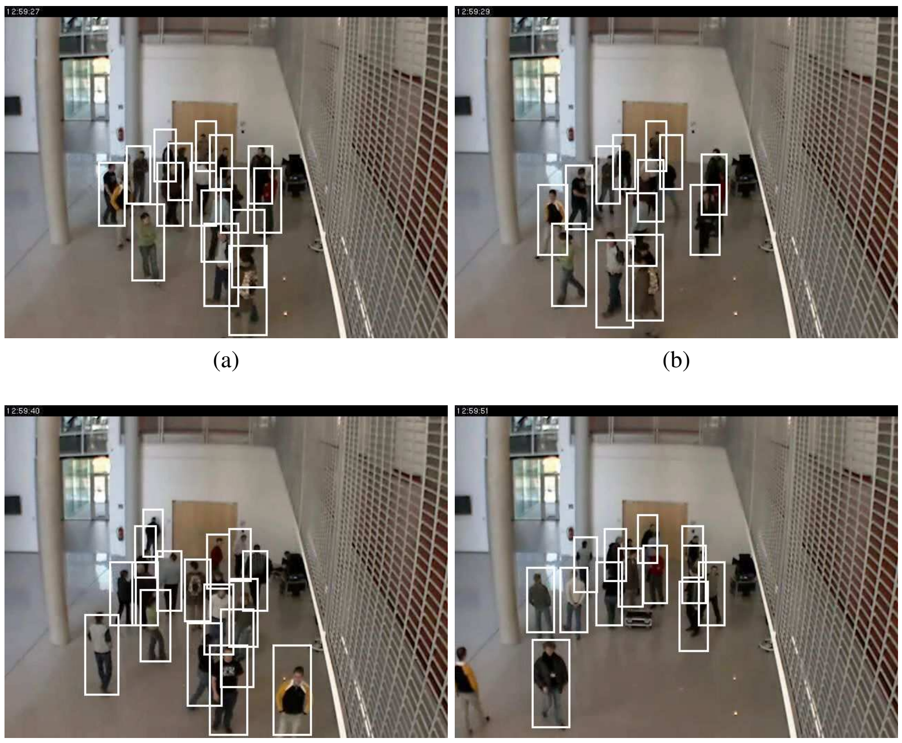


Fig. 14. Conservative Learning Framework on a more complex scene.

4. S.-C. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *Proc. SPIE Visual Communications and Image Processing*, pages 881–892, 2004.
5. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.
6. F. de la Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1):117–142, 2003.
7. A. Demiriz, K. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.
8. Y. Freund and R. E. Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.
9. 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.
10. D. Hall, J. Nascimento, P. C. R. E. Andrade, P. Moreno, S. P. T. List, R. Emonent, R. B. Fisher, J. Santos-Victor, and J. L. Crowley. Comparision of target detection algorithms using adaptive background models. In *Proc. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 113–120, 2005.
11. P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

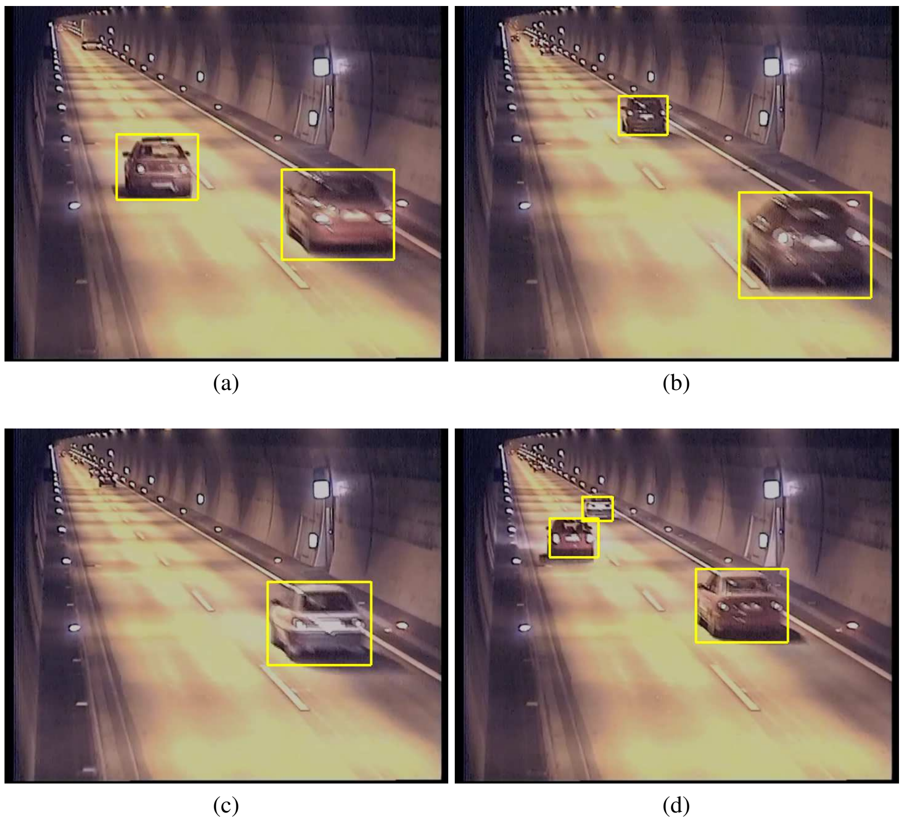


Fig. 15. Conservative Learning Framework applied for learning a car detector.

12. D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B.Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Proc. Intern. Conf. on Pattern Recognition*, volume I, pages 126–131, 1994.
13. A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, 2000.
14. K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, pages 53–60, 2004.
15. 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.
16. A. Levy and M. Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. *IEEE Trans. on Image Processing*, 9(8):1371–1374, 2000.
17. Y. Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004.
18. N. Littlestone. Learning quickly when irrelevant attributes abound. *Machine Learning*, 2:285–318, 1987.
19. B. Lo and S. A. Velastin. Automatic congestion detection system for underground platforms. In *Proc. IEEE Intern. Symposium on Intelligent Multimedia , Video and Speech Processing*,

pages 158–161, 2001.

20. N. J. McFarlane and C. P. Schofield. Segmentation and tracking of piglets. *Machine Vision and Applications*, 8(3):187–193, 1995.
21. 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.
22. T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.
23. A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proc. European Conf. on Computer Vision*, volume II, pages 71–84, 2004.
24. N. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.
25. N. Oza and S. Russell. Online bagging and boosting. In *Proceedings Artificial Intelligence and Statistics*, pages 105–112, 2001.
26. J.-H. Park and Y.-K. Choi. On-line learning for active pattern recognition. *IEEE Signal Processing Letters*, 3(11):301–303, 1996.
27. M. Piccardi. Background subtraction techniques: a review. In *Proc. IEEE Intern. Conf. on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.
28. F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.
29. P. M. Roth and H. Bischof. On-line learning a person model from video data. In *Video Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2006.
30. P. M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. Conservative visual learning for object detection with minimal hand labeling effort. In *Proc. DAGM Symposium*, volume 3663 of *LNCS*, pages 293–300. Springer, 2005.
31. P. M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proc. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 223–230, 2005.
32. H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.
33. D. Skočaj, H. Bischof, and A. Leonardis. A robust PCA algorithm for building representations from panoramic images. In *Proc. European Conf. on Computer Vision*, volume IV, pages 761–775, 2002.
34. D. Skočaj and A. Leonardis. Weighted and robust incremental method for subspace learning. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume II, pages 1494–1501, 2003.
35. 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.
36. K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, pages 228–235, 2000.
37. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
38. 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.
39. P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume II, pages 734–741, 2003.