VISUAL ON-LINE LEARNING IN DISTRIBUTED CAMERA NETWORKS*

C. Leistner, P. M. Roth, H. Grabner, H. Bischof

Graz University of Technology
Inffeldgasse 16/2, 8010 Graz, AUSTRIA
{leistner,pmroth,bischof}@icg.tu-graz.ac.at

A. Starzacher, B. Rinner

Klagenfurt University
Lakeside B02, 9020 Klagenfurt, AUSTRIA
{firstname.lastname}@uni-klu.ac.at

ABSTRACT

Automatic detection of persons is an important application in visual surveillance. In general, state-of-the-art systems have two main disadvantages: First, usually a general detector has to be learned that is applicable to a wide range of scenes. Thus, the training is time-consuming and requires a huge amount of labeled data. Second, the data is usually processed centralized, which leads to a huge network traffic. Thus, the goal of this paper is to overcome these problems, which is realized by a person detection system, that is based on distributed smart cameras (DSCs). Assuming that we have a large number of cameras with partly overlapping views, the main idea is to reduce the model complexity of the detector by training a specific detector for each camera. These detectors are initialized by a pre-trained classifier, that is then adapted for a specific camera by co-training. In particular, for co-training we apply an on-line learning method (*i.e.*, boosting for feature selection), where the information exchange is realized via mapping the overlapping views onto each other by using a homography. Thus, we have a compact scene-dependent representation, which allows to train and to evaluate the classifiers on an embedded device. Moreover, since the information transfer is reduced to exchanging positions the required network-traffic is minimal. The power of the approach is demonstrated in various experiments on different publicly available data sets. In fact, we show that on-line learning and applying DSCs can benefit from each other.

Index Terms— visual on-line learning, object detection, multi-camera networks

1. INTRODUCTION

Due to an increasing number of surveillance cameras and limited available human resources for analyzing the upcoming data autonomous video analysis (*e.g.*, person detection or detection of unusual events) is becoming more and more important. In particular, in recent years there has been a crucial scientific interest to improve automatic person detection. Thus, more sophisticated data representations and more effi-

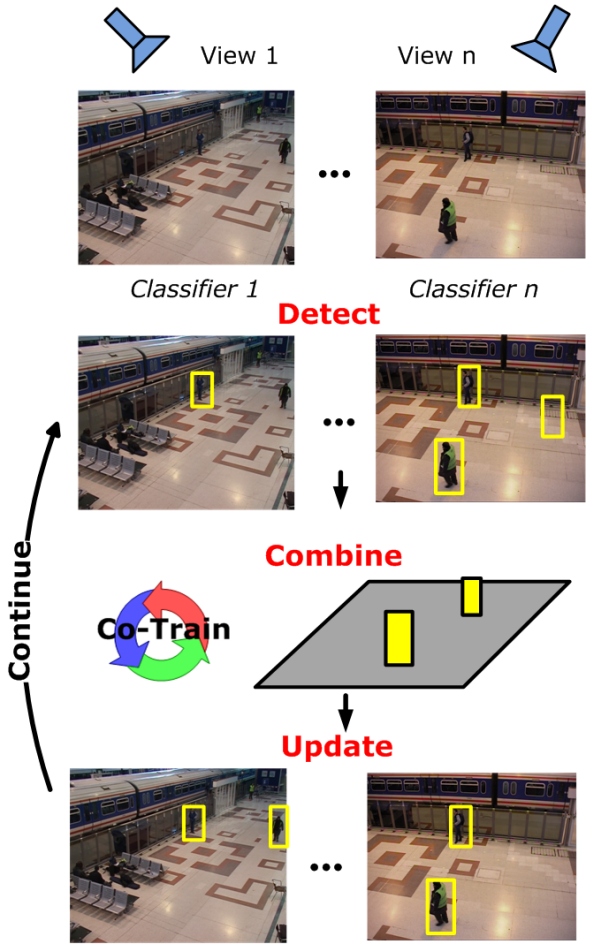


Fig. 1. A Scene is observed by multiple cameras with *partly* overlapping fields of views (first row). At each camera a classifier is applied to detect persons (second row). In order to improve the detection results (lower false positive rate as well as increasing detection rate) the classifier is updated (last row) in a co-learning manner by combining the decision of all n camera views (third row).

*This work was supported by the FFG project EVis (813399) under the FIT-IT program, 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.

cient learning methods were developed. In addition, due to large number of cameras the acquisition costs and the energy consumption for such systems are increasing and huge network resources and powerful computational back-ends are required. Therefore, distributed smart cameras (DSCs) are getting more and more established in autonomous surveillance systems since their advantages are manifold: low costs, scalability, low energy consumption, sustainability, *etc.*

Smart cameras, in general, combine video sensing, processing and communication on a single embedded platform [1, 2]. They represent a prominent example for embedded computer vision and have been the subject of study for quite some time in both, research labs and companies. Research on smart cameras intensified significantly in the last decade. Many prototypes have been implemented on various platforms (*e.g.*, [3–5]) and smart cameras have been successfully demonstrated in many applications (*e.g.*, [6–8]). A more detailed overview can be found in [9].

Moreover, smart cameras are highly applicable for distributed systems. Thus, DSCs embody the trend in sensor networks to increase in-network processing [10]. For multi-camera applications, image processing migrates from central workstations to the distributed embedded sensors. This distributed computing approach helps to reduce the communication load within the network of cameras and to increase the reliability and scalability of the multi-camera application. Such networks can take advantage of the basic techniques of ad-hoc networking developed for sensor networks, but they also need additional layers to manage the local and distributed processing.

However, these reduced resources (i.e., memory, computational power, and bandwidth) limit the computer vision methods, that can be applied. Thus, common approaches collecting data over time at a central node, which is analyzed and merged later on, can only be applied to some extent. For instance, it is not possible to compute dense stereo [11], which allows for increasing the robustness in multi-camera systems. Neither can we use fusion methods on image data for detection, even though very good results are obtained using these methods on other platforms [12]. Thus, especially for person detection, in most DSC systems each mote operates isolated, mostly applying simple motion detection algorithms, which achieve adequate results. However, motion detectors yield high false detection rates (*e.g.*, due to shadows, reflections, illumination changes, *etc.*), cannot detect non-moving targets, and cannot discriminate between different kinds of (moving) objects.

In contrast, by using appearance-based methods based on powerful machine learning algorithms (*e.g.*, [13]) higher detection rates can be achieved while keeping false positive rates low. Such approaches, however, demand a huge amount of labeled training data. Yet in practice, there is often not enough labeled data available. Moreover, hand-labeling of data is tedious and in some cases not even feasible. Additionally, as these detectors are usually trained in the lab for broad application they have to cover a wide variety of possible scenarios and might thus be quite complex. But assuming a stationary camera setup, which is the case for most surveillance applications, we have only to cover a certain fixed scene. Thus, a specialized detector would perform better than a general detector in terms of both, accuracy and efficiency (e.g., [14]). In addition, since the complexity of the task is reduced the amount of required training samples can be reduced.

The contribution of this paper is twofold. First, we show that we can train a person detector on-line on a smart camera, which has the advantage that a detector can be tuned to a specific scene. Therefore, the complexity of the detector is reduced while the precision and the recall are increased. The second contribution is that we can train these detectors utilizing multiple partly overlapping cameras in a co-training (semi-supervised) [15] manner. In order to allow for camera collaboration we require only rough geometric knowledge (*i.e.*, ground-plane homography) of the scene. Then the cameras can teach each other by exchanging detection results (coordinates), which makes our approach also suitable for large camera networks. The overall principle of the proposed approach is illustrated in Fig. 1.

In particular, we apply the approach of Roth *et al.* [16] to train the off-line classifier, which we then steadily improve by co-training using on-line boosting for feature selection [17]. For co-training, we use the homography between two cameras, which allows to project a patch from one local camera coordinate system to another. Hence, only the location and not the whole image has to be transfered between the cameras, which dramatically reduces the required bandwidth. Moreover, since we apply an on-line learning method, a sample can be discarded directly after the update, which reduces the memory requirements and the computational costs for updating. In the experimental evaluation, we show that the proposed approach is highly suitable for embedded systems due to its thin required system resources. In order to show general applicability, we demonstrate the method on two publicly available data sets.

The remainder of this paper is as follows. In Section 2 we review co-training and on-line boosting for feature selection. Based on that, in Section 3, we introduce our co-training system using multiple camera networks with partly overlapping views. In addition, we show that the traffic within the multi-camera network is quite small and it is perfectly suitable for an embedded system, described in Section 4. Experiments on two challenging datasets for person detection in Section 5 further illustrate these advances. Finally, we conclude the paper with a summary and an outlook in Section 6.

2. PRELIMINARIES

2.1. Co-Training

It is well known (e.g., Nigam et al. [18]) that also unlabeled samples contain information about the joint distribution of the data. Since unlabeled samples can be obtained significantly easier than labeled samples the main goal would be to take advantage of the statistical properties of the labeled and unlabeled data by using a semi-supervised approach. Hence, a seed classifier, that was trained from a smaller number of labeled samples, can be improved by taking into account a large number of available unlabeled samples. This, in fact, is afforded by co-training, which was originally proposed by Blum and Mitchell [15].

The main idea is to split the instance space into two views[1] and to train a separate classifiers for each view. These classifiers are then applied in parallel, where one classifiers teaches the other (*i.e.*, unlabeled samples, that are confidently labeled by one classifier, are added to the training set of the other classifier). It was proven in [15] that co-training converges if two strong conditions are fulfilled. First, the two views must be conditional independent and, second, each of them should be able to solve the task. Hence, to satisfy these conditions training samples are required for which one of the classifiers is confident whereas the other one is not. Since it is hard to assure these conditions in practice – in particular the first one – these requirements were relaxed later on [19]. Nevertheless, a fairly strong assumption on the training algorithms remains, *i.e.*, they should never provide a hypothesis that is "confident but wrong".

For object recognition, co-learning was applied by Levin *et al.* [20]. In an off-line setting, they start with a small number of hand labeled samples and generate additional labeled examples by applying co-training of two boosted classifiers. One is trained directly from gray-value images whereas the other is trained from background subtracted images. The additional labels are generated based on confidence-rated predictions. After some samples are newly labeled (using confidence-rated update rules), the training process is started again from scratch. In general, the approach is not limited to two views but can be extended to multiple views (e.g., [21, 22]). Zhou and Li [22] extended the original co-training approach for three classifiers. Moreover, Javed et al. [21] apply an arbitrary number of classifiers and extended the method for on-line learning. In particular, they first generate a seed model by off-line boosting, which is improved later on by on-line boosting. The co-training is then performed on feature level, where each feature (*i.e.*, global PCA features) corresponds to a base classifier. If an unlabeled sample is labeled very confidently by a subset of such base classifiers it is used for both, updating the base classifiers and the boosting parameters.

[1]For instance, in order to discriminate between apples and bananas one "view" might be shape while the other might be color.

2.2. On-line Boosting

Boosting, in general, is a widely used technique in machine learning for improving the accuracy of any given learning algorithm [23]. In this work, we focus on the (discrete) AdaBoost algorithm, which has been introduced by Freund and Shapire [24]. The goal is to learn a binary classifier, *i.e.*, to learn a mapping $H : \mathbf{x} \rightarrow \{-1, 1\}$. This classifier

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

corresponds to a linear combination of N weak classifiers h_n. A weak classifier has to perform only slightly better than random guessing and is trained using a weight distribution (initialized uniformly) over the set of labeled training samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$. With respect to the error of the trained weak classifier h_n the corresponding voting weight α_n as well as the weight distribution are updated. This is repeated until a certain stopping criterion is met.

Furthermore, as has been shown by Friedman *et al.* [25], boosting provides a confidence measure

$$P(y = 1|\mathbf{x}) = \frac{e^{H(\mathbf{x})}}{e^{H(\mathbf{x})} + e^{-H(\mathbf{x})}}. \tag{2}$$

Boosting can be also applied for feature selection [26]. The basic idea is that each feature corresponds to a weak classifier and that boosting selects an informative subset from these features. In fact, various different feature types may be applied, but similar to the seminal work of Viola and Jones [27] in this work we use Haar-like features, which can be calculated efficiently using integral data-structures.

Boosting, was originally developed for off-line learning, *i.e.*, all training samples must be given in advance. In contrast, for on-line learning methods at each time only *one* training sample $(\mathbf{x}, y)$ is provided to the learner. Since for many applications it is advantageous having an on-line learning method Grabner and Bischof [17] introduced an on-line approach for boosting for feature selection, which is based on the on-line boosting algorithm of Oza *et al.* [28]. The main idea is to introduce *selectors* and to perform on-line boosting on these selectors and not directly on the weak classifiers. A selector $h_n^{sel}(\mathbf{x})$ can be considered a set of M weak classifiers $\{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$. Once a fixed number of N selectors $h_1^{sel}, .., h_N^{sel}$ was initialized with random features the selectors are updated whenever a new training sample $(\mathbf{x}, y)$ is available. For that purpose, all weak classifiers within a selector are updated and the weak classifier with the smallest estimated error is selected and the voting-weight α_n for the n-th selector h_n^{sel} is updated. Since no weight distribution over the training samples is available the importance λ (initialized with 1) of an example is used for training instead, where the sample and the importance are propagated through the set of selectors. In fact, it is increased if the sample is mis-classified

and decreased otherwise. Thus, the algorithm focuses on the hard examples.

Contrary to the off-line version, an on-line classifier is available at any time of the training process, which allows continuous learning and improving (*i.e.*, re-training) an existing classifier.

3. SYSTEM APPROACH

The overall camera network is depicted in Fig. 2. We have a setup with n partly overlapping cameras, each of them observing the same 3D scene. In general, the objects-of-interest can move in the world coordinate system $\{x_w, y_w, z_w\}$. But since the main goal in this paper is to learn a person detector, we can assume that the objects-of-interest (*i.e.*, the persons) are moving on a common ground-plane. However, having overlapping camera views the local image coordinate system $\{x_i, y_i\}$ can be mapped onto each other by using a homography based on an identified point in the ground-plane. In addition, for each camera an estimation of the ground-plane is required. Both, the calibration of the ground-plane and the estimation of the homography, are discussed more detailed in Section 3.1. Once we have calibrated the scene we can start co-training. In fact, in our approach the different views on the data are realized by different camera views. The thus defined co-training procedure is discussed in Section 3.2. In addition, in this section we summarize the required system resources for the proposed approach, *i.e.*, the memory requirements, the computational costs, and the necessary data transfer between the cameras.

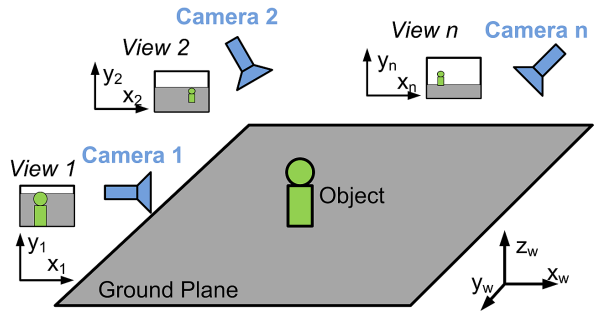


Fig. 2. System overview of the proposed approach: multiple partly overlapping cameras observe a scene and collaborate during update phase.

3.1. Scene Calibration

As illustrated in Fig. 2 the size of one and the same object in the camera coordinate system depends on the absolute position of the object within the world coordinate system. Since the objects (*i.e.*, the persons) are constrained to move on the ground-plane we can estimate the ground-plane for all cameras obtaining the approximative expected size of the object on a specific position on the ground-plane. In fact, for the current setups we estimated the ground-plane manually by selecting at least four corresponding points in each image, but to have an autonomous system an unsupervised autonomous approach (*e.g.*, [29]) might be applied as well.

In addition, similarly to Khan and Shah [30], we use homography information to map one view onto another. It is well known (see, *e.g.*, [11]) that points on a plane from two different views are related by a planar homography. The principle is depicted in Fig. 3.

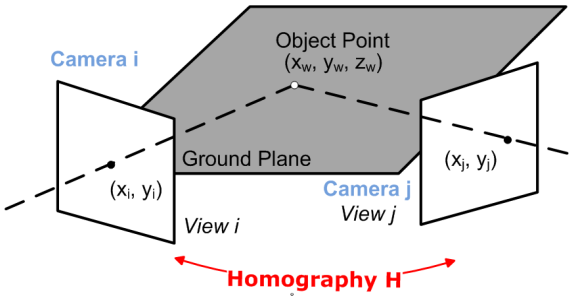


Fig. 3. Homography induced by a plane.

Hence, the plane induces a homography $\mathbf{H}$ between the two views, where the homography $\mathbf{H}$ maps points $\mathbf{x}_i$ from the first view to points $\mathbf{x}_j$ in the second view:

$$\mathbf{x}_j = \mathbf{H}\mathbf{x}_i \ . \tag{3}$$

In particular, we estimate the homography by selecting corresponding points manually. Once we have estimated the homography $\mathbf{H}$ between all matrices we can map one view onto another. For that purpose, we first estimate the base points of detections in all camera coordinate systems. Then, these base points are mapped onto a different camera coordinate system by the estimated homography. Finally, by using the thus obtained new base points in the new view we can superimpose the detection (*i.e.*, the bounding-box) from original view onto the projected one. In this way, we can verify if a detection in one view was also reported in a different one.

3.2. On-line Co-training

Similar to Levin *et al.* [20] in our co-training framework we apply boosted classifiers. Thus, since the strong classifier in Eq. (1) provides the probability given in Eq. (2), which can be interpreted as a confidence measure, we can apply confidence-based learning. But in contrast to existing approaches our approach differs in two main points: first, we consider the different camera views as independent views for co-training and, second, we apply an on-line method, which ensures a more efficient learning.

To start the training process, we first train a general prior H^P by off-line boosting. Thus, a classifier is trained using a fixed set of positive $\mathcal{X}^+$ and negative labeled samples $\mathcal{X}^-$ [27]. Such a classifier is trained emphasizing on a high recall rate rather than on a high precision and can therefore be applied on all different views. Then, this classifier H^P is cloned and used as an initial classifier for all camera views. Even exactly the same classifier is applied for that purpose due to the different camera positions we get the independent observations required for co-training. Since it has been shown in [16] that off-line classifiers can be easily re-trained using on-line methods (*i.e.*, all acquired statistics are interpreted as if they have been estimated on-line) these cloned classifiers can be re-trained on-line and adapted for a specific camera.

In particular, to improve the corresponding classifiers, in the following we propose a re-training approach for updating n cameras, where we verify or falsify the obtained detections. Assuming a low-bandwidth scenario we want to avoid to exchange any image data in terms of pixel information. Thus, to minimize the required data exchange we perform collaboration (*i.e.*, co-training) only on patches of interest. For that purpose, we define a confidence threshold Θ to classify the predictions. All predictions $H_i \geq \Theta$ of classifier i are assumed to be "correct" decisions and thus are not further considered. From all others, regardless if the prediction is positive or negative, the corresponding confidences of the other classifiers are requested. Thus, assuming that the response is $0 < H_i(\mathbf{x}) < \Theta$ two different cases have to be considered.

Due to homography projection errors and classification errors, bounding boxes might be wrong aligned (label jitter). Especially, for positive updates, wrong alignment might lead to noisy updates, which ends up in drifting and thus in corrupt classifiers. This could be limited by using motion information. But as discussed in Section 1, other approaches, that do not depend on a motion cue require central information merging, which makes them hardly applicable in a typical smart camera setup without a central node.

Therefore, we propose a simple strategy, which limits the problem of wrong detection alignments without having to build a central confidence map. Each classifier holds only the confidence map of its own predictions. For all requested subpatches $\mathbf{x}$ for which $|H((\mathbf{x})| \geq \Theta$ the confidence map is super-imposed with the requested confidences of the diverse corresponding classifiers on the projected locations. After that, in order to get the "interesting" positive samples and the "interesting" negative decisions, in a post-processing step simple non-maximum suppression and non-minimum suppression are applied on the merged confidences, respectively. Then, only such samples are used for updating, which have the highest positive and highest negative disagreement. To further increase stability, we keep a small pool of "correct" patches, in order to perform an additional conservative verification and falsification step. We will further refer to these update strategies as *verification* and *falsification*.

Verification If all responses of the other classifiers $j = 1, \ldots, n$ are also positive, the example is verified and added to the pool of positive examples: $\mathcal{X}^+ = \mathcal{X}^+ \cup \mathbf{x}$.

Falsification If all responses of the other classifiers are negative, the example is classified as a false positive. Thus, the classifier H_i is updated immediately using $\mathbf{x}$ as a negative example. After each negative update the pool of positive samples $\mathcal{X}^+$ is checked if it is still consistent; otherwise a positive update is performed with the corresponding sample.

With this very conservative update strategy the arising label noise can be minimized and thus the detections keep stable over time (*i.e.*, drifting can be limited). Since we are continuously learning over time these few updates are sufficient to adapt to the specific scene. Moreover, since the updates are the computationally most expensive steps by minimizing the number of required updated the over-all runtime can dramatically be reduced. Note, if H_i has a negative response nothing has to be done. However, the positive samples are collected during the verification step and the local pool of scene-specific samples $\mathcal{X}$ increases.

3.3. Resources

It has been shown that off-line trained object detectors (*i.e.*, obtained by boosting for feature selection) are highly suitable for embedded systems (*e.g.*, [31]). The main advantage is that the applied classifier can be trained using a powerful computer and finally only the compact representation consisting of a small number of features has to be stored on the embedded device. In contrast, when applying an on-line learning method (*i.e.*, on-line boosting for feature selection) a huge amount of features (*i.e.*, $\mathcal{O}(NM)$, where M corresponds to the number of selectors and N to the features for each selectors) has to be stored on the embedded system. Thus, in the following we discuss the advantages of the proposed approach considering the required resources (especially memory) and show that the method can even be applied if the system's resources are limited.

As features, which correspond to weak classifiers, we use similar to [27] Haar-wavelets. Since these features can have different sizes, aspect ratios, and locations within a given subwindow all of these variations have to be stored. For instance, even considering a small window size of 64×32 pixels[2], we have to select from several hundred thousand different features, in order to add only few selected to our final ensemble classifier. Even for the simplest feature type, we have to store at least two rectangles (each consisting of one x and one y coordinate, respectively, as well its width and height). Additionally, for each feature its statistics (*i.e.*, mean and variance for

[2]This is the typical patch size used for person detection.

both positive and negative distributions) and its final decision threshold Θ have to be stored . Again considering a 64×32 patch, such a system generates a maximum of $2,655,680$ features, resulting in at least 240 MB of required memory. Note, this number grows dramatically [32] with the training patch size. Fortunately, this set is highly over-complete (*i.e.*, only a small sub-set, usually 10%, is required in order to get proper results). Hence, choosing a subset of only 10% of all possible features reduces the required memory to 24 MB. For embedded systems, this amount, however, can still be far too high. Since in our approach on-line learning allows for training highly scene-specific classifiers, they can be very compact. In a typical scenario, we require only 100 selectors, where each selector typically holds 150 different features, resulting in a total number of $15,000$ features to be stored. Hence, we only need 500 KB of memory, which finally is a reasonable amount for most embedded platforms.

Considering the computational complexity, the algorithm performs in $\mathcal{O}(N)$ for detecting target objects. Scanning the window is very fast due to integral image structures, which allow to evaluate each rectangle sum in constant time. Moreover, the updates can be performed very efficiently in $\mathcal{O}(NMS)$, where S is the number of new samples. In our approach, we use only $S = 2$ per frame.

Since in our approach each mote acts as autonomously as possible no visual information, *i.e.*, images, has to be exchanged among the cameras. Thus, from each view, only a certain number of sub-patches confidence responses $\mathcal{O}(|H(\mathbf{x})| < \Theta)$ is required for co-training and have to be transfered between the cameras. For that purpose, only the corresponding coordinates (at overlapping areas) and their confidences have to be transmitted. Hence, depending on the choice of Θ, typically only a few hundred bytes per frame have to be exchanged.

4. EMBEDDED PROTOTYPING PLATFORM

For our experimental evaluations we use a high-performance embedded platform, which is shown in Fig. 4. The MICROSPACE EBX (*MSEBX945*) embedded computer board from *DigitalLogic AG* serves as single-camera platform. It offers a compact EBX single-board construction (146mm $\times$ 203mm), that allows for flexible positioning in real-world settings. Additionally, it supports various interfaces such as RS-232, LAN 100MB, FireWire over MiniPCI and USB. The CPU-module (*SMX945-L7400*) consists of an Intel Core 2 Duo CPU running at 2×1500 MHz with a 667 MHz FSB. The main characteristics of the MSEBX945 platform can be summarized as: 2048MB DRAM Min-Max, USB 2.0, RS232C, COM-Interface, 10/100BASE-T, 1GB-LAN PCIe, Mini PCI Slot and PS/2 Interface. The total power consumption lies in the range of 12–15 W.

Note that this platform is also capable of interfacing additional sensors as well as performing various sensor fusion algorithms [33]. We have implemented and tested interfaces to the following sensors:

- *Laser sensor (Noptel CM3-30)* for distance measurements and generation of altitude profiles
- *Audio sensor (via USB)* for stereo audio recordings
- *Vision sensor (Baumer FWX14-K08 camera)* for single shot and video streaming (resolution 1392×1040).
- *Environmental sensor (ELV ST-2232)* for light and temperature measurements

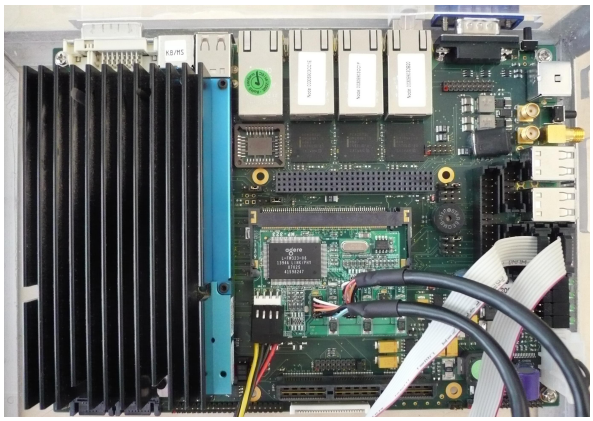


Fig. 4. Our embedded prototyping platform MSEBX945.

5. EXPERIMENTS

In all experimental setups we assume static cameras with partly overlapping views. For all cameras sharing a view-point area, we estimate the ground-plane homography as described in Section 3. The cameras are synchronized using simple NTP protocol[3] and are interconnected with each other over common Ethernet. To get the learning process started we trained an initial off-line classifier using ten positive labeled samples and twenty randomly chosen negative samples, respectively. We favored higher recall rate than higher precision. To illustrate the high suitability of our approach in practice as well as to increase comparability, we performed the experiments on standard surveillance datasets for multiple cameras.

5.1. PETS 2006

In our first experiment, we evaluated our approach on the *PETS 2006*[4] data set. In particular, the dataset shows the concourse of a train station from four different views. For our experiments we have selected sequences from two of the four views (frontal view/Camera 3 and side view/Camera 4),

[3] `http://www.ntp.org`, (June 28, 2008)

[4] `http://www.pets2006.net`, (June28 ,2008)

which are different in view angle, size, and geometry. For training and evaluation we selected independent sequences from Dataset S7 (Take 6-B) and Dataset S5 (Take 1-G). Although precise calibration data for the *PETS 2006* data set is publicly available, the homography was estimated as described before.

To start the learning process the initial classifier was evaluated on both camera views. Later these initial classifiers were updated by co-training. To demonstrate the learning progress after a pre-defined number of processed training frames we stored a classifier, which was then evaluated on the independent test sequence. The thus obtained results (*i.e.*, we evaluated the recall, the precision, and the F-measure) for specific time stamps t (*i.e.*, $t = 0$, $t = 20$, and $t = 50$) are summarized in Table 1:

Classifier 1			
t	recall	precision	F-measure
0	0.85	0.32	0.46
20	0.82	0.44	0.58
50	0.85	0.78	0.82

Classifier 2			
t	recall	precision	F-measure
0	0.72	0.68	0.70
20	0.75	0.69	0.72
50	0.75	0.95	0.84

Table 1. *PETS 2006*: recall, precision, and F-measure for *classifier 1* and *classifier 2* after $t = 0$, $t = 10$ and $t = 50$ iterations.

It can be seen, that even only a small number of frames were processed the precision was significantly improved while the recall rate stays at the same level. The same can be seen from the precision-recall curves for both classifiers, which are shown in Fig. 5. Moreover, the improving classifier performance over time is illustrated in Fig. 6.

5.2. Pets 2001

In our second experiment, we demonstrate our approach on an outdoor data set, *i.e.*, *PETS 2001*[5]. Compared to the previous experiment, this scenario is more challenging since the target objects are smaller and the background is characterized by higher variability. In particular, from several available sets, we chose *dataset 2*, which covers the scene from two different view-points.

The experiments were performed in the same way as for *PETS 2006* described in Section 5.1. The obtained results for $t = 0$, $t = 20$, and $t = 50$ are summarized in Table 2. The corresponding precision-recall curves are given in Fig. 7.

[5] `http://www.cvg.rdg.ac.uk/PETS2001`, (June 28, 2008)

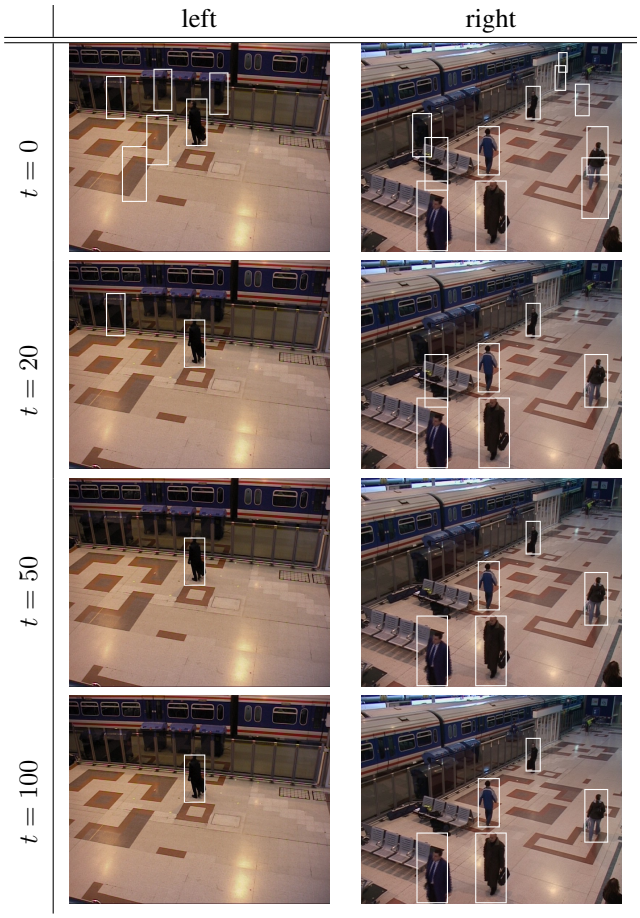


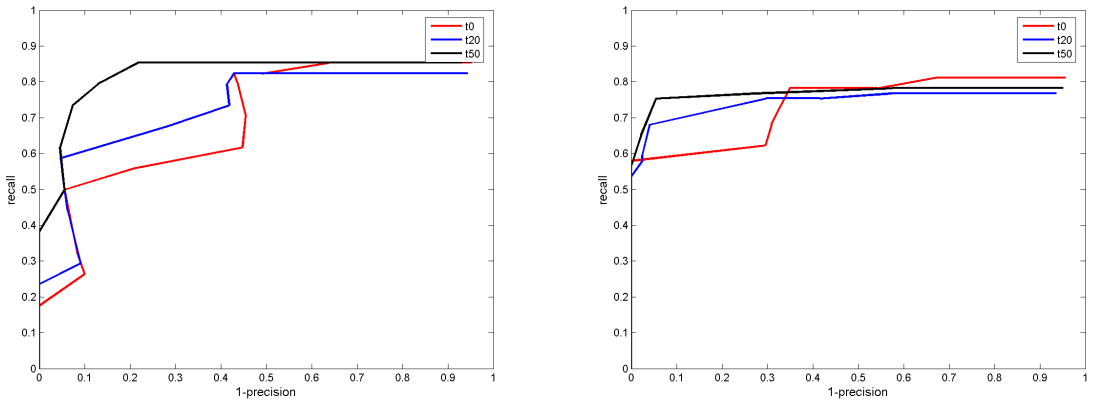
Fig. 6. *PETS 2006*: increasingly better detection results over co-training iterations.

Classifier 1			
t	recall	precision	F-measure
0	0.48	0.83	0.61
20	0.43	0.82	0.56
50	0.62	0.76	0.68

Classifier 2			
t	recall	precision	F-measure
0	0.56	0.69	0.62
20	0.44	0.47	0.45
50	0.69	0.79	0.73

Table 2. *PETS2001*: recall, precision, and F-measure for *classifier 1* and *classifier 2* after $t = 0$, $t = 10$ and $t = 50$ iterations.

Similar to the results for the *PETS 2006* data set it can be seen that the precision is increased. Additionally, for this specific data set also the recall was significantly increased. Since the scenario is more complex the pre-trained prior can not handle all variability of the positive samples. Hence, we



(a) classifier 1 (b) classifier 2

Fig. 5. *PETS 2006*: precision-recall-curves for *classifier 1* (a) and *classifier 2* (b).

additionally benefit from the proposed update strategy. But due to the higher complexity of the scenario the final overall performance is worse compared to the simpler *PETS 2006* discussed in the previous section. Again, finally, we show some illustrative results of the increasingly improving detectors in Fig. 8.

5.3. Detailed Performance Evaluation

Finally, we give some detailed performance measures on our prototype platform. For performance evaluation tests of our approach on the embedded platform we chose to set our focus on Idle/System and User CPU load as well as on free/buffered and cached amount of memory. Fig. 9 shows the breakdown of the CPU load during a five minutes object detection processing loop with two detection stops in-between. The performance test is done on a sequence of images, each with a resolution of 348×260. In Fig. 10 the memory load breakdown of the same sequence is shown. The two other figures (Fig. 11 and Fig. 12, respectively), picture the CPU and RAM load breakdown analogously for a 90 second lasting image sequence (with one detection stop at second 60).

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel approach for on-line learning classifiers in networks of distributed smart cameras. Our main contribution is to motivate and demonstrate that both, on-line learning (*i.e.*, on-line boosting for feature selection) and applying autonomously acting smart cameras, can benefit from each other. Contrary to previous methods, we do not rely on motion information and only have to label a few samples in the start-up phase of the system. Performance details of our approach were given on two challenging and publicly available standard data sets on pedestrian detection with

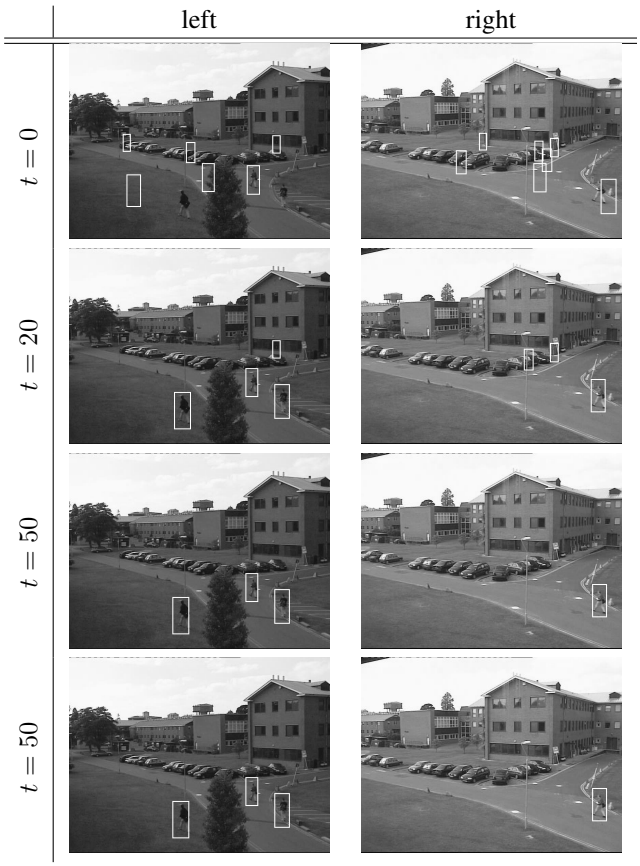
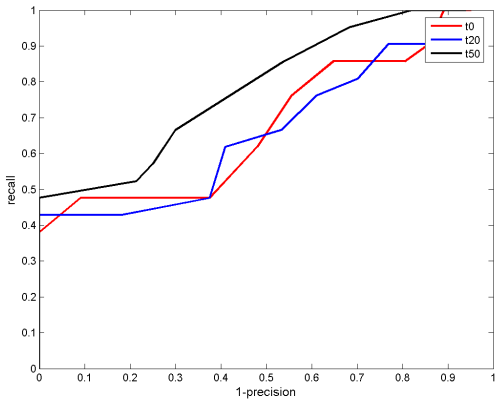
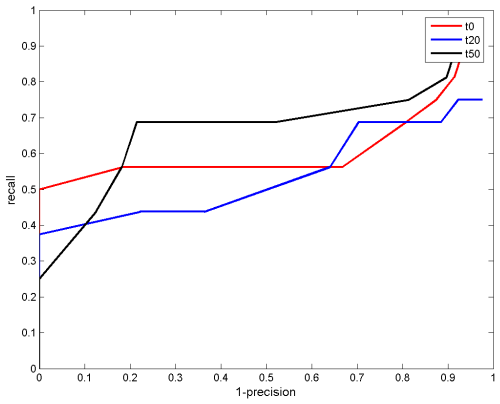


Fig. 8. *PETS 2001*: increasingly better detection results over co-training iterations..

wide-baseline views. Since we see our platform as a multi-sensor fusion system, in future work, we plan to integrate additional sensors (*e.g.*, laser and audio) in order to acquire further different "views" for our co-training approach.



(a) classifier 1



(b) classifier 2

Fig. 7. *PETS 2001*: precision-recall-curves for *classifier 1* (a) and *classifier 2* (b).

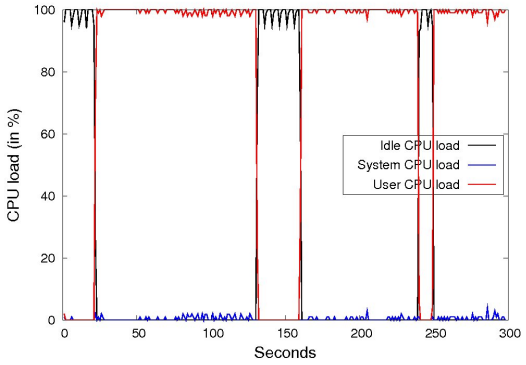


Fig. 9. CPU load breakdown (5 minutes)

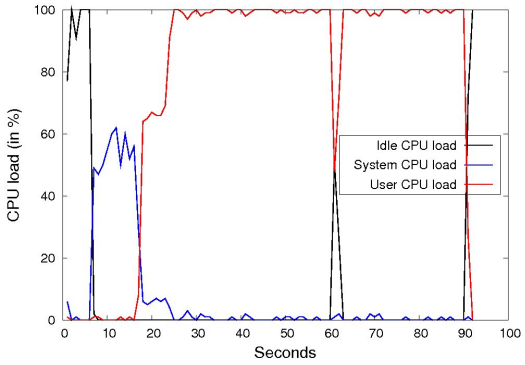


Fig. 11. CPU load breakdown (90 seconds)

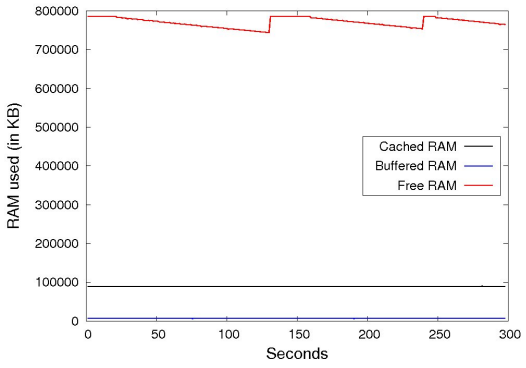


Fig. 10. Memory load breakdown (5 minutes)

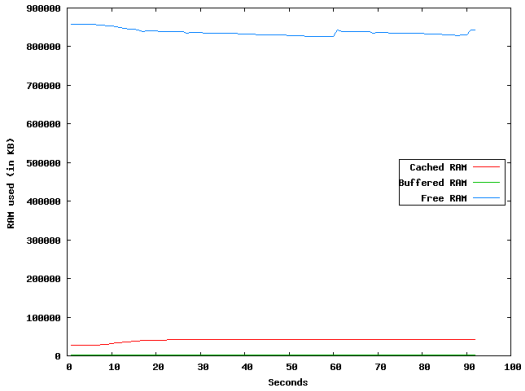


Fig. 12. Memory load breakdown (90 seconds)

7. REFERENCES

[1] W. Wolf, B. Ozer, and T. Lv, "Smart Cameras as Embedded Systems," *Computer*, vol. 35, no. 9, pp. 48–53, Sept. 2002.

[2] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, "Distributed Embedded Smart Cameras for Surveillance Applications," *Computer*, vol. 39, no. 2, pp. 68–75, Feb. 2006.

[3] B. Heyrman, M. Paindavoine, R. Schmit, L. Letellier, and T. Collette, "Smart camera design for intensive embedded computing," *Real-Time Imaging*, vol. 11, pp. 282–289, 2005.

[4] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin, "Camera Mote with a High-Performance Parallel Processor for Real-Time Frame-Based Video Processing," in *Proc. of the ACM/IEEE International Conf. on Distributed Smart Cameras (ICDSC 2007)*, Vienna, Austria, Sept. 2007, pp. 109–116.

[5] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," in *Proc. of the 6th International Symposium on Information Processing in Sensor Networks IPSN 2007*, 25–27 April 2007, pp. 360–369.

[6] T. E. Boult, R. Johnson, T. Pietre, R. Woodworth, and T. Zhang, "A Decade of Networked Intelligent Video Surveillance," in *ACM Workshop on Distributed Camera Systems*. 2006, ACM.

[7] C. Wu and H. Aghajan, "Model-based Human Posture Estimation for Gesture Analysis in an Opportunistic Fusion Smart Camera Network," in *Proc. of the IEEE International Conf. on Advanced Video and Signal-based Surveillance (AVSS 2007)*, London, UK, Sept. 2007.

[8] S. Fleck, R. Loy, C. Vollrath, F. Walter, and W. Strasser, "Smartclassysurv–A Smart Camera Network for Distributed Tracking and Activity Recognition and its Application to Assisted Living," in *Proc. of the ACM/IEEE International Conf. on Distributed Smart Cameras (ICDSC 2007)*, Vienna, Austria, Sept. 2007, pp. 109–116.

[9] B. Rinner and W. Wolf, "An Introduction to Distributed Smart Cameras," *Proc. of the IEEE*, vol. 96, no. 10, Oct. 2008, to appear.

[10] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, pp. 921–960, 2007.

[11] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, second edition, 2004.

[12] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multi-camera people tracking with a probabilistic occupancy map," *IEEE Trans. on PAMI*, vol. 30, no. 2, pp. 267–282, 2008.

[13] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proc. International Conf. on Computer Vision*, 2003, vol. 2, pp. 734–741.

[14] H. Grabner, P. M. Roth, and H. Bischof, "Is pedestrian detection really a hard task?," in *Proc. IEEE Intern. Workshop on Performance Evaluation of Tracking and Surveillance*, 2007, pp. 1–8.

[15] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT: Proc. Workshop on Computational Learning Theory*, 1998, pp. 92–100.

[16] P. Roth, H. Grabner, C. Leistner, M. Winter, and H. Bischof, "Interatctive learning a person detector: Fewer clicks - less frustration," in *Proc. Workshop of the Austrian Association for Pattern Recognition*, 2008.

[17] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2006, vol. I, pp. 260–267.

[18] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, no. 2/3, pp. 103–134, 2000.

[19] M.-F. Balcan, A.Blum, and K. Yang, "Co-training and expansion: Towards bridging theory and practice," in *Advances Neural Information Processing Systems*. MIT Press, 2004.

[20] A. Levin, P. Viola, and Y. Freund, "Unsupervised improvement of visual detectors using co-training," in *Proc. International Conf. on Computer Vision*, 2003, vol. 2, pp. 626–633.

[21] O. Javed, S. Ali, and M. Shah, "Online detection and classification of moving objects using progressively improving detectors," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005, vol. I, pp. 696–701.

[22] Q. Zhu, S. Avidan, and K.-T. Cheng, "Learning a sparse, corner-based representation for background modelling," in *Proc. IEEE Intern. Conf. on Computer Vision*, 2005, vol. I, pp. 678–685.

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

[24] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[25] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[26] K. Tieu and P. Viola, "Boosting image retrieval," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2000, pp. 228–235.

[27] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2001, vol. I, pp. 511–518.

[28] 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.

[29] R. Pflugfelder and H. Bischof, "Online auto-calibration in man-made worlds," in *Proc. Digital Image Computing: Technqiues and Applications*, 2005, pp. 519–526.

[30] S. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Proc. European Conf. on Computer Vision*, 2006, vol. IV, pp. 133–146.

[31] C. Arth, H. Bischof, and C. Leistner, "Tricam - an embedded platform for remote traffic surveillance," in *Proc. CVPR Workshop on Embedded Computer Vision*, 2006.

[32] R. Lienhart and J. Maydt, "An extended set of haar-like features for object detection," in *Proc. International Conf. on Image Processing*, 2002, pp. 900–903.

[33] A. Klausner, A. Tengg, and B. Rinner, "Distributed multi-level Data Fusion for Networked Embedded Systems," *IEEE Journal on Selected Topics in Signal Processing*, vol. 2, 2008, to appear.