On-line Learning of Unknown Hand Held Objects via Tracking

Peter M. Roth Michael Donoser Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology
Inffeldgasse 16/II, 8010 Graz, Austria
{pmroth,donoser,bischof}@icg.tu-graz.ac.at

Abstract

For many computer vision applications labeled/segmented data is needed. Manually assigning labels or segmenting images is a time consuming and tedious task and becomes infeasible for a huge amount of data (e.g., when analyzing a video stream). Thus, this paper proposes a new approach to minimize the manual labeling/segmentation effort for learning an object detector by automatically extracting training data directly from a video sequence. Therefore, a robust background model, a tracker and an on-line learning method are combined. The main idea is to track an object through a video sequence and to directly use the obtained image patches, showing the object from different views, to incrementally update an existing model which in turn can be used for detection. As the tracker is initialized automatically by change detection, no user interaction is needed! Thus, an unknown object can be learned without having any prior information. To show the benefit of the proposed approach the framework is demonstrated on several typical objects that can be found on a desktop.

1 Introduction

Given an input image, the task of object detection is to decide if a cropped patch of an image contains a specific object or not. Thus, a detector based on a certain classifier (e.g., AdaBoost [33], Neural Network [28], Winnow [15] or PCA [11]) has to be trained. However, for training a classifier a set of labeled data (positive and negative examples) is needed. A lot of research has been focused on efficient (unsupervised) training of detectors but only little attention has been paid to efficiently labeling and acquiring suitable training data. Thus, training examples are usually obtained by hand labeling. As a huge amount of data is required, i.e., a higher number of objects should be learned under different view points and lightening conditions, this is not very convenient. Therefore, the hand labeling effort should be minimized or even completely avoided.

Negative examples (i.e., examples of images not containing the object) are usually obtained by a bootstrap approach [31]. Starting with a few negative examples a classifier is trained. The obtained classifier is applied to images that do not even contain 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. Therefore, obtaining negative examples is usually not much of a problem.

Automatically obtaining a set of positive examples is a more difficult task. Existing approaches to minimize the labeling effort (e.g., [14, 22, 27]) are based on two stages. In the first stage, an initial classifier is trained using a smaller number of examples. In the second stage, the classifier is applied on a training sequence and the detected patches are added to the set of positive examples. Levin et al. [14] start with a small set of hand labeled data and generate additional labeled examples by applying co-training of two classifiers. To completely avoid hand labeling Nair and Clark [22] propose to use motion detection to obtain the initial training set, an idea that was previously addressed by Baumberg and Hogg [2] to minimize the manual effort of learning the shape model of persons from a video stream. New examples are acquired by applying a detector obtained by on-line learning (Winnow). To get an even better initial discriminant classifier Roth et al. [27] proposed to apply a generative model on the motion data to robustly extract the true positives only. A new positive example is added if it fits to both, the discriminant and the generative model; in the same way negative examples can be obtained. As a disadvantage of these methods an initial classifier has to be trained and therefore the algorithms are biased by

the examples used to train the initial classifiers. Thus, a great number of potential new examples, even different views of the same object, would not be added because they do not fit to the current model!

Other approaches use "weakly labeled data" (e.g., [26]) or apply Active Learning (e.g., [32, 34]). Where for learning from weakly labeled data some information (e.g., the object-of-interest is known to be in the training images) has been provided, Active Learning can be used to label both, negative and positive samples. However, still a (small) set of labeled samples reliably representing the classes is needed in the beginning.

To overcome the drawbacks described above a tracker can be applied. Based on an one-time initialization of the object-of-interest, a robust tracker is able to follow the object through the entire video sequence – assuming that the frame-rate is high enough such that the location of the object is only slightly changed from frame to frame. If tracking works on a longer video sequence, different views of the same object can be obtained. Thus, new positive examples are obtained without any user interaction needed. The extracted image patches can be directly used as input for an (incremental) learning algorithm. Moreover, by using change detection information the tracker can be initialized automatically. In this paper we apply the Approximate Median [19] filter to model the background, an MSER tracker [6] and incremental PCA [30] for learning. But as the approach is quite general other methods may be applied. Thus, we have a framework to learn even unknown objects directly from a video sequence.

Sivic et al. [29] have proposed a similar approach (apply tracking for data acquisition). But there are two main differences: (1) local features are used for learning and (2) as a face detector based on AdaBoost is applied to obtain the first examples, the framework can not be used for learning an object representation from scratch.

The outline of the paper is as follows: In Section 2 we introduce the new learning framework and describe the components used. To show the power of the approach the framework is demonstrated on different hand held objects in Section 3. Finally, conclusions are drawn in Section 4.

2 Learning Framework

The proposed learning framework, consisting of three components, is depicted in Figure 1. First, an adaptive background model is estimated using the Approximate Median algorithm [19], a simple but fast and effective background modeling approach. If a consistent background model is obtained, the object of interest is presented to the system. For change detection a binary image is computed by pixel-wise thresholding the difference image between the current frame and the background model. Second, if an obtained blob is stable over time, the corresponding patch in the current frame is used to initialize the tracker. In particular we apply a tracker that is based on the MSER detector [6], but in general any other blob-based tracker (e.g., mean shift color tracker [4]) may be used. In fact, compared to change detection, using a tracker produces more accurate masks, i.e., for hand held objects the hand is segmented as well (see Figure 2(a-b)). Finally, the patches obtained by the tracker are directly used for on-line learning. Due to the expected redundancy in the data it is more convenient to use a generative method such as PCA. Moreover, the patches obtained by tracking are only slightly changing from frame to frame. Therefore, they are highly appropriate for on-line learning. Thus, we apply an incremental PCA algorithm [30]. But as it was proposed for Conservative Learning [27] a discriminative classifier like AdaBoost [33] can be trained from (a subset) of the thus obtained data later on!

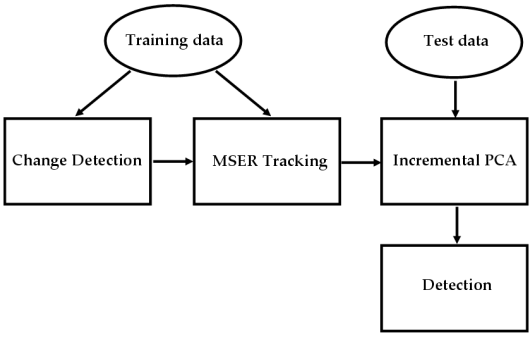


Figure 1. The proposed learning framework.

2.1 Change Detection

Having a stationary camera a common approach to detect forground 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

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

To estimate the background model $\mathbf{B}_t$ several (adaptive) approaches including running average [12], tem-

poral median filter [16] or eigenbackgrounds [25] have been proposed. A simple but computationally efficient method was developed by McFarlane and Schofield [19]. The Approximated Median filter computes an approximation of the temporal median by incrementing the current estimate (only one reference image has to be stored!) by one if the input pixel value is larger than the estimate and by decreasing it by one if smaller:

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

This estimate eventually converges to the real median. To avoid that non-moving forground objects are accumulated into the background model and to preserve a robust median estimation spatial and temporal weights may be assigned for the updates. Having a consistent background model, a tracker can be robustly initialized if a detected foreground blob is stable over time.

To demonstrate the initialization process Figure 2 shows three subsequent frames (input image and binary foreground image) of this first stage of our framework. After a consistent background model was computed the object is put into the scene (Figure 2 (a)-(b)). If the obtained blob is stable over time the tracker is initialized with the corresponding region of the input image (Figure 2 (c)).

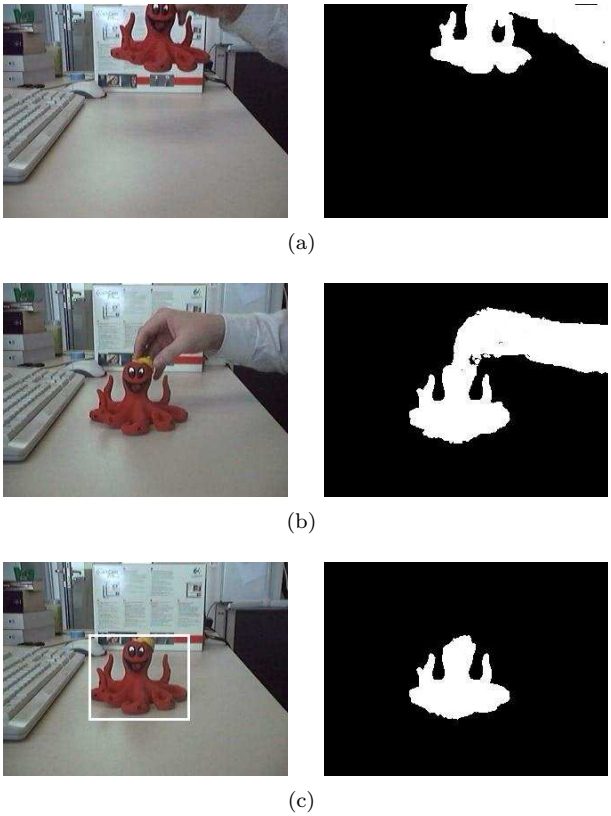


Figure 2. Initialization of the tracker by change detection.

2.2 MSER-Tracker

For tracking the objects we apply a tracker that is based on the detection of Maximally Stable Extremal Regions (MSERs) that was proposed by Donoser and Bischof [6]. Thus, this section basically summarizes the main concept of MSER-tracking. In addition, the cropping of patches for learning based on the tracking results is explained.

The Maximally Stable Extremal Region (MSER) detector from Matas et al. [17] has proven to be one of the best interest point detectors in computer vision. Evaluations by Mikolajczyk and Schmid [20] and Fraundorfer and Bischof [8] revealed that the MSER detector performs best on a wide range of test sequences.

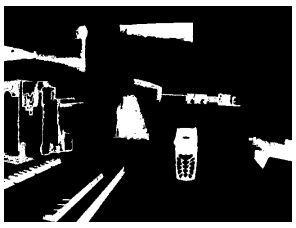
MSERs are connected regions which can be detected in any image whose pixel values are of a totally ordered set. All MSERs are defined by an extremal property of the intensity function in the region and on its outer boundary. MSERs have properties that form their superior performance as stable local detector. The set of MSERs is closed under continuous geometric transformations and is invariant to affine intensity changes. Furthermore MSERs are detected at different scales.

In the paper from Matas et al. [17] MSER detection is applied to single gray scale images. But the concept can be easily extended to color images by defining a suitable ordering relationship on the color pixels. In our case, we fit a multivariate Gaussian distribution to the RGB values of the region-of-interest initialized by the change detection algorithm. Then, the RGB values of all image pixels are ordered by their Mahalanobis distance [7] to this Gaussian distribution. The calculated distances can be visualized as gray scale image, which is shown in Figure 3(b), wherein dark areas represent small distances, whereas bright areas represent pixels with very differing color.

The detection of MSERs can be implemented in an efficient way by analysis of a data structure called the component tree that recently has been used by Couprie et al. [5] for efficient implementation of watershed segmentation. The component tree is a rooted, connected tree and can be created for any image with pixel values that are part of a totally ordered set. Each node of the component tree represents a connected region within the input image I_{in}. For MSERs we only consider extremal regions R_i as nodes, which are defined by

(a) Input image.

(b) Color probability image.



(c) MSER detection result.

Figure 3. Illustration of MSER detection on color images.

$$\forall p \in R_i \, , \forall q \in boundary(R_i) \rightarrow I_{in}(p) \geq I_{in}(q). \quad (3)$$

These extremal regions are identified as connected regions within binary threshold images I^k_{bin}, which are the result of the calculation

$$I^k_{bin} = I_{in} \geq k, \quad (4)$$

where $k \in [min(I_{in})\, max(I_{in})]$. Each node of the component tree is assigned the corresponding gray value k at which it was determined.

The edges within the tree define an inclusion relationship between the extremal regions. Thus, for a region R_i that is the son of a region R_j within the tree,

$$\forall p \in R_i \rightarrow p \in R_j \quad (5)$$

is fulfilled. By moving in the component tree upwards, the corresponding gray value k of the extremal regions becomes lower, which leads to increased region sizes. The root of the tree represents a region which includes all pixels of the input image I_{in}.

MSERs are identified by analysis of the component tree. For each connected region R_i within the tree a stability value Ψ is calculated.

$$\Psi(R_i) \; = (|R^{g-\Delta}_j| \; - |R^{g+\Delta}_k|)/|R^g_i|, \quad (6)$$

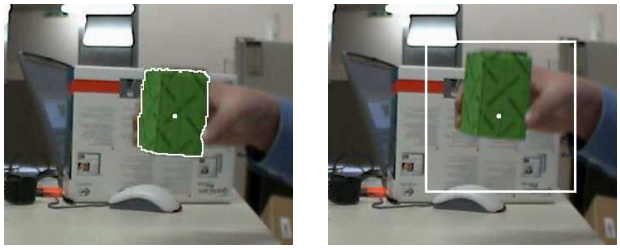
where $|.|$ denotes the cardinality, R^g_i is a region which is obtained by thresholding at a gray value g and Δ is a stability range parameter. $R^{g-\Delta}_j$ and $R^{g+\Delta}_k$ are the extremal regions that are obtained by moving upwards respectively down-wards in the component tree from region R_i until a region with gray value $g-\Delta$ respectively $g+\Delta$ is found. MSERs correspond to those nodes of the tree that have a stability value Ψ, which is a local minimum along the path to the root of the tree. Thus, maximally stable regions are those regions which have approximately the same region size within 2Δ neighboring threshold images. Figure 3(c) shows an exemplary MSER detection result. The detected extremal regions are shown in white.

Various algorithms have been proposed to compute the component tree, the most efficient algorithm by Najman and Couprie [23] was used for our implementation. The complexity of the creation process for the component tree becomes $O(N\alpha(N))$, where $N = n+m$, n is the number of pixels and m is the number of arcs in the image (i.e., approximately $2n$ for the 4-neighborhood). The function α is the inverse Ackermann function [1], which is a very slowly growing function, that is for all practical purposes below 4. Thus, analysis of the component tree enables the detection of MSERs in quasi-linear time, which improves the original implementation that runs in $O(N \log\log N)$ time – of course for practical image sizes there is not much difference.

The component tree allows not only the detection of MSERs within a single image but in addition it constitutes the basis for the extension to MSER tracking [6]. This algorithm improves the computational time for MSER calculation and additionally provides more stable results compared to single frame based MSER detection. In the following, the main ideas of the algorithm are summarized:

First, the tracker is initialized by selecting a region-of-interest (ROI) within the image I_t at time t. To have a full automatic system, where no user interaction is needed, the this selection is performed based on the results of the change detection algorithm. Next, MSER detection is performed by analyzing the component tree of the selected ROI. The biggest, detected MSER is tracked at the image I_{t+1} by performing two steps. (1) A new region of interest (ROI) of predefined size, centered around the center of mass of the MSER to be tracked, is propagated to the next frame (if a motion model is available it can be incorporated here). Then, the component tree for this region is built in quasi-linear time by the previously described algorithm. (2) The entire tree is analyzed and the node which best fits to the input MSER is chosen as the tracked extremal region representation. Thus, not necessarily the most stable extremal region is chosen as tracked region representation. Figure 4 illustrates this

tracking concept.



(a) Input image t. The MSER to be tracked is shown with white border.

(b) Region of interest (ROI) around center of mass in image $t+1$.

Figure 4. ROI definition for tracking of single MSER.

The best fit to the input MSER is identified by comparing incrementally computed feature vectors that are built for each of the connected regions of the component tree. The region, which has the smallest weighted Euclidean distance between its feature vector and the one from the input MSER, is chosen as tracked representation.

The features calculated are mean gray value, region size, center of mass, width and height of the bounding box and stability. The weights for the features can be used to adapt to different kinds of input data. Please note, that due to efficiency reasons all of the features of the extremal regions are computed incrementally [18] during creation of the component tree. Thus, no additional computation time is required. The update takes place each time connected components are united by the union step.

Finally, the required image patch is created by cropping the bounding box of the detected, tracked MSER representation from the image I_{t+1}. The presented steps are repeated again and again, which creates a set of image patches, showing the object-of-interest from different viewpoints. Figure 5 illustrates the MSER tracking concept for image patch creation.

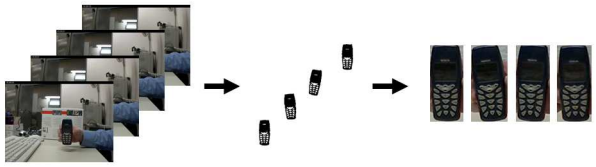
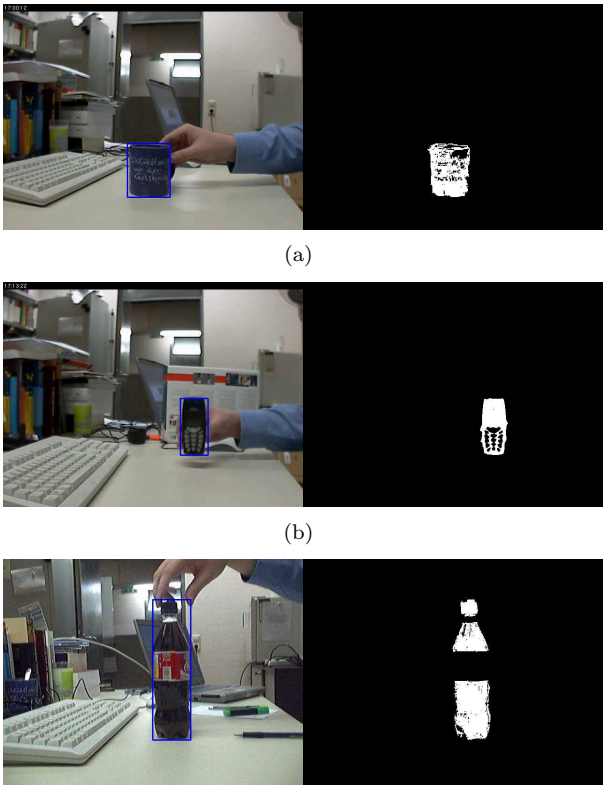


Figure 5. Concept of MSER tracking for patch extraction.

Examples of tracked objects and its corresponding MSERs are depicted in Figure 6. As can be seen the MSERs may contain holes (texture, smaller regions of different color, etc.). But since the patches are cropped based on the corresponding bounding boxes this is not a problem at all. Moreover, objects may be represented by more than one MSER (Figure 6(c)). Thus, these regions are tracked in parallel and a new training patch is obtained by cropping the bounding box covering all of the MSERs. Examples of cropped patches that are used for learning are shown in Figure 7.



(a)

(b)

(c)

Figure 6. Detected MSERs during tracking.

2.3 Incremental PCA

The image patches obtained by tracking are highly correlated and need to be compressed which can be efficiently performed by PCA. The basic idea of PCA is to map high-dimensional input data to a low-dimensional subspace by finding the directions with the highest variance. Hotelling [11] has shown that the principal axes maximizing the variance are given by the eigenvectors of the covariance-matrix of the data.

Let $\mathbf{x}_i \in \mathbb{R}^m$ be an individual image represented as a vector, and $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ a set of images

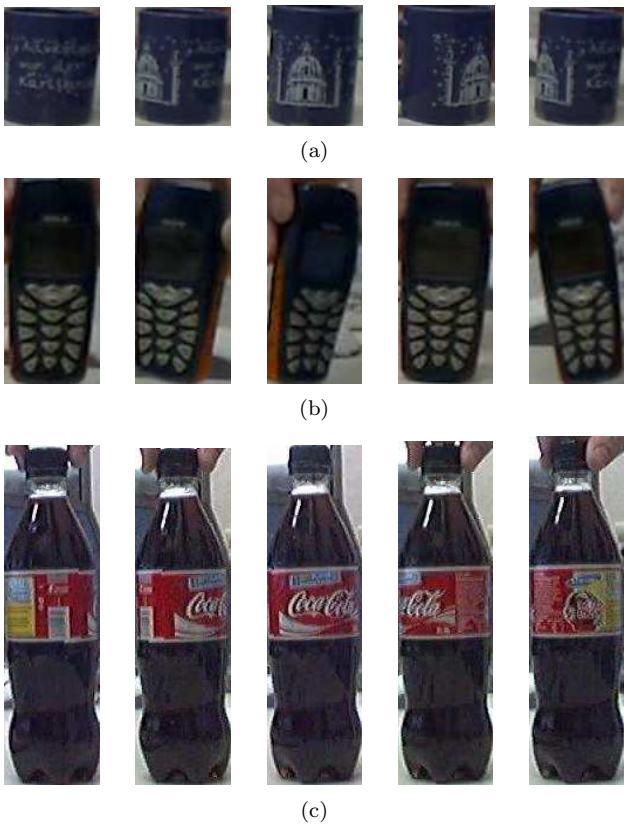


Figure 7. Tracking results for training data sets.

assuming that $\mathbf{X}$ is mean normalized. Let $\mathbf{Q}$ be the covariance matrix of $\mathbf{X}$. Due to the properties of the covariance-matrix (symmetric and pos. semi-definite), SVD on $\mathbf{X}$ can be applied to efficiently compute the eigenvectors $\mathbf{u}_i \in \mathbb{R}^m$ of $\mathbf{Q}$; $\mathbf{U} = [\mathbf{u}_1, \dots \mathbf{u}_n] \in \mathbb{R}^{m \times n}$.

As most information is covered within the eigenvectors according to the largest eigenvalues, usually only k ($k < n$) eigenvectors are needed to represent an image $\mathbf{x}$ to a sufficient degree of accuracy:

$$\tilde{\mathbf{x}} = \sum_{i=1}^{k} a_i(\mathbf{x})\mathbf{u}_i = \mathbf{U}\mathbf{a} \qquad (7)$$

In the recognition stage, an unknown test sample $\mathbf{y}$ is projected to the eigenspace encompassing the training images by calculating a standard projection

$$a_i(\mathbf{y}) = \mathbf{u}_i^T \mathbf{y} = \sum_{j=1}^{m} u_{ji} y_j \ , \ \ i = 1 \dots k \ , \qquad (8)$$

or, as a robust procedure [13], by solving a system of linear equations

$$y_{r_i} = \sum_{j=1}^{k} a_j(\mathbf{y}) u_{r_i,j} \ , \ \ i = 1 \dots q \ , \qquad (9)$$

evaluated at $q \geq k$ points $\mathbf{s} = (s_1, \dots s_q)$. Once we have obtained the parameters $a_i(\mathbf{y})$ we can reconstruct the image using (7) and determine the reconstruction error $\epsilon = ||\mathbf{y} - \tilde{\mathbf{y}}||$. The detection may either be performed based on this error [13] or by estimating the distance between the projected test sample and the projected training data [24].

The most efficient approach is to use an on-line PCA algorithm that is trained as new data arrives. Therefore, different incremental PCA approaches have been proposed that are based on incremental SVD-updating (e.g., [3, 21, 30]). Since the method can be extended in a robust way, i.e., corrupted input images (containing occlusions or unreliable pixel values) may be used for incrementally updating the current model, we apply a simplified version of the incremental PCA method of Skočaj and Leonardis [30].

Assuming that an eigenspace was already built from n images, at step $n+1$ the current eigenspace is updated with a new image $\mathbf{x}$. First, $\mathbf{x}$ is projected into the current eigenspace $\mathbf{U}^{(n)}$ and the image is reconstructed. Next, the difference $\mathbf{r}$ (residual vector) between the original image and its reconstruction is estimated. A new basis $\mathbf{U}'$ is created by enlarging the current basis $\mathbf{U}^{(n)}$ by the residual vector $\mathbf{r}$. Since $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$ the current images as well as the new image can be represented in the new basis. Finally, the new subspace $\mathbf{U}^{(n+1)}$ is estimated by rotating the subspace $\mathbf{U}'$ by $\mathbf{U}''$, where $\mathbf{U}''$ is obtained by performing PCA on the new representation. 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 [10].

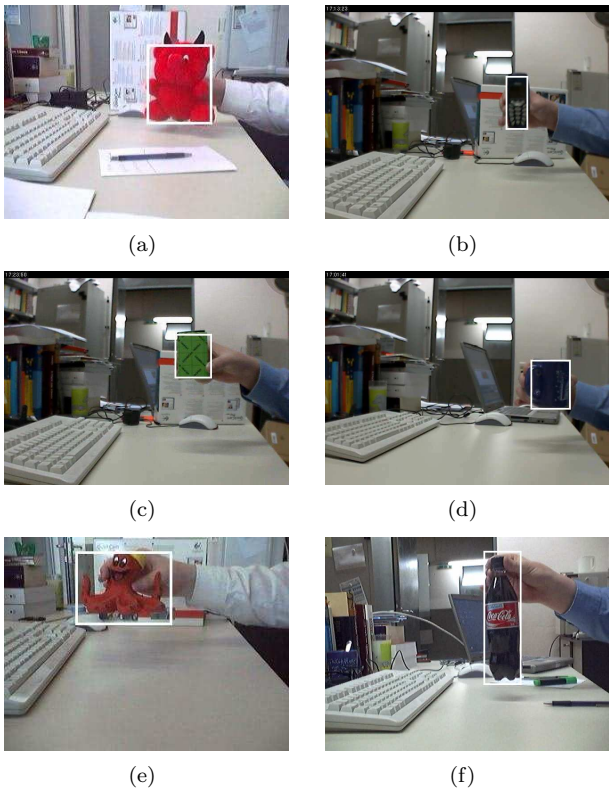
To obtain an initial model, the batch method may be applied on a smaller set of training images. Alternatively, to have a fully incremental algorithm, the eigenspace may be initialized using the first training image $\mathbf{x}$: $\boldsymbol{\mu}^{(1)} = \mathbf{x}$, $\mathbf{U}^{(1)} = \mathbf{0}$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

3 Experimental Results

To demonstrate the approach we have trained and evaluated detectors for several objects that can be found on a desktop. The objects are hand held in the training stage as well as in the evaluation stage. We have created separate sequences to train the classifiers and independent test sequences for evaluation. To learn an object representation different views of the

target object were presented. Patches of different size were resized to a common patch size according to the initialization of the tracker.

The experiments are divided into three parts: First, a background model is estimated and the tracker is initialized. Second, the object is tracked and an object representation is incrementally learned. Finally, the previously computed classifier is applied for a detection task on an independent test sequence. For the results shown in Figure 8 and Figure 9 for each object 200-300 patches obtained by tracking were used for on-line learning; to build a PCA model of sufficient accuracy only 10-15 eigenvectors were needed. Thus, Figure 8 depicts detection results on the test sequences for different data sets. For Figure 9 a more complex test sequence was created. Three different objects were presented at different locations and varying scales. However, all objects were detected!

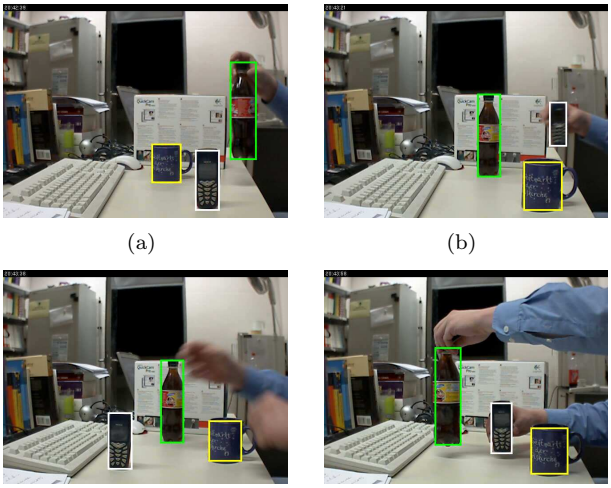


(a) (b) (c) (d) (e) (f)

Figure 8. Detections: simple test sequences.

4 Conclusion

We have presented a framework for on-line learning an object representation even for unknown objects directly from a video sequence. Therefore, a background model (Approximated Median), a tracker (MSER tracker) and an on-line learning method (incremental PCA) were combined. The tracker is initialized by change detection and the patches obtained are used as input for the incremental learner. Thus, no user interaction is needed! Moreover, no labeled or segmented images are required. We have demonstrated the framework on different desktop objects, but as our approach is quite general we could learn any other object (e.g., faces) if an (automatic) initialization of the tracker is available. The presented approach can simply be extended by adding robustness in the detection [13] as well as in the learning stage [30]. Furthermore, a selected subset of patches (avoiding redundancy in the data!) obtained by the tracker may be used as positive examples for conservative learning [27] or more general for on-line boosting [9].



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

Figure 9. Detections: complex test sequence.

Acknowledgment

This work has been supported by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04 and the EU FP6-507752 NoE MUSCLE IST project. Part of this work has also been carried out within the K-plus Competence center Advanced Computer Vision funded under the K plus program.

References

[1] W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99:118–133, 1928.

[2] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *Proc. European Conf. on Computer Vision*, volume I, pages 299–308, 1994.

[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.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 142–149, 2000.

[5] M. Couprie, L. Najman, and G. Bertrand. Quasi-linear algorithms for the topological watershed. *Journal of Mathematical Imaging and Vision*, 22(2-3):231–249, 2005.

[6] M. Donoser and H. Bischof. Efficient maximally stable extremal region (MSER) tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2006. to be published.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2000.

[8] F. Fraundorfer and H. Bischof. A novel performance evaluation method of local detectors on non-planar scenes. In *Workshop Proc. Empirical Evaluation Methods in Computer Vision (CVPR)*, 2005.

[9] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2006. to be published.

[10] 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.

[11] H. Hotelling. Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[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*, 1994.

[13] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, 2000.

[14] 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.

[15] N. Littlestone. Learning quickly when irrelevant attributes abound. *Machine Learning*, 2:285–318, 1987.

[16] 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.

[17] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conf.*, pages 384–393, 2002.

[18] J. Matas and K. Zimmermann. A new class of learnable detectors for categorisation. In *Proc. Scandinavian Conf. on Image Analysis*, pages 541–550, 2005.

[19] N. J. McFarlane and C. P. Schofield. Segmentation and tracking of piglets. *Machine Vision and Applications*, 8(3):187–193, 1995.

[20] K. Mikolajczyk and C. Schmid. Comparison of affine-invariant local detectors and descriptors. In *Proc. European Signal Processing Conf.*, 2004.

[21] H. Murakami and V. Kumar. Efficient calculation of primary images from a set of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 4(5):511–515, 1982.

[22] 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.

[23] L. Najman and M. Couprie. Quasi-linear algorithm for the component tree. In *SPIE Vision Geometry XII*, volume 5300, pages 98–107, 2004.

[24] S. K. Nayar, H. Murase, and S. A. Nene. Parametric appearance representation. In *Early Visual Learning*, pages 131–160.

[25] N. M. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[26] C. Rosenberg and M. Hebert. Training object detection models with weakly labeled data. In *Proc. British Machine Vision Conf.*, pages 577–586, 2002.

[27] P. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proc. IEEE Workshop on VS-PETS*, pages 223–230, 2005.

[28] 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.

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

[30] 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.

[31] 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.

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

[33] 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.

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