Computer Vision Winter Workshop 2006, Ondřej Chum, Vojtěch Franc (eds.)
Telč, Czech Republic, February 6–8
Czech Pattern Recognition Society

Tracking for Learning an Object Representation from Unlabeled Data

Peter M. Roth, Michael Donoser, and Horst Bischof

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

Abstract *For learning an object representation a huge amount of labeled data is needed. To minimize the labeling effort this paper proposes a new approach for learning from unlabeled data. The main idea is to combine a tracker and a learning method by directly feeding the learning algorithm with patches obtained by the tracker. In particular we apply an MSER based tracker and batch PCA for learning. But in general any tracker and any learning algorithm may be used. In a first step, the object-of-interest is initialized manually. Then, the object is tracked through a video sequence and a set of image patches, showing the object from different views, is extracted. The obtained patches are passed to PCA and a reconstructive model of the object is learned. Human input is reduced to a one-time initialization of the tracker. The approach is demonstrated on realistic scenes including face detection and detection of hand held objects.*

1 Introduction

In the past, numerous successful approaches for object detection including faces [23], pedestrians [27], cars [2], bikes [18], etc. have been proposed. Given an input image, the task is to decide if a cropped patch contains a specific object or not. Therefore a detector based on a classifier (e.g., AdaBoost [7], Winnow [11], Neural Network [20] or support vector machine [25]) has to be trained. However, for training a classifier a set of labeled data (positive and negative examples) is needed. Much research has been focused on efficient training of detectors but only little attention has been paid for labeling and acquiring suitable data. Therefore, training data is usually obtained by hand labeling a large number of images. As a huge amount of data is required, this is not very convenient. Thus, to keep the learning process as automatic as possible, the hand labeling effort should be minimized.

Negative examples (i.e., examples of images not containing the object) are usually obtained by a bootstrap approach [22]. Starting with a few negative examples a classifier is trained. 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. Therefore, obtaining negative examples is usually not much of a problem.

Automatically obtaining a set of positive examples is a more difficult task. Existing approaches to minimize the labeling effort (e.g., [10, 15, 19]) are based on two stages. In the first stage, an initial classifier is trained using a smaller number of examples; in the second stage, the classifier is applied on a training sequence and the detected patches are added to the set of positive examples. Levin et al. [10] start with a small set of hand labeled data and generate additional labeled examples by applying co-training of two classifiers. To completely avoid hand labeling Nair and Clark [15] propose to use motion detection to obtain the initial training set. New examples are acquired by applying a detector obtained by on-line learning (Winnow). Similar to [15] for Conservative Learning [19] proposed by Roth et al. motion information is used to create an initial data set. But as motion detection may return false positives a generative model is applied to robustly extract the true positives only. Thus, an even better initial discriminant classifier is obtained. A new positive example is added if it fits to both, the discriminant and the generative model. As a main disadvantage of these methods an initial classifier has to be trained. Therefore the algorithms are biased by the examples used to train the initial classifiers. Thus, different views of the same object that do not fit to the current model would not be used for updates. Additionally, (manually) labeled data is necessary to train the classifier.

Automatically labeling both, negative and positive samples, can be achieved by applying Active Learning (e.g., [24, 28]). Given an active learner $l(f, s, D)$, where f is a classifier trained on the labeled data set D, the sample function s automatically selects those unlabeled samples that provide most additional information for explicit labeling. However, still a 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 a manual, 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 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 may be achieved. Thus, new positive examples are obtained without any further user interaction.

The extracted image patches can be directly used as input for a learning algorithm. Therefore, the main contribution of

this paper is to propose a framework that combines a tracker and a learning algorithm. Thus, the human input and labeling effort can be reduced to manually selecting the object-of-interest once in the beginning. The proposed framework is depicted in Figure 1.

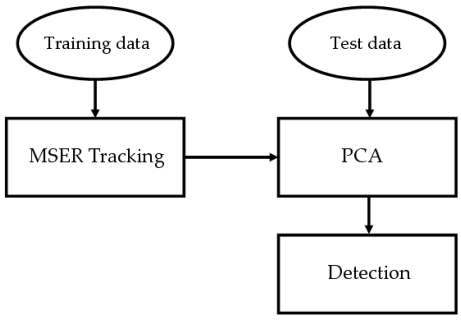


Figure 1: The proposed learning framework.

In particular, we apply a tracker that is based on the MSER detector [12] and batch PCA [8] for learning. But since the approach is very general any other tracker (e.g., Mean shift color tracker [3], etc.) and any other learning algorithm (e.g., AdaBoost [26]) may be applied. But due to the expected redundancy of the data obtained by the tracker a generative method such as PCA would be more appropriate. To show the power and the generality of the approach the framework is demonstrated on two different tasks: (1) face detection and (2) detection of hand held objects.

The outline of the paper is as follows: In Section 2 we describe the tracking algorithm used and summarize the applied learning method. Experimental results of learning faces and different desktop objects are presented in Section 3 and finally, conclusions are drawn in Section 4.

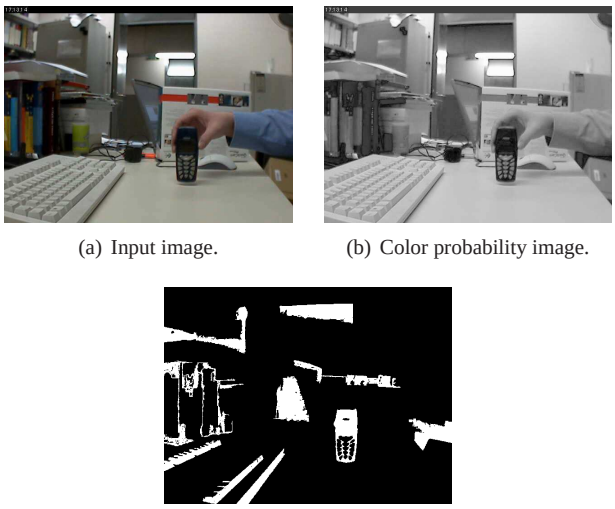
2 Tracking and Learning

2.1 Tracking

The first part of our concept requires a stable tracking method for video sequence analysis. We introduce a novel algorithm for tracking of Maximally Stable Extremal Regions (MSERs). The presented algorithm improves the computational time for MSER calculation and additionally provides more stable results compared to single frame based MSER detection. The object-of-interest to be tracked is initialized by hand and MSER tracking detects a set of image patches containing the object-of-interest in each frame of the video sequence.

The Maximally Stable Extremal Region (MSER) detector from Matas et al. [12] has proven to be one of the best interest point detectors in computer vision. Evaluations by Mikolajczyk and Schmid [14] and Fraundorfer and Bischof [6] 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.



(a) Input image. (b) Color probability image.

(c) MSER detection result.

Figure 2: Illustration of MSER detection on color images.

In the main paper from Matas et al. [12] 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 pixels from the manually initialized object-of-interest. Then, the RGB values of all image pixels are ordered by their Mahalanobis distance [5] to this Gaussian distribution. The calculated distances can be visualized as gray scale image, which is shown in Figure 2(b), wherein dark areas represent small distances, whereas bright areas represent pixels with very differing color.

MSER tracking is implemented in the most efficient way by analysis of a data structure called the component tree. The component tree is a structure which allows the detection of MSERs within a single image and, in addition, constitutes the basis for the extension to MSER tracking. The component tree has been recently used by Couprie et al. [4] 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

$$\forall p \in R_i \, , \forall q \in boundary(R_i) \rightarrow I_{in}(p) \geq I_{in}(q). \quad (1)$$

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

$$I^k_{bin} = I_{in} \geq k, \quad (2)$$

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,

$$\forall p \in R_i \rightarrow p \in R_j \tag{3}$$

is fulfilled. By moving in the component tree up-wards, 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}.

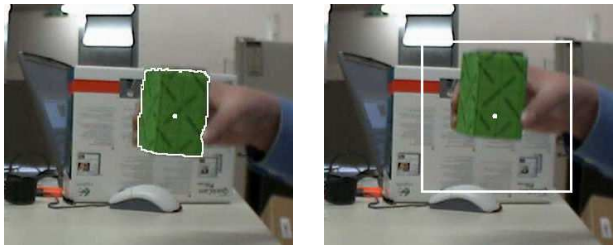
MSERs are identified by analysis of the component tree. For each connected region R_i within the tree a stability value Ψ is calculated.

$$\Psi(R_i) \ = (|R_j^{g-\Delta}| \ - |R_k^{g+\Delta}|)/|R_i^g|, \tag{4}$$

where $|.|$ denotes the cardinality, R_i^g is a region which is obtained by thresholding at a gray value g and Δ is a stability range parameter. $R_j^{g-\Delta}$ and $R_k^{g+\Delta}$ are the extremal regions that are obtained by moving up-wards 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 2(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 [16] was used for our implementation. The complexity of the creation process for the component tree becomes $\mathrm{O}(N\alpha(N))$, 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 $\mathrm{O}(N \log\log N)$ time – of course for practical image sizes there is not much difference.

MSER tracking starts with the manual initialization of a region-of-interest (ROI) within the image I_t at time t. Then, MSER detection by analysis of the component tree is applied to this ROI. The biggest, detected MSER is tracked at the image I_{t+1} by performing the two following steps. First, 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. Second, 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 3 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 3: ROI definition for tracking of single MSER.

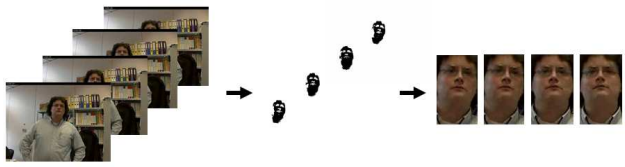


Figure 4: Concept of MSER tracking for patch extraction.

The best fit to the input MSER is identified by comparing 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 [13] 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 4 illustrates the MSER tracking concept for image patch creation.

2.2 Learning

For the second part of our concept we apply batch PCA [8] to learn the object representation from the patches obtained by tracking.

As images can be considered high dimensional vectors, let us introduce the notation: 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 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 $\mathbf{Q}$; $\mathbf{U} = [\mathbf{u}_1, \ldots \mathbf{u}_n] \in \mathbb{R}^{m \times n}$.

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:

$$\tilde{\mathbf{x}} = \sum_{i=1}^{k} a_i(\mathbf{x})\mathbf{u}_i = \mathbf{Ua} \tag{5}$$

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

$$a_i(\mathbf{y}) = \mathbf{u}_i^T\mathbf{y} = \sum_{j=1}^{m} u_{ji}y_j \ , \ i = 1 \dots k \ , \tag{6}$$

or, as a robust procedure [9], by solving a system of linear equations

$$y_{r_i} = \sum_{j=1}^{k} a_j(\mathbf{y})u_{r_i,j} \ , \ i = 1 \dots q \ , \tag{7}$$

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 (5) and determine the reconstruction error $\epsilon = ||\mathbf{y} - \tilde{\mathbf{y}}||$. The detection may either be performed based on this error [9] or by estimating the distance between the projected test sample and the projected training data [17].

3 Experimental Results

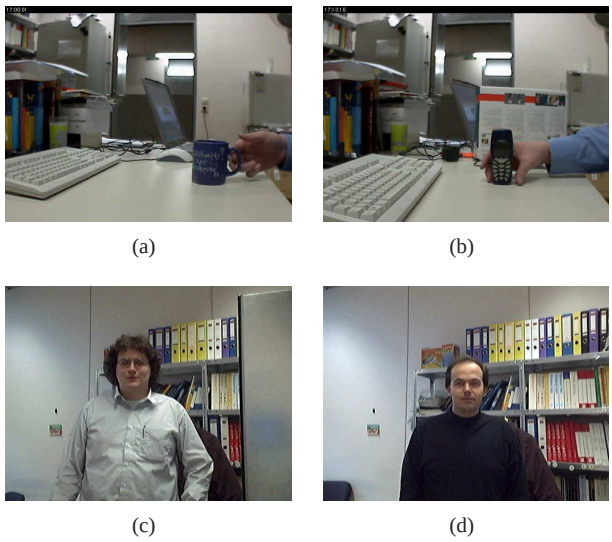
3.1 Description of Experiments

To demonstrate the approach we have defined two different tasks: detection of hand held objects (e.g., mobile phone, cup, etc.) and face detection for person recognition. Therefore, we have created separate sequences to train the classifiers and independent test sequences. Examples for the training sequences are depicted in Figure 5. For the hand held objects shown in Figure 5(a)-(b) different views of the target object were presented. As the learning method is patch-based only small rotations of the objects are permitted to preserve the aspect ratio; patches of different size were resized to a common patch size. For face detection the goal was to learn the natural variance of facial expressions and postures of the head (see Figure 5(c)-(d)).

For both scenarios the experiments are divided into three parts: First, the object of interest is manually selected in the training sequence to initialize the tracker and the object is tracked. Second, an object representation is learned by PCA using the patches obtained by the tracker. Finally, the previously computed classifiers are applied for a detection task on an independent test sequence.

3.2 Results

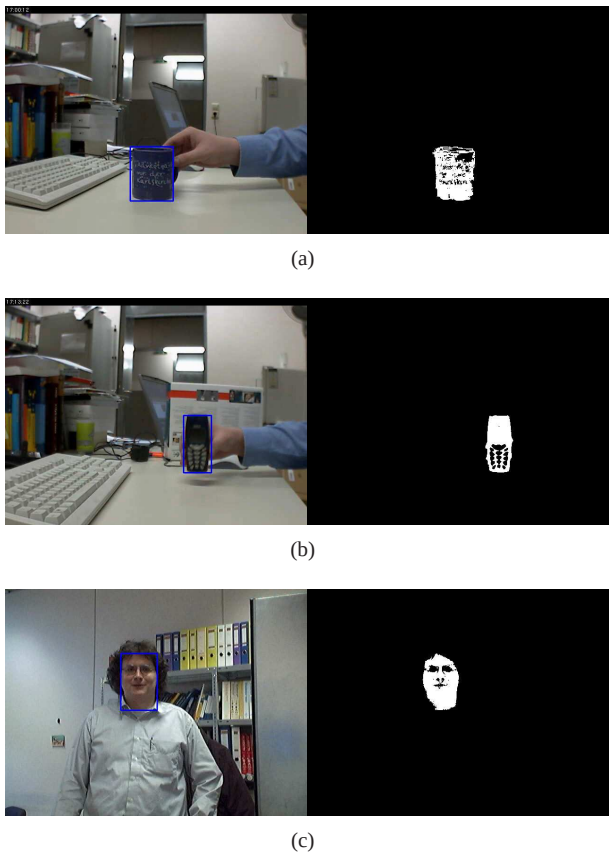
As the main contribution of this paper is to show that patches obtained by tracking an object can be used to train an object detector we show both, tracking results and results of detection. First, Figure 6 depicts the tracked objects and its corresponding MSERs. The MSERs may contain holes (eyes, texture, etc.). But since the patches are cropped based on the corresponding bounding boxes this is not a problem at all.



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

Figure 5: Training sequences used for tracking.

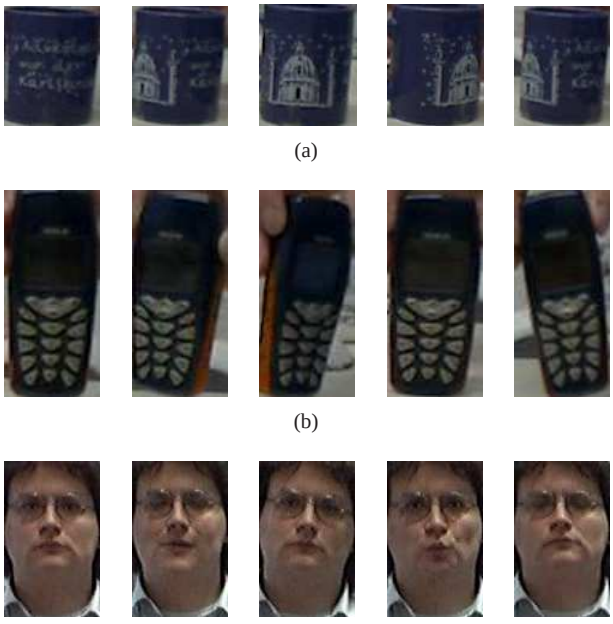
The obtained patches are resized to the same size and used for learning a PCA model. Examples of cropped patches are shown in Figure 7.



(a) (b) (c)

Figure 6: Detected MSERs during tracking.

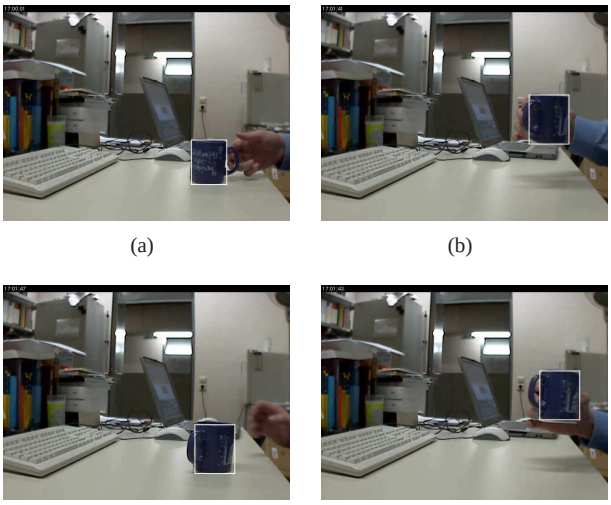
Next, we show some detection results. For the cup sequence 270 patches were extracted from a training sequence. Due to redundancy in the obtained set only every fifth patch was used for training the PCA model. For the detection task 5-10 eigenvectors were sufficient. The test data set is



(a) (b) (c)

Figure 7: Tracking results for training data sets.

more complex since the cup is presented at different positions. Therefore, the scale of the object is changing. But by evaluating the detector on three different scales we get perfect detections. The results are shown in Figure 8.

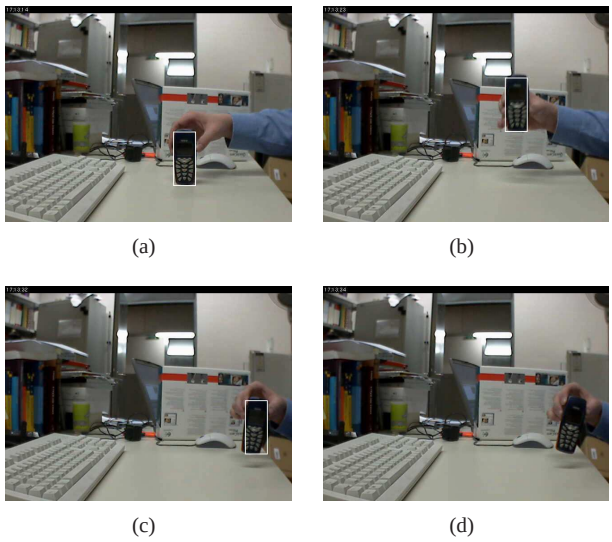


(a) (b) (c) (d)

Figure 8: Detection results for the cup sequence.

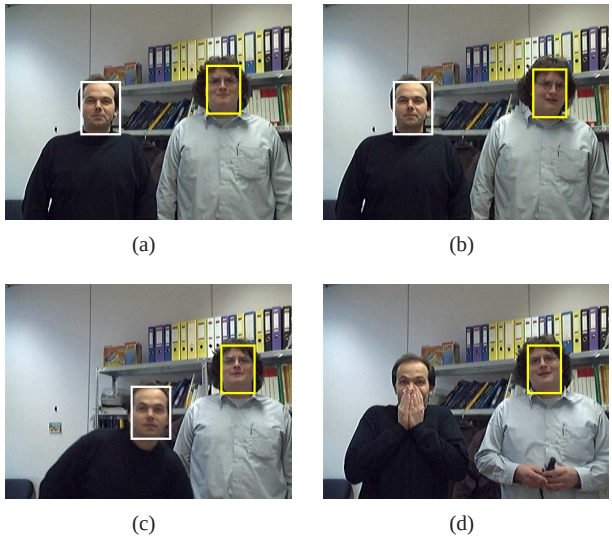
For the the mobile phone data set 300 patches were obtained by tracking but only 30 were used for PCA training. For the reconstructive model 5-10 eigenvectors were sufficient to achieve the desired accuracy. As can be seen in Figure 9 the object is always detected if the rotation of the mobile phone is within the specific range that was learned before. Otherwise the detection will fail (see Figure 9(d)).

Finally, we show some results for the person detection task. In the tracking stage more than 500 patches were extracted for both test persons (one separate sequence for each



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

Figure 9: Detection results for the mobile phone sequence.



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

Figure 10: Detection results for the person detection task.

test person). For learning the PCA model 60 patches and 5-10 eigenvectors were used for the reconstruction. The obtained classifiers are evaluated together on a single test sequence. As one can see in Figure 10 the faces are detected and the persons are recognized (white and yellow bounding boxes). Since the presented approach can simple be extended by adding robustness [9] even the person with the hands in front of the face (see Figure 10(d)) would be detected.

4 Conclusion

We have presented a new framework for learning an object representation from unlabeled data by combining a tracker with a learning method. Thus, an appearance based model of an object can directly be learned from a training sequence. Therefore, no training images taken on a turn table or obtained by manual segmentation are needed. The main idea is to use the output of the MSER based tracker directly as

input for batch PCA. The tracker is initialized one-time by manually selecting the object-of-interest. Then, the object is tracked and different views of the object can be learned. Thus, the human input for learning can be reduced to manually selecting an object once in the beginning. As our approach is quite general we have demonstrated the framework on different scenarios including desktop scenes and a face detection task. To take advantage of tracking an object - obtaining different views of the object that slightly change from frame to frame - an on-line learning method (e.g., incremental PCA [21]) may be applied. Furthermore the patches obtained by the tracker can be used as positive examples for boosting [26] or for conservative learning [19]. Thus, our next steps will be to automatically initialize the tracker by a color-based model (faces) and by change detection information (desktop scenes) and to apply a more suitable (on-line) method for learning.

Acknowledgement

This work has been supported by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04.

References

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

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. European Conf. on Computer Vision*, volume IV, pages 113–130, 2002.

[3] 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.

[4] 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.

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

[6] 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.

[7] Y. Freund and R. E. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[8] I. T. Jolliffe. *Principal Component Analysis*. Springer, 1986.

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

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

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

[12] 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.

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

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

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

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[24] 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.

[25] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[26] 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.

[27] 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.

[28] 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.