Adaptive Computer Vision: Online Learning for Object Recognition

Holger Bekel, Ingo Bax, Gunther Heidemann, and Helge Ritter

AG Neuroinformatics, Bielefeld University
P.O. Box 10 01 31, D-33501 Bielefeld, Germany
{hbekel, ibax, gheidema, helge}@techfak.uni-bielefeld.de

Abstract. The "life" of most neural vision systems splits into a one-time training phase and an application phase during which knowledge is no longer acquired. This is both technically inflexible and cognitively unsatisfying. Here we propose an appearance based vision system for object recognition which can be adapted online, both to acquire visual knowledge about new objects and to correct erroneous classification. The system works in an office scenario, acquisition of object knowledge is triggered by hand gestures. The neural classifier offers two ways of training: Firstly, the new samples can be added immediately to the classifier to obtain a running system at once, though at the cost of reduced classification performance. Secondly, a parallel processing branch adapts the classification system thoroughly to the enlarged image domain and loads the new classifier to the running system when ready.

1 Introduction

The introduction of neural networks to the field of computer vision has brought about a change of paradigms: No longer hard-wired knowledge is used to solve recognition tasks, instead, domain specific knowledge is acquired from examples, in a way both technically easier and cognitively more adequate. However, most neural recognition systems are still a half-hearted realization of this idea, because knowledge acquisition ends after an initial training phase. To accomplish online-learning, three basic requirements have to be fulfilled: (*i*) Flexibility of the neural system to allow the fast incorporation of new knowledge without performing an entire training cycle; (*ii*) close to real-time processing within the entire system; and (*iii*) a subsystem for human-machine interaction that allows to present new object knowledge in a natural manner of communication.

The system proposed in this paper is part an office task assistance system. To fulfill the aforesaid requirements, a neural three-stage system is applied that combines feature extraction with classification. When trained online, the last and most easy to train stage can be quickly adapted to provide a provisional solution. While the system is running continuously, in a parallel thread a new version of the neural system is trained from scratch and loaded to the running system when ready to improve performance.

To cope with the requirements of processing speed and human-machine interaction, an attentional subsystem allows the fast localization of regions of interest for object classification, and the simultaneous evaluation of pointing gestures to establish a common focus of attention (FOA) of human and machine.

While view-based systems often require large training sets [9, 8] to cover the variety of possible views, the system proposed here needs only a few frames of an object to facilitate interactive online training. This is achieved by artificially multiplying the available images to obtain new object views.

While solutions to several of the subtasks outlined above have been investigated, integration to larger vision systems is still rarely to be found. The *Perseus* system is able to reference objects a user is pointing at [3]. Ref. [10] proposes a communication model between image processing and the adjacent data interpretation for object recognition. A system for hand tracking and object reference that also allows the integration of modalities other than vision was proposed for the Cora robot system [13]. The approach presented here goes beyond these systems in three aspects: (*i*) Several vision capabilities are integrated in a common framework, (*ii*) using the neural approach, a single type of system deals with two sub-tasks in a unified way, and (*iii*) online learning is realized in a human-machine interaction loop.

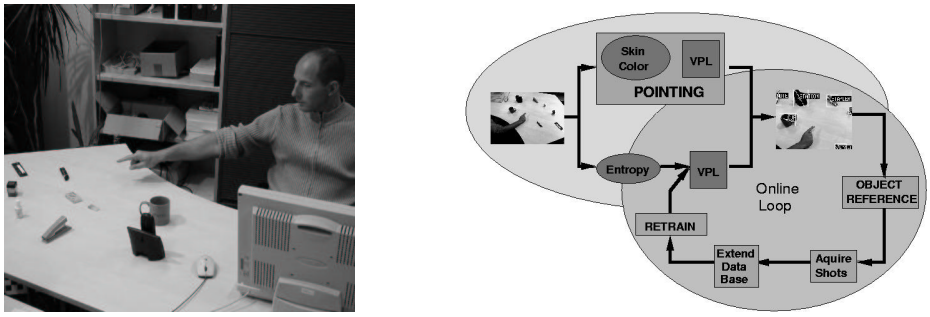

Fig. 1. Left: Office scenario, the user is pointing at objects. Right: Processing flow, starting with an image of the desk (left). In parallel, the scene is scanned (a) for pointing gestures (upper branch) using skin color segmentation and one instance of the VPL-classifier, and (b) for known objects (lower branch). If an object is pointed at, the "online loop" (right ellipse) is started. Once the referenced object location is identified, images are acquired. The database is extended by artificially distorted views (scale/shear transformation, translatory offset), then the VPL-classifier employed for object recognition is retrained (section 3.2). Note the two instances of the VPL are independent from each other.

The experimental setup is part of the VAMPIRE project (Visual Active Memory Processes and Interactive REtrieval). The work is aimed at the development of an active memory and retrieval system in the context of an augmented reality scenario. An important subtask is the recognition of objects and simple actions

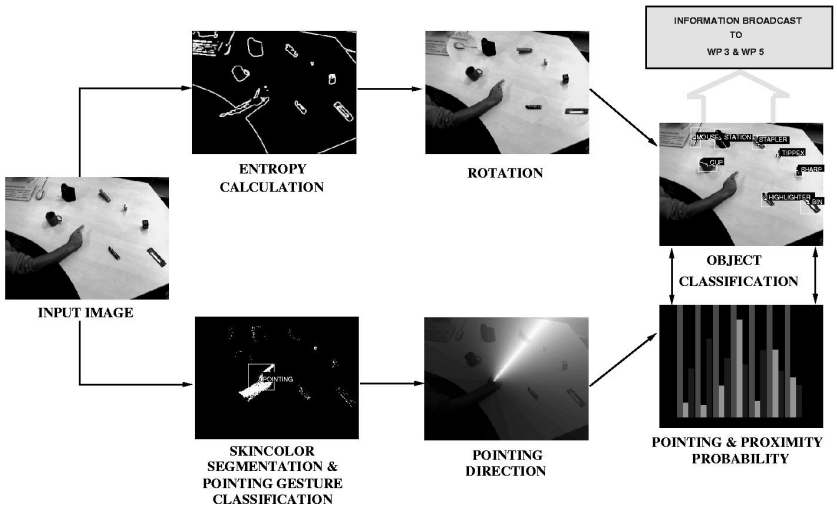

Fig. 2. Intermediate processing results. In the upper branch, the input image is processed for object classification, in the lower branch, hand gestures are detected. Objects are located by saliency maps, the example shows an entropy map. Hands can be detected by skin color. Located objects are virtually rotated to a normalized position to facilitate recognition. Recognized pointing gestures are "translated" into an attention map which shows a "beam" of activation in the pointing direction. The attentional subsystem described in [1] establishes the correspondence between pointing direction and one of the objects. In the right column, labeled objects and, below, probabilities for objects being pointed at are shown.

in an office environment. Here, a user sits in front of a desk, two cameras observe the scene, which permanently classify objects on the desk and interprete pointing gestures (Fig. 1 left). If either a completely unknown object is presented or if the system does not recognize an already trained object correctly, the user leads the system by gestures to train or retrain the object classifier. We will first describe the standard data flow of the object recognition (section 2), then in section 3 the online training triggered by hand gestures.

2 Object recognition system

Fig. 1 depicts the processing flow of the trained recognition system in the left ellipse, Fig. 2 shows some intermediate processing results. For object localization, saliency maps are computed using different mechanisms as described in [1]. Fig. 2 depicts only the "entropy map" as an example, which derives "conspicuity" of regions from their information content after the algorithm of [4]. The method relies on the assumption that semantically meaningful areas have also a high information content in the sense of information theory. Several saliency maps are integrated in the attentional subsystem to a joint saliency map. Parameterization of this module allows the selection of the scale on which structures

are evaluated (here: scale of office objects). Maxima of the joint saliency map indicate candidate regions for object classification. The representation of object locations by attention maps facilitates the integration of pointing directions to establish object reference (Fig. 2, for details see [1]).

Subsequently, the neural net based VPL-classifier classifies the candidate regions. The result is either a class number for an already trained object, or a reserved class label for "unknown". The VPL is a neural classification architecture which is particularly well suited for a fast, online training and retraining from a small data base. "VPL" stands for three processing stages, which combine feature extraction from the pixel level with classification: **V**ector quantization, **P**CA and **L**LM-networks (see [1] and references therein). The VPL-classifier extracts features from the input by local PCA, which are subsequently classified by a bank of LLM-networks. An overview of the processing flow is given in Fig. 3.

The VPL is trained as follows: The first level ("V") uses vector quantization (VQ) to partition the input space. For VQ, the algorithm proposed in [2] is employed. In the second level ("P"), for the training data assembled in the Voronoi tessellation cells of each of the resulting reference vectors, the principal components (PCs) are computed by the neural algorithm proposed in [12] to reduce dimensionality. I.e., to each reference vector a single layer feed forward network is attached for the successive calculation of the local PCs. In combination, the first two processing stages perform local PCA, which can be viewed as a nonlinear extension of simple, global PCA [14].

On the third processing level, to each PCA-net one "expert" neural classifier of the Local Linear Map – type (LLM network) is attached. The LLM network is related to the self-organizing map [5], see e.g. [11] for details. After the unsupervised training of the first and second level, the LLM-nets are now trained supervised.

The trained VPL-classifier is applied to classify in succession each of the candidate regions. Input are the raw pixel data of windows of pre-defined size located at the maxima of the saliency map. To each input vector, the best match reference vector is selected. Features are then extracted by projection of the input onto the local PCs. The overlap with the PCs is the input to the attached LLM-net, which yields the final classification.

3 Online Learning

If object classification is erroneous or new objects are to be added to the set, the user can activate the teaching mode. The teaching mode is realized as a finite state machine. It is activated by keyboard input, then the new or wrongly classified object on the desk must be indicated by a pointing gesture. The system memorizes the position of the object to be learned and starts to acquire shots of the object. After a sample image of the object has been taken, the system waits for the users hand to reappear and move the object to a different pose. When the hand is out of sight again, the system takes the next image automatically, and so on. The user decides when all relevant poses have been captured and finally

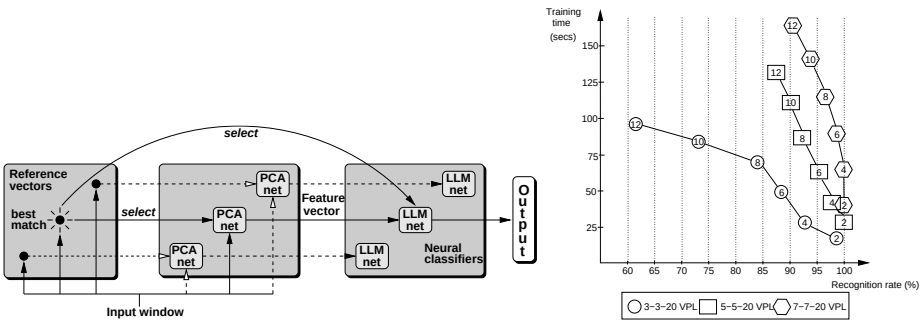

Fig. 3. Left: VPL-classifier. Features are extracted by local PCA and classified subsequently by neural classifiers. Right: Classification rates and training times for different size parameters of the VPL-classifier with respect to the number of objects learned (denoted within the symbols) using the *full training mode*.

ends the procedure by declaring the object either as "new" by giving it a new label, or as "improved" by using an old label (again using the keyboard).

After image acquisition, the system trains the currently used VPL-classifier in *fast training mode* and resumes classification. In parallel, a new VPL-classifier is trained in *full training mode*. In the following, the components of the online training are described.

3.1 Pointing Gesture Recognition

For pointing gesture recognition, a system proposed in [1] is applied, which can be described only in short. It is based on an adaptive skin color segmentation motivated by [7]. If a skin colored blob is found, the corresponding image region is classified by another instance of the VPL-classifier ("VPL-HAND"), which is not connected to the module employed for object recognition. VPL-HAND yields two pieces of information: (a) whether the skin-colored blob is a pointing hand at all, and, if so, (b) the pointing direction. The attention module then establishes the correspondence of the pointing direction and the referenced object, as described in [1].

3.2 Retraining the Classifier

The *full training mode* of the VPL-classifier comprises the three steps described in section 2: VQ, local PCA, and training of the LLM-networks. The novel, labeled object views are added to the existing set of training views, then a VPL-classifier is trained from scratch. Training time depends approximately linearly on the number of objects, most time consuming is the training of the PCA-nets. Therefore, the *fast training mode* leaves the V- and P-level unchanged and retrains only the LLM-nets. The method relies on the assumption that the existing feature extraction by local PCA is able to capture also the novel object.

So, for the newly captured views only feature vectors are computed and added to the existing set of feature vectors (which must be memorized). Each of the LLM-nets is then trained anew for the best match samples of its reference vector. Naturally, recognition rates are not as high as for the fully trained classifier.

To minimize the effort the user has to spend on teaching the system new objects, the set of online-captured training views is artificially expanded by two methods:

Scale/shear expansion: The appearance of objects within the workspace differs in scale and shear due to varying camera distance. Since the 3D-position of a newly acquired object is known, scaling and shearing transformations can be used to generate additional artificial views which cover the range of camera distances.

Translatory offset: The attentional subsystem is not always able to locate the reference frame — from which features are extracted — exactly at the object center. Therefore, object views with minor translatory offsets are added to the training set to improve classification robustness.

4 Evaluation

The VPL-classifier was tested for two aspects of online learning:

- How do different size parameters for the VPL-classifier affect training times and classification rates with respect to the number of objects?
- How do classification rates and training times differ for the *full training mode* and the *fast training mode* ?

For systematic evaluation, for each of 12 typical office objects (e.g. stapler, sharpener, highlighter) a set of 60 images was recorded in the following way: On the desk, six fixed positions were marked and the objects were placed at each of these positions. Then, 10 arbitrary poses of each object were recorded. The resulting set of images contains 720 samples of size 61×61 pixels. The 120 images recorded at a fixed reference position were used for training, the remaining 600 for testing.

The size parameters of the VPL are the number of reference vectors N_V, the number of local principal components N_P, and the number of LLM-nodes, N_L. So, VPL-size is given in the form N_V-N_P-N_L. Fig. 3, right, shows the results using the size parameters 3-3-20, 5-5-20 and 7-7-20 for the *full training mode*. The 3-3-20 classifier can be trained quite fast, but the recognition rate drops significantly as the number of objects to be learned is increased. The 5-5-20 and the 7-7-20 classifiers have better capabilities to learn more objects at high recognition rates, but the computational time needed for training increases.

Fig. 4 shows recognition rates for classifiers that were first trained in *full training mode* to recognize 2, 4, ... 10 objects and subsequently extended to recognize additional objects using the *fast training mode*. As expected, in all cases the recognition rate for a fixed total number of objects is better if all of them were trained in *full training mode*, as compared to some being learned in *fast*

training mode. The recognition rate drops when more objects are trained in *fast training mode*. However, this decay is much smaller if the classifier initially holds more objects (acquired in *full training mode*), because in this case the feature extraction covers a larger variety of appearances. While for two initial objects (recognized by 100%) performance drops dramatically, for eight objects the still good initial performance of 90% only reduces to 82%. In all cases, however, the drop in recognition rate after adding just *one* new object is tolerable.

Fig. 4 shows the average training times comparing fast and fully trained classifiers with respect to different numbers of objects. Training times for *fast training mode* clearly remain below the *full training mode*.

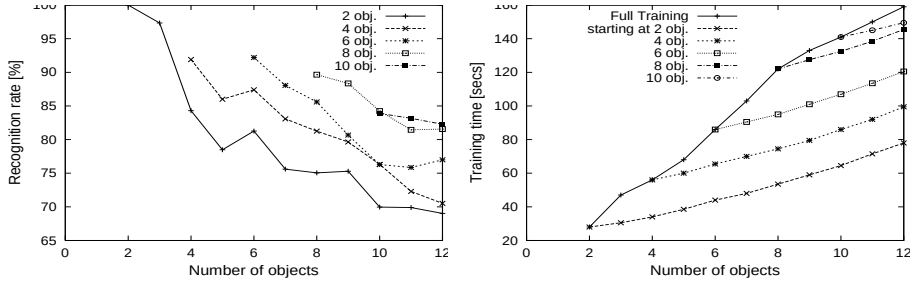

Fig. 4. Left: Classification rates for the *fast training mode*: Each curve visualizes the drop of the classification rate starting with classifiers that were fully trained on 2, 4, 6, 8 and 10 objects, respectively, and extended using the *fast training mode*. Size parameters were 3-8-20. Right: Training times for the *fast training mode* compared to full training mode. The main curve represents the training times for the *full training mode* and the forked curves represent training times for the *fast training mode* starting off with classifiers trained on 2, 4, 6, 8 and 10 objects, respectively.

5 Conclusion

We have presented a computer vision system for interactive online object learning guided by pointing gestures. The learning mechanism allows both to acquire new object knowledge and to improve classifications of already known objects. The system relies on a neural classifier, which builds a view based object representation. The system can be adapted fast to obtain a provisional version, while a full training is performed in the background. The performance of the provisional, fast training improves the more objects the system already knows — a property which is plausible also from a cognitive point of view.

An important goal of future research is a "self-diagnosis" of the system, which can give confidence values for the systems ability to classify objects correctly. By this means, an estimate for the necessary object views could be given during online training, depending on the object's complexity. Moreover, the system

could ask for more training views of objects where classification appears unreliable. The self-diagnosis should also be able to judge whether the existing feature extraction is sufficient for a newly acquired object. Thus, it would be possible to decide in which cases *fast training mode* makes sense. Another goal is accelerating the offline feature adaptation, a promising approach is e.g. proposed in [6].

6 Acknowledgment

This work was supported within the project VAMPIRE, which is part of the IST programme (IST-2001-34401).

References

1. G. Heidemann, R. Rae, H. Bekel, I. Bax, and H. Ritter. Integrating context-free and context-dependent attentional mechanisms for gestural object reference. In *Proc. Int'l Conf. Cognitive Vision Systems*, pages 22–33, Graz, Austria, 2003.
2. G. Heidemann and H. Ritter. Efficient Vector Quantization Using the WTA-rule with Activity Equalization. *Neural Processing Letters*, 13(1):17–30, 2001.
3. Roger E. Kahn, Michael J. Swain, Peter N. Prokopowicz, and R. James Firby. Gesture recognition using the perseus architecture. Technical Report TR-96-04, 19, 1996.
4. T. Kalinke and W. von Seelen. Entropie als Maß des lokalen Informationsgehalts in Bildern zur Realisierung einer Aufmerksamkeitssteuerung. In B. Jähne, P. Geißler, H. Haußecker, and F. Hering, editors, *Mustererkennung 1996*, pages 627–634. Springer Verlag Heidelberg, 1996.
5. T. Kohonen. *Self-Organizing Maps*. Springer Verlag, 1995.
6. A. Leonardis, H. Bischof, and J. Maver. Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627, 2002.
7. H. J. Andersen M. Stoerring and E. Granum. Physics-based modelling of human skin colour under mixed illuminants. *Robotics and Autonomous Systems*, 35(3–4):131–142, 2001.
8. B. W. Mel. SEEMORE: Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation*, 9:777–804, 1997.
9. H. Murase and S. K. Nayar. Visual Learning and Recognition of 3-D Objects from Appearance. *Int'l J. of Computer Vision*, 14:5–24, 1995.
10. J. C. Ossola, F. Bremond, and M. Thonnat. A communication level in a distributed architecture for object recognition. In *8th International Conference on Systems Research Informatics and Cybernetics*, Aug 1996.
11. H. J. Ritter, T. M. Martinetz, and K. J. Schulten. *Neuronale Netze*. Addison-Wesley, München, 1992.
12. T. D. Sanger. Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network. *Neural Networks*, 2:459–473, 1989.
13. C. Theis, I. Iossifidis, and A. Steinhage. Image Processing Methods for Interactive Robot Control. In *Proc. IEEE Roman International Workshop on Robot-Human Interactive Communication*, Bordeaux and Paris, France, 2001.
14. M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.