Neural Networks, Vol. 18, No. 5/6, 2005

Interactive Image Data Labeling Using Self-Organizing Maps in an Augmented Reality Scenario

Holger Bekel, Gunther Heidemann and Helge Ritter
Neuroinformatics Group
Bielefeld University
P.O. Box 10 01 31, D-33501 Bielefeld, Germany
E-mail: {hbekel,gheidema,helge}@techfak.uni-bielefeld.de

***Abstract*— We present an approach for the convenient labeling of image patches gathered from an unrestricted environment. The system is employed for a mobile Augmented Reality (AR) gear: While the user walks around with the head-mounted AR-gear, context-free modules for focus-of-attention permanently sample the most "interesting" image patches. After this acquisition phase, a Self-Organizing Map (SOM) is trained on the complete set of patches, using combinations of MPEG-7 features as a data representation. The SOM allows visualization of the sampled patches and an easy manual sorting into categories. With very little effort, the user can compose a training set for a classifier, thus, unknown objects can be made known to the system. We evaluate the system for COIL-imagery and demonstrate that a user can reach satisfying categorization within few steps, even for image data sampled from walking in an office environment.**[1]

I. Introduction

The increase of computational power accompanied by ever continuing miniaturization has made mobile systems become a part of the everyday life. One of the main challenges in mobile computing is the development of user assistance systems which can support a human in variable and "generic" tasks. Examples for such tasks in everyday life are the memorization of faces or the retrieval of lost objects. In this paper, we describe a mobile system which is capable of interactive object learning from examples and retrieving objects out of view. The focus of this contribution is on the efficient organization and visualization of the online labeling process using the mobile gear by means of self-organizing maps (SOMs) [Kohonen, 1982], [Kohonen, 1995].

The mobile system used here is an augmented reality (AR) gear with two head-mounted cameras and a display by which the user views the original scene together with overlays ("augmentations") of the system (Fig. 1). Our system employs the view-based approach for object recognition and interactive object learning.

[1]An abbreviated version of some portions of this article appeared in [Bekel et al., 2005], as part of the IJCNN 2005 conference proceedings, published under the IEEE copyright.

The concept of memorizing views instead of explicit geometric and surface reflectance models offers the well-known advantage that object knowledge can be conveniently acquired from samples (e.g. [Koenderink and van Doorn, 1979], [Murase and Nayar, 1995], [Mel, 1997]). However, still sample views must (*i*) be labeled, (i.e. the object class must by supplied by the user in some way), and (ii) be made available in sufficient number to cover the complete "appearance space" of the object. While this problem applies to view based recognition in general, it imposes particular difficulties to mobile systems, where labeled view acquisition requires user effort.

In earlier versions of the AR-system presented here, the user had to present the objects to the system one by one and isolated on homogeneous background. Using the mobile interface, typically 5–10 sample views were acquired per object quite conveniently, then the classifier was re-trained internally [Heidemann et al., 2004a], [Heidemann et al., 2004b]. However, this procedure still requires considerable effort and is, moreover, unsatisfying from a cognitive point of view: Children do not learn to recognize an object in a one-time process of memorizing views under a given class label! Instead, children explore their surroundings by repeated interaction with many objects and often learn the label (i.e. the class) of an object much later. I.e., they form the *concept* of a certain object much earlier than learning its name.

Led by these considerations, the concept of the mobile object recognition system presented here can be summarized as follows: The system samples views from its environment *permanently* while the user moves around in a restricted environment (e.g. an office). Since it is impossible to memorize the entire input stream of images, context-free modules for focus-of-attention provide a selection of possibly "interesting" image patches, a subset of which hopefully contains the relevant objects. After a period of sampling, the user can switch into "labeling" mode. The system then trains a SOM using features extracted from the sample patches and visualizes prototypes of the samples in the spatial organization imposed by the SOM. Thus, the system provides a "pre-categorization" of the viewed domain, which allows the user to select and label the subset of relevant objects (e.g. cup, pencil) while discarding others

(e.g. background structures).

SOMs are particularly well-suited for the combined task of (*i*) mapping a high-dimensional and highly non-linear data distribution to only two dimensions while (*ii*) conserving local neighborhood relations for fast and easy-to-use visualization, see e.g. [Kohonen, 1995], [Martinetz and Schulten, 1994], [Villmann et al., 2003]. There are numerous applications of SOMs for unsupervised clustering and visualization, examples can be found in [Somervuo and Kohonen, 1999] or [Tokutaka et al., 1999]. Most similar in thought to the present application is probably the PICSOM system, which organizes images using tree-structured SOMs to support the user in retrieval tasks [Laaksonen et al., 2002]. An overview of the multifaceted applications of SOMs is given in [Oja et al., 1999].

However, pre-categorization achieved by the SOM can only be as good as the underlying data representation. Therefore, features for the characterization of the extracted image patches have to be found which not only reduce data dimensionality but also generate descriptions of high variance, a well-balanced cluster structure and high discriminative power. Only if feature extraction fulfills these criteria, is there a chance to obtain semantically meaningful categories by clustering techniques. In the context of compression, refs. [Manjunath et al., 2001] and [Sikora, 2001] propose MPEG-7 features to encode efficiently the majority of image characteristics. Since it is well known that combining different types of features helps to fulfill the above criteria [Deselaers et al., 2004], the combinations of MPEG-7 features have been analyzed statistically in [Eidenberger, 2003]. In this work, SOMs are used to analyze contributions of different MPEG-7 features to variance and cluster formation. Because of the good results MPEG-7 features have achieved, we employ part of the feature set in combination with a SOM for fast clustering, visualization and interactive labeling in the mobile system.

After this short outline of the idea, we will describe first the scenario and the interactive functionality of the system in section II, then the features sets (section V) and the iterative SOM training (section III). The system is evaluated in section VI on the COIL-dataset, an example for the efficiency achieved by iterative SOM projections is given in a "real world" scenario in section VII. Results and future lines of development are discussed in section VIII.

II. System Description

A. Scenario

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 in an office environment. Here, a user sits in front of a desk or moves around in his office (Fig.1). Two head-mounted cameras observe the scene in front of the user, the images are permanently visualized in a "see-through-loop" to the head-mounted display, together with overlayed "augmentations" of the system. Such augmentations are e.g. classifications of recognized objects, as well as tools for interaction like buttons and menus (Fig. 3). The long term goal of the project is a "personal assistant", which acquires knowledge about the environment and can answer queries for information about objects or retrieve "lost" objects.

Fig. 1. User with head mounted cameras and display. The input is looped back into the display and enhanced by visual augmentations.

The system brings up questions of multi-modal interaction with mobile systems [Heidemann et al., 2004a], online categorization and learning, and active memory architectures. No effort was spent on miniaturized hardware: The user wears a backpack with a laptop for frame grabbing, early image processing, visualization, audio in/output and communication tasks. Additional processing units are connected via WLAN.

B. System architecture and control flow

The system consists of several independently running modules which are organized in a flat architecture. There are basically three types of components: *Input* modules for processing of vision and speech, *output* modules for display of augmentations and menus as well as auditory feedback, and a *control* module (Fig. 2).

The input modules are independent of each other and provide a continuous stream of processing results. The control module is realized as a finite state machine. Its state depends on the current task, e.g. "acquisition of object samples", "structuring image data", or other menu interaction. Depending on the state, the control module selectively evaluates the currently relevant data from the input modules, switches between states,

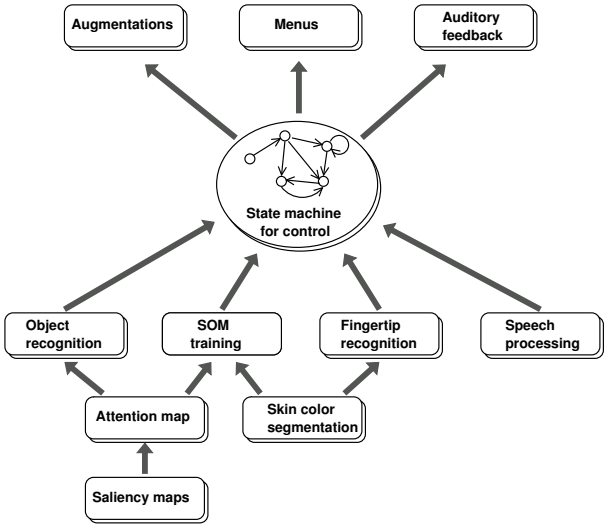

Fig. 2. System architecture: Input modules running in parallel are evaluated by a central control module, realized as a state machine. State transitions reflect varying modes or tasks. The state machine is the "client" of the visual and auditory output servers, to which it directs its output.

and sends data to the output modules. The latter display augmentations, give auditory feedback, and show appropriate menus, highlighting, etc. The output modules are in the role of servers, which act on requests of the client, i.e. the control module. For efficiency, the control module deactivates input modules currently not needed.

Since the focus of this paper is the categorization and semi-automatic labeling of image data, we refer to [Heidemann et al., 2004a] for a more detailed description of the control flow and the modules.

C. Menu based control

For communication with the system, on the right hand side of the image displayed in the AR-gear a semi transparent menu is overlayed. Fig. 3 shows the "SOM" sub-menu, which is displayed to label image data interactively. There are three types of menu buttons: *triggers*, which cause a certain action to be carried out, *check-buttons*, which can be turned "on" or "off", and sets of exclusively coupled *radio-buttons*. E.g., as long as the *check-button* "Show Examples" is active, the examples matched on a selected node are being displayed. The *trigger* "Retrain Som" retrains the SOM (after selection of new feature weights, see section V).

For reliable menu communication, buttons must be first *selected*, then *pressed*. Selection can be carried out by speech, naming the label of the button, or by gesture (user indicates a button with his/her finger). After selection by gesture, the button can be pressed either using speech ("Yes"), or by moving the finger inwards. If a button was selected by speech, it must also be "pressed" using speech. Menu operation is accompanied by visual and auditory feedback for selection and pressing. In principle, eye-tracking would be another suitable method for fast menu control, however, here we prefer a method which is intuitively simple for the user and does not require additional apparatus. Of the various functionalities

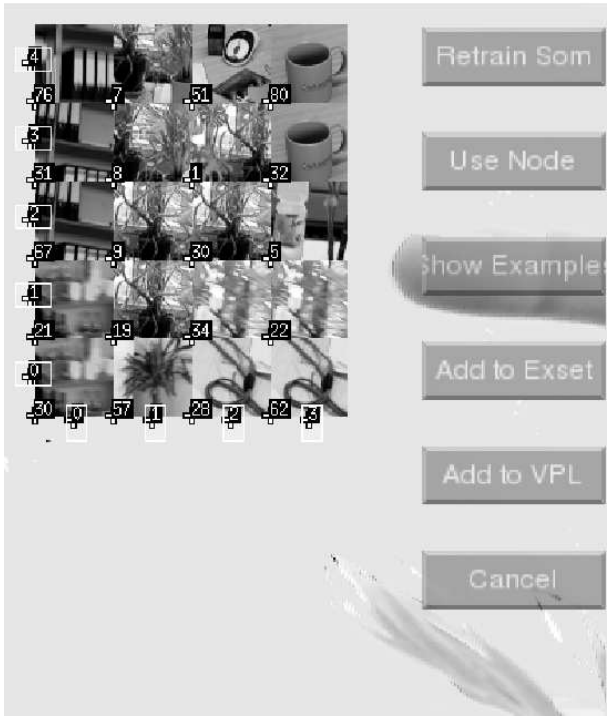

Fig. 3. Menu control by fingertip: The user carries out a "pressing" gesture on a button. The system analyzes the position and the movement trajectory to recognize "Selected" and "Pressed" events. Here the user wants to display the best match examples of a selected node of the overlaid SOM.

controlled by menus, here only the interactive acquisition of image data and the subsequent labeling using SOMs will be explained in the following.

III. Image Acquisition

The acquisition of image patches is controlled by the "Record" sub-menu. In this mode, the system permanently records image patches supposed to be salient by the attention module. Selection of the most salient patches not only restricts the amount of data to be memorized, but is in itself part of the recognition and categorization system: Only such local patterns which can be reliably detected by the attention module are promising candidates for labeling and training.

The attention module is therefore an important component of the system. It consist of several independently working modules for *interest point* (IP) detection, specialized on different structures such as edges and corners [Harris and Stephens, 1988] or color symmetries [Heidemann, 2004]. Here, we can only outline the most simple but highly effective IP-detector. It draws on principles realized in human visual system [Treue, 2001]: The image stream of one camera is convolved with a 3×3 Laplacian filter. The output is multiplied pixel-wise (not convolved) with a Gaussian kernel, which is centered in the image and has a variance of half the image size. Thus, the focus is set on the center of the image. Subsequently, the image is smoothed by convolution with a 5×5 Gaussian kernel. Variance of this kernel is chosen such that object borders obtained by the Laplace-filtering become connected (Fig. 4). Connectivity

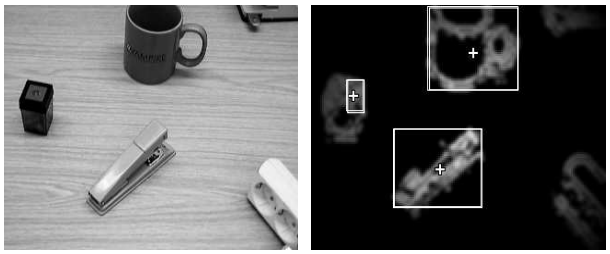

Fig. 4. Left: Image patch of the camera input. Right: Output of the attention module. The white frame surrounds the computed salient regions marked by a cross at the center.

analysis of a binarized version yields IPs centered on salient regions, around which patches of pre-defined size are recorded to an intermediate memory.

The acquisition of patches suffers from two types of disruptions: Blurring by motion, and the recording of identical patches if there is no movement for some time. Therefore, we employ a motion suppression algorithm to stop recording during strong movements. The algorithm computes the location of *corresponding* IPs over the last frames. An IP I_i^t is defined to correspond to I_j^{t-1} if $I_j^{t-1} = \min_k d(I_i^t, I_k^{t-1}) < t_1$, where $d(\cdot,\cdot)$ is the distance in the image plane, and t_1 is a proximity threshold. To discard unstable IPs caused by flicker or other disruptions, the operator starts motion computation only if the correspondence for an IP can be established for at least three frames. Motion is assumed "strong" and patch recording suppressed if

$$\frac{1}{n}\sum_{j=1}^{n}|\frac{1}{\Delta t}\sum_{i=1}^{\Delta t} I_j^t - I_j^{t-i}| > t_2, \qquad (1)$$

holds, where n is the number of stable pursued IPs and t_2 a threshold for the strength of the tolerated movement. Once a period of strong motion ends, IP-centered patches are recorded *only once* to avoid multiple sampling of static scenes. So, samples are recorded after each movement, then the system waits for the next movement.

IV. Image Features

MPEG-7 defines a standard for image analysis and compression [MPEG-7, 2001], including the definition of color, edge and texture features. Based on a study of [Eidenberger, 2003], we employ a combination of three of these features to cover as much as possible of the image characteristics: (*i*) The edge histogram which represents the distribution of edges (tailored version of the algorithm), (*ii*) the color layout descriptor, and (*iii*) the scalable color descriptor. The descriptors can be outlined only in short, see [Manjunath et al., 2001] for details.

A. Edge Histogram

The spatial distribution of edges is captured by the edge histogram descriptor. First, the image patch is subdivided in 4×4 sub-images. In the standard MPEG-7 descriptor, for each of these sub-images an edge histogram is computed. The edge directions are sorted into five prototypical directions: horizontal, vertical, 45 deg, 135 deg, and non-orientation specific. To compute these histograms, the sub-images are divided into a desired number of blocks. Each of these blocks is treated as a 2×2 pixel image, where the new pixel gray values are computed by averaging the values of the corresponding pixels. The edge strength of each block is computed by convolution with 2×2-filters for each direction, for details see [Manjunath et al., 2001]. In the standard MPEG-7 descriptor, the dominant edge type of this block contributes to the corresponding histogram if its strength exceeds a given threshold. The behavior of this descriptor depends on the block size. Tab. I depicts the block size for different numbers of blocks for COIL images of size 128×128. To avoid loss of information for small images, a modified descriptor was applied which evaluates all five values for the local edge strength to gain a continuous value for each edge type per sub-image. Thus, for both descriptors the image partitioned in 16 sub-images results in 80 values. Additionally, performance is increased using for each edge type a global average value which is appended to the feature vector, as proposed in [Park et al., 2000]. The performance for both descriptors with different numbers of blocks is compared in section VI-A.

Number of Blocks	128	256	512	1024	2048
Edge Length of Block	12	8	6	4	2

TABLE I

Block sizes and corresponding edge lengths of the blocks for edge feature computation of the test datasets.

B. Color Layout

To take into account the spatial color distribution, the "compact color layout descriptor" is applied. First, RGB color values are transformed to YCrCb. Then the image is divided into 64 blocks (on an 8×8 grid) and the average color computed for each block. On the 8×8 grid of average color values, a discrete cosine transform is computed. For the combined feature vector used here, the low-frequency coefficients are selected by zig-zag scanning. As proposed in [Manjunath et al., 2001], we use six Y, three Cr and three Cb coefficients.

C. Scalable Color

The "scalable color descriptor" is a global color feature. It is based on a histogram of HSV color values and a subsequent Haar transform for efficiency and to make the histogram scalable.

The basic steps of this algorithm are as follows. The HSV space is quantized to 256 bins, which include 16 levels in H, 4 levels in S, and 4 levels in V. Bin values are non-uniformly quantized to a value ranging from 0 to 2^{11}. Before performing the Haar transform, all bin values are further normalized by a nonlinear quantization to 4-bit values, giving higher significance to the small values with higher probability.

Finally, a Haar transform calculates high-pass (realized by a difference operation on adjacent bins) and low-pass (realized by a sum-up operation on adjacent bins) coefficients. Regarding the histogram as a 16×16 matrix, a histogram with half the number of bins is obtained by summing pairs of adjacent histogram lines. Performing this step iteratively, the Haar transform results in a histogram of 128, 64, 32, ... bins. Thus, the scalability is given by the number of iterations. This process is also used to encode the histogram using the high-pass coefficients which can be seen as its fine resolution representation. Due to the expectation that adjacent colors vary only slightly, they usually have small values which can be mapped to 8-bit values by a linear quantization. The feature vector contains these integer values of the desired number of bins. Here, a quantization of 64 bins is applied.

V. Iterative Labeling

After sample patches have been memorized over a period of time as outlined in section III, the features described in section IV are extracted. The patches sampled from the office environment comprise highly different structures, such as parts of shelves, edges of tables as well as objects on the desk. So, only part of the patches are samples of relevant objects. Since labeling the huge amount of patches manually is a cumbersome task, in a first step ("pre-screening") the samples are mapped onto a SOM to discard irrelevant patches. Then, iterative mapping of the remaining sample patches to (new) SOMs allows labeling. We will first describe the standard SOM, then the iterative labeling procedure using SOMs with changeable weights for individual feature blocks.

A. Self Organizing Map

We apply a standard SOM as proposed by Kohonen [Kohonen, 1995]. SOMs provide the possibility to project high dimensional data onto a low-dimensional grid while preserving topology. One major advantage of the SOM is its ability to visualize data of high dimensionality [Vesanto, 1999].

The SOM consists of a set of prototype or weight vectors $\mathbf{w}_i \in \mathbb{R}^n$ which are associated with the nodes of a regular low dimensional grid. After initialization with random weights, the prototypes are trained iteratively using the learning rule

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) \cdot h_{c(\mathbf{x}),i}(t) \cdot (\mathbf{x}(t) - \mathbf{w}_i(t)), \quad (2)$$

where t is the iteration step index and $c(x)$ denotes the best matching prototype with the smallest euclidian distance to a randomly chosen sample $\mathbf{x}(t) \in \mathbb{R}^n$. The adaption is controlled by the neighborhood function $h_{c(\mathbf{x}),i}(t)$ and the learning rate function $\alpha(t)$, both decreasing monotonically with time t. Usually, a Gaussian neighborhood function is applied:

$$h_{c(\mathbf{x}),i}(t) = \exp\left(-\frac{||\mathbf{r}_i - \mathbf{r}_{c(\mathbf{x})}||^2}{2\sigma^2(t)}\right), \quad (3)$$

with $\mathbf{r}_i$ and $\mathbf{r}_{c(x)}$ denoting the position of the corresponding grid nodes. The width of the neighborhood function is varied by the monotonically decreasing function $\sigma(t)$. In contrast to other vector quantization algorithms, the constraint of the neighborhood function leads to preserving topology. Thus, SOMs are especially well-suited for structuring and visualizing image data in our approach.

B. Iterative Labeling and Retraining

Data labeling is carried out in iterative steps of training SOMs, visualization, and labeling. First, the features described in section IV are used to form the combined feature vector $\mathbf{x}$, which is a concatenation of the weighted feature blocks:

$$\mathbf{x}^\star = (w_e * x_1, \ldots, w_e * x_m, w_c * x_{m+1}, \ldots, w_c * x_n), \quad (4)$$

with a common weight w_e for each component of the m-dimensional edge feature block, and a weight w_c for the $(n-m)$-dimensional block of color features. For the first step, the weights w_e and w_c are computed to normalize the variance of each feature block.

For the sample patches represented by the $\mathbf{x}^\star$, a two-dimensional SOM is trained. To visualize the resulting SOM, both the feature data set and the corresponding image patches must be kept in memory. For each SOM node, the corresponding image patch to the best match feature vector is displayed in the AR-gear. Additionally, for each node the number of the best match feature vectors is depicted, see Fig. 8, top. If visualization of a SOM node by only its best match sample patch seems not sufficient to the user, the system can be told to display the entire subset of patches mapped on a SOM node, e.g. by the speech command "Show me best match examples of node (3,5)" or selecting the corresponding row and column with the fingertip. Now the user has three possibilities:

(1) In case the image patches show **irrelevant patterns** or blurred patches, the user discards these patches, because it is not useful to classify them.
(2) If all image patches of a node show **the same object** (though maybe different views), the user can assign a class-label and add the samples to the training data base of the classifier.
(3) If samples mapped to one node stem from **different objects** and can thus not be given a class label, the user can assign a group ID instead to these examples. If the examples mapped on adjacent nodes show the same mixture of objects, they get the same group ID.

For the samples of case (1) and (2), the labeling procedure is finished since they have either been discarded or have received a class label and can be added to the training data set. For the yet unlabeled samples of case (3), SOM-training is repeated to visualize the subset again, now spread out over all nodes of the SOM for a better visualization. In case group IDs have been assigned, it is also possible to train individual SOMs just for the group to achieve a still better visualization. This iterative process of SOM-training and labeling is accompanied by adjusting the relative feature weighting w_e/w_c. Thus, the user can influence the categorization by emphasizing the feature type (edges or color) which he/she thinks to be more discriminative. For examples see section VII. Once the training

set is complete, a suitable classifier can be trained (which is no subject of this paper, see [Heidemann et al., 2004b]).

VI. Evaluation on standard datasets

For parameter adaptation and a reproducible evaluation of system performance, the system was first tested on two standard data sets:

"Orig": A subset of 20 objects from the COIL-100 image library [Nene et al., 1996], with 72 views for each object. Thus, the dataset contains 1440 RGB-images of 128×128 pixels (Fig. 5).

"Distort": The images of **orig** were distorted by random translation of +/- 5 pixels in x- and y-direction and random scaling of +/- 10%.

A. Adjustment of the edge histogram descriptor

As described in section IV-A, the number of blocks is an important parameter of the edge histogram descriptor. While MPEG-7 features are mostly used in image retrieval applications describing the characteristics of sizable images, here only small image patches are processed. Hence, the behavior of the descriptor has to be adjusted to the application. Therefore, we compare the standard MPEG-7 edge histogram descriptor with a modified descriptor as described above. For the modified descriptor, not only the vast majority of the edge type for a block leads to an entry in the histogram, but *all* edge values are processed.

For both datasets and both descriptors, feature vectors were computed for different edge lengths of the blocks (see Tab. I). On this feature set, a SOM of 8×8 nodes was trained in 15000 steps with exponentially decreasing of step size α from 0.9 to 0.1 and width σ from 2.0 to 1.0 (equation 3). This parameterization proved to yield good results to embed the SOM in the feature space with low errors and is kept constant for the mobile system and further evaluation.

To each SOM node, the class label of the *majority* of the matched objects is assigned. All other objects mapped to this node are counted as errors. Fig. 6 depicts the average error rates for ten runs for both datasets and five different block sizes using both features. For the standard edge histogram descriptor, error rates decrease with decreasing block size.

Generally, the error rates of the modified descriptor are lower than the error rates of the standard descriptor. They exhibit a clear minimum at block size 4×4 pixel. Thus, for small images the modified descriptor preserves discriminative characteristics better. For the following evaluation, only the modified descriptor with this parameterization will be used.

B. System performance evaluation

To judge the manageability of the proposed system, different aspects are important. Computational cost for the visualization must be small to facilitate online use. The employed edge- and color descriptors lead to satisfying times, e.g., training of a 8×8–SOM with 15000 steps on a dataset **orig** (1440 128×128 RGB-images) requires, on average, 5.3 secs on a standard Pentium IV 2.5 GHz processor.

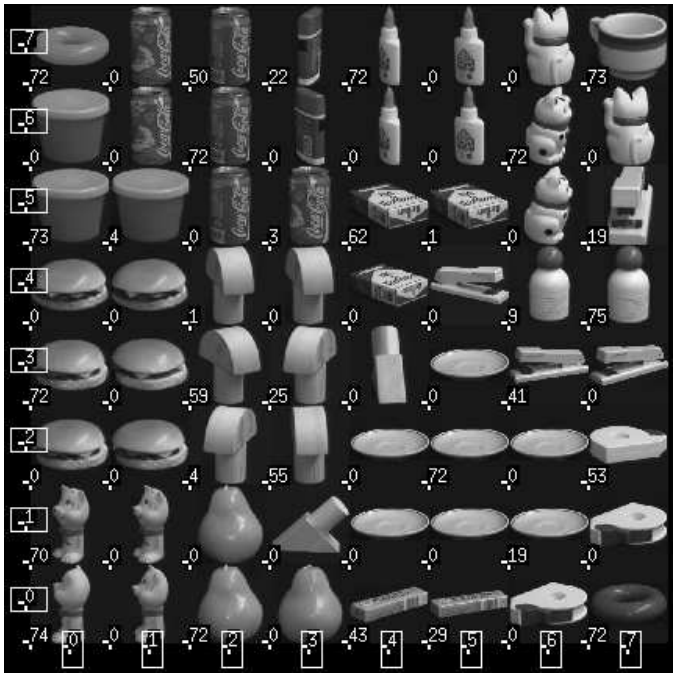

Fig. 5. Projection of coil imagery onto a SOM, see text.

Features	Dataset	First Training Hit	First Training Errors	First Training Error Nodes	Retrain Error	Retrain Hit
all	orig	33.5	**1.083**	3.1	**0.069**	13.2
all	distort	33.9	**1.861**	3.9	**0.083**	19.7
color	orig	34.0	**2.013**	4.3	**0.333**	18.3
color	distort	31.7	**2.569**	4.5	**0.243**	19.9
edge	orig	56.4	**16.736**	36.8	–	–
edge	distort	57.2	**16.125**	39.0	–	–

TABLE II

Error rates and number of error nodes for SOM training (for details see section VI.)

Further, labeling effort must be evaluated. A good measure is the number of nodes the user has to "touch" for labeling the dataset, which is equal to the number of nodes to which samples of more than one class are assigned (such that the user has to retrain for these samples). Another important quantity is the total number of errors in the first SOM, since it reflects the remaining effort (a node with only 1% of wrong samples is more likely to split up into "pure" nodes already in the next iteration than a node with 49% of wrong samples).

Tab. II shows the results for a 8×8–SOM: The first column indicates the used feature(s) (*all* standing for both edges and color), the second the dataset. Column *Hit* gives the percentage of nodes onto which semantically meaningful samples are mapped (opposed to unusable samples), *Errors* gives the percentage of samples mapped to such nodes onto which different objects are mapped, and *Error nodes* is the (absolute) number of nodes out of 64 onto which different objects are mapped.

Even for the first training, *Errors* is low, the positive effect of combining edge and color descriptors is obvious. From *Error nodes*, it becomes clear that user effort is manageable

even considering the constraints of a mobile system — only 3–4 nodes need re-training, the rest undergoes labeling at once. Fig. 5 shows that, on the whole, the dataset is very well structured. In contrast, Fig. 7 shows a selection of objects which are often mapped together onto SOM nodes and require retraining. The numbers are the sum of the entries of the corresponding confusion matrix. Retraining these samples using a 4×4–SOM leads to error rates below 0.1% (Tab. II, right part), so, only one additional training/labeling step is sufficient. Moreover, the number of nodes need to be labeled was only 13.2–19.7 in this last step (rightmost column).

Summarizing, the combination of edge and color features leads to good pre-categorization by the SOM and thus low labeling effort, both in terms of the number of nodes that have to be handled and the number of retraining steps needed. Adjusting relative weighting of edge and color features facilitates the process, as illustrated also in section VII.

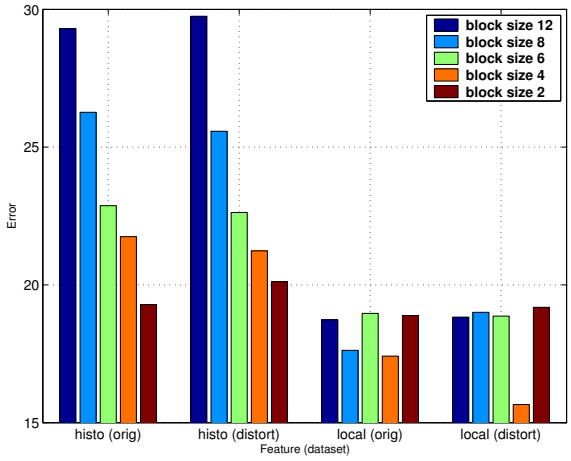

Fig. 6. Error rates for the local edge histogram (*histo*) and the absolute local edge values (*local*) for dataset **orig** selected from COIL-100 and dataset **distort**.

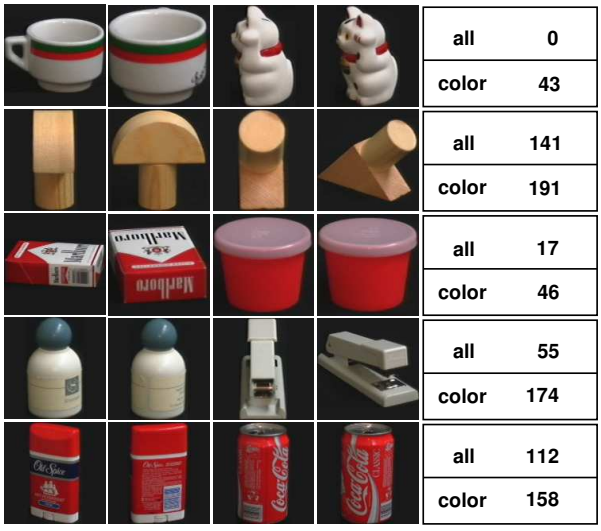

Fig. 7. Object pairs often mapped together on SOM nodes with the number of confusions as sum of the values in the confusion matrix for a combination of *all* features in comparison to the combination of the two *color* features.

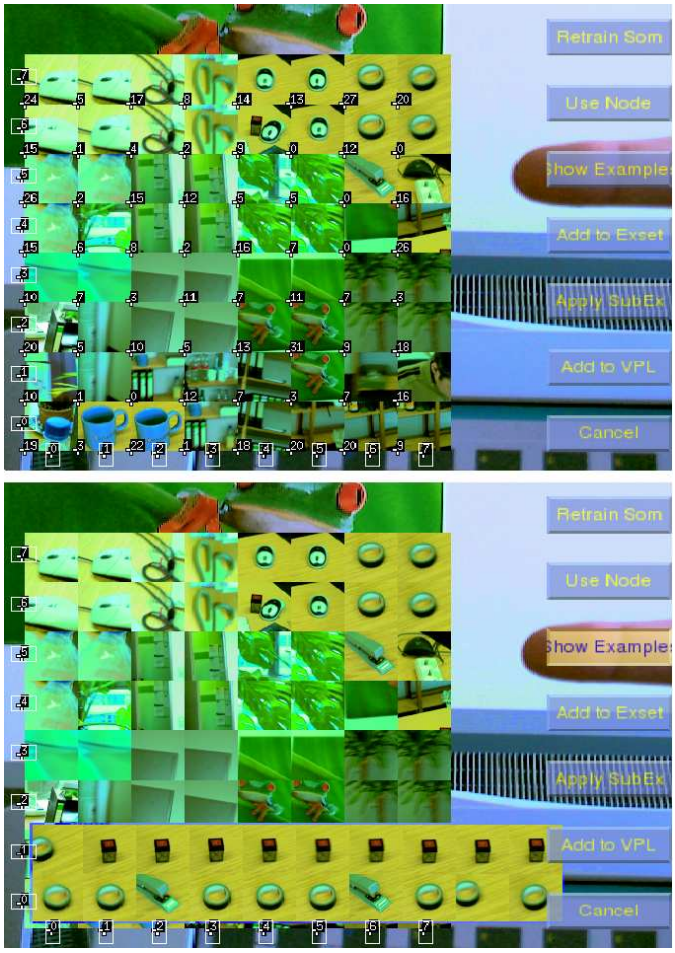

Fig. 8. Top: Displayed best match examples of a SOM trained with 15000 steps with equal weighting of color and edge features. The digits in the edge of each example present the number of image patches matched on this node. The displayed SOM is overlayed on the image stream while the user works in the office. In the background the poster of the frog also projected onto the SOM can be seen. The user just selected the bottom "Show Example" by the fingertip. Bottom: Displayed examples matched on the selected node (7,7).

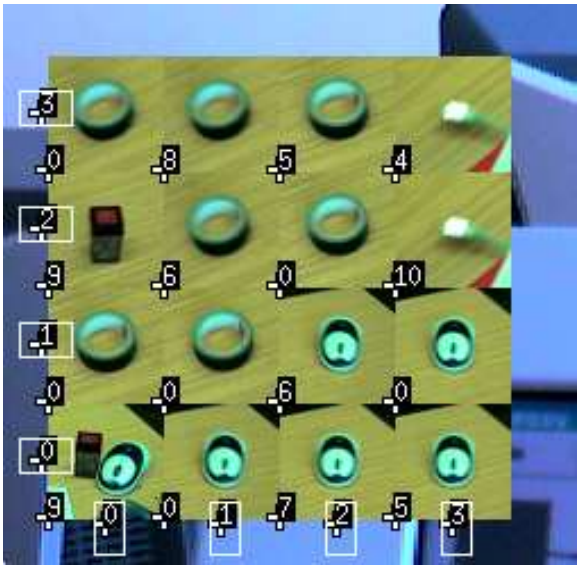

Fig. 9. 5×5 SOM picture retrained on best match examples of four nodes of Fig. 8 depicted in the display. Already in the second step the objects are separated by training the subset with higher weight of the edge feature.

VII. Example from an office environment

To illustrate the functionality for real imagery, a user has been moving about with the AR-gear in an office for five minutes, sampling 670 patches from 220 frames which have passed the motion filter. The office contains a desk, lamps, plants, posters etc. Fig. 8, top, shows the corresponding SOM (first training). E.g., bright objects without distinctive edges are mapped onto nodes in upper left region. Green patches of a poster of a frog are mapped to the middle right, adjacent to image patches displaying parts of plants. The lower middle region of the SOM depicts parts of shelves, including folders, books etc. dominated by edge features. Due to the dominant color of the desk, the upper right part shows different objects lying on the desk.

Since the objects on the desk are of interest to the user, he displays all image patches mapped onto node (7,7) (Fig. 8, bottom). These patches are now shown overlaid on the SOM, highlighted in the display by a blue frame. The image patches mapped onto this node now obtain the same group ID. To improve discrimination of the different objects of this group, edge features are supposed to be more discriminative than color features. Therefore, the user shifts the feature weighting in favor of the edge features and trains a new SOM on this subset. The resulting SOM is depicted in Fig. 9. Already in this second step, the objects are completely spread out onto different nodes and can be easily labeled.

VIII. Conclusion and Outlook

We have presented an augmented reality system capable of acquiring image data for online object learning in a natural and easy to use way. Training images are acquired simply by walking about, modules for focus-of-attention select the most relevant patches. Subsequently, the sampled patches can be clustered, visualized and labeled using SOMs in iterative steps. Using two qualitatively different types of features between which the user can adjust weighting, the effort for labeling huge numbers of training data is small enough for use in a mobile system. If the acquired database covers a sufficient variety of views, objects can be recognized in arbitrary pose.

The most promising line of future development appears to be improving the self-organization capabilities of the system. So far, the approach works semi-automatic, since the system forms categories based on low-level features, which hopefully correspond to real world semantics. But wherever this assumption does not apply, the user has to intervene. In future work, we are planning a *supervising module*, which learns from the users interventions. This module should learn to know (*i*) typical cases where low level similarities or dissimilarities are misleading, and (*ii*) the corrections of the user. Thus, it should be possible to learn an additional distance measure, which can gradually improve the up to now purely low-level distance measure.

Acknowledgment

We would like to thank Stefanie Schwassmann for her assistance in the implementation of the system. — This work was supported within the project VAMPIRE, which is part of the IST program (IST-2001-3401).

References

[Bekel et al., 2005] Bekel, H., Heidemann, G., and Ritter, H. (2005). SOM Based Image Data Structuring in an Augmented Reality Scenario. In *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Canada.

[Deselaers et al., 2004] Deselaers, T., Keysers, D., and Ney, H. (2004). Features for image retrieval: A quantitative comparison. In *DAGM-Symposium*, pages 228–236.

[Eidenberger, 2003] Eidenberger, H. (2003). How good are visual MPEG–7 features. In *Proceedings SPIE Visual Communications and Image PRocessing Conference*, pages 5150:476–488, Lugano.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. In *Proc. 4th Alvey Vision Conf.*, pages 147–151.

[Heidemann, 2004] Heidemann, G. (2004). Focus-of-Attention from Local Color Symmetries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(7):817–830.

[Heidemann et al., 2004a] Heidemann, G., Bax, I., Bekel, H., Bauckhage, C., Wachsmuth, S., Fink, G., Pinz, A., Ritter, H., and Sagerer, G. (2004a). Multimodal interaction in an augmented reality scenario. In *Proc. Int'l Conf. Multimodal Interfaces ICMI 2004*, pages 53–60. ACM Press.

[Heidemann et al., 2004b] Heidemann, G., Bekel, H., Bax, I., and Ritter, H. (2004b). Interactive Online Learning. In *Proc. PRIA 2004*, pages 44–48.

[Koenderink and van Doorn, 1979] Koenderink, J. J. and van Doorn, A. J. (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216.

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biol. Cybernetics*, 43:59–69.

[Kohonen, 1995] Kohonen, T. (1995). *Self-Organizing Maps*. Springer Verlag.

[Laaksonen et al., 2002] Laaksonen, J., Koskela, M., and Oja, E. (2002). Picsom–self-organizing image retrieval with mpeg-7 content descriptors. *IEEE Transactions on Neural Networks*, 13(4):841–853.

[Manjunath et al., 2001] Manjunath, B. S., Ohm, J.-R., Vasudevan, V. V., and Yamada, A. (2001). Color and Texture Descriptors. *IEEE Trans. on Circuits and Systems for Video Technology*, 11(6):703–715.

[Martinetz and Schulten, 1994] Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Networks*, 7(3):507–522.

[Mel, 1997] Mel, B. W. (1997). SEEMORE: Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation*, 9:777–804.

[MPEG-7, 2001] MPEG-7 (2001). Overview of the MPEG-7 standard (version 5.0). ISO/IEC JTC1/SC29/WG11 N4031.

[Murase and Nayar, 1995] Murase, H. and Nayar, S. K. (1995). Visual Learning and Recognition of 3-D Objects from Appearance. *Int'l J. of Computer Vision*, 14:5–24.

[Nene et al., 1996] Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia Object Image Library: COIL-100. Technical Report CUCS-006-96, Dept. Computer Science, Columbia Univ.

[Oja et al., 1999] Oja, M., Kaski, S., and Kohonen, T. (1999). Bibliography of self-organizing map (som) papers: 1998-2001 addendum.

[Park et al., 2000] Park, D. K., Jeon, Y. S., and Won, C. S. (2000). Efficient use of local edge histogram descriptor. In *MULTIMEDIA '00: Proceedings of the 2000 ACM workshops on Multimedia*, pages 51–54, New York, NY, USA. ACM Press.

[Sikora, 2001] Sikora, T. (2001). The mpeg-7 visual standard for content description-an overview. *IEEE Trans. Circuits Syst. Video Techn.*, 11(6):696–702.

[Somervuo and Kohonen, 1999] Somervuo, P. and Kohonen, T. (1999). Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159.

[Tokutaka et al., 1999] Tokutaka, H., Yoshihara, K., Fujimura, K., Iwamoto, K., and Obu-Cann, K. (1999). Application of self-organizing maps (som) to auger electron spectroscopy (aes). *Surface and Interface Analysis*, 27(8):783–788.

[Treue, 2001] Treue, S. (2001). Neural correlates of attention in primate visual cortex. *Trends in Neurosciences*, 24(5):295–300.

[Vesanto, 1999] Vesanto, J. (1999). Som-based data visualization methods. *Intelligent Data Analysis*, 3(2):111–126.

[Villmann et al., 2003] Villmann, T., Merenyi, E., and Hammer, B. (2003). Neural maps in remote sensing image analysis.