

Incremental GRLVQ: Learning Relevant Features for 3D Object Recognition

Tim C. Kietzmann*, Sascha Lange, Martin Riedmiller

*Institute of Computer Science, Institute of Cognitive Science
University of Osnabrück, D-49069 Osnabrück¹*

Abstract

We present a new variant of Generalized Learning Vector Quantization (GRLVQ) in a computer vision scenario. A version with incrementally added prototypes is used for the non-trivial case of high-dimensional object recognition. Training is based upon a generic set of standard visual features, the learned input weights are used for iterative feature pruning. Thus, prototypes and input space are altered simultaneously, leading to very sparse and task-specific representations. The effectiveness of the approach and the combination of the incremental variant together with pruning was tested on the Coil100 database. It exhibits excellent performance with regard to codebook size, feature selection and recognition accuracy.

Key words: object recognition, relevance learning, feature selection, incremental learning vector quantization, adaptive metric

1 Introduction

The task of object recognition and automatic learning of object representations has been widely addressed in the computer vision literature. However, one of the remaining and most prominent issues is the problem of high dimensionality. As a result, dealing with visual data poses great challenges to machine learning techniques, and the reduction of input space through feature selection can be seen as one of the keys to more efficient and reliable object recognition.

* Corresponding author.

Email addresses: tkietzma@uos.de (Tim C. Kietzmann), salange@uos.de (Sascha Lange), martin.riedmiller@uos.de (Martin Riedmiller).

¹ This work was partly granted by DFG-SPP 1125.

In general, the optimal selection of visual features is highly task- and data-dependent. Whereas some measurements might be used successfully in one situation or for one data set, it can happen that they might be useless in other conditions or tasks, even when dealing with the same objects. Accordingly, the choice of an appropriate feature space is very extensive. An algorithm capable of automatic and task-specific model and feature learning is needed in order to overcome the problems described.

In the area of machine learning, Generalized Relevance Learning Vector Quantization (GRLVQ) has been shown to improve performance and stability of the learning process [13]. As demonstrated by Hammer and Villmann [13], the approach is very well capable of handling artificial data sets including the Iris and Satellite data from the UCI repository. Moreover, the approach was successfully used for rule extractions [9] and data prediction in time series [44]. In addition to the promising learning performance, it is its capability of relevance learning which makes it especially appealing. The mechanism is able to give higher weights to important input dimensions and lower weights to unimportant ones. Whereas the latter idea was first introduced within the RELIEF-framework, proposed by Kira and Rendell [19], which was later extended by Kononenko [24], GRLVQ is able to adjust relevances during the learning process and is able to deal with multiple-class problems directly.

As in [44], we use the resulting relevances to completely prune the weakest dimensions during the learning process. This is especially valuable from an efficiency point of view and thus for real-time vision systems because feature extractions and computations induce high costs with regard to time and memory consumption. Thus, every feature which does not have to be calculated is highly beneficial.

The number of clusters formed by the training data and thus the number of codebook vectors needed changes with varying dimensionality of input space. Based on this and the fact that the selection of features is done automatically, it becomes clear that an optimal manual selection of the size of the codebook, i.e. the number of prototypes for every class, is not possible. In order to overcome this problem, we applied an incremental version of GRLVQ (iGRLVQ), which enables the system to automatically select this parameter for each class. This is done by starting with only one prototype for each object and then adding new representations for the instances which repeatedly cause errors. Because of this on-demand recruitment, comparably sparse models are created. Another effect of iGRLVQ is that the amount of domain knowledge which has to be provided by manual tuning of parameters is reduced since the size of the codebook is automatically selected online and in a task-specific fashion.

Being equipped with a mechanism with the ability of online feature and code-

book size selection such as iGRLVQ, the remaining question was how to define the initial setup of the feature space. While this important decision is normally done externally and prior to learning, our approach is to use a generic, non-specialized set of features. Because of the great variety, the initial number and thus dimensionality of the input space provided is assumed to be very high at the start. As a solution, we let the relevance learning mechanism decide on the most important dimensions and thus reduce the number of features needed during learning.

Taken together with the aforementioned problems in computer vision, the synthesis of model and relevance learning is an ideal solution. Integrating the prototype-based incremental learning mechanism with a generic set of visual features, coupled with the ability of online feature learning, leads to a very effective object recognition architecture. Consequently, our system is called Feature and Incremental Learning of Objects Utility (FILOU).

The main contribution of this work is a solution to an interesting and non-trivial learning task, namely automatic feature selection for visual object recognition, based on GRLVQ. We integrate the current state of research on the various subproblems into an optimized system. By testing it on multiple benchmarking problems, we prove that it forms an excellent solution in several respects. Moreover, we argue for the special importance of the combination of incremental methods together with iterative input pruning and provide empirical evidence. Results are presented for recognition performance and feature selection capabilities.

In the course of this paper, we will first discuss related work. Then, the proposed object recognition architecture FILOU is introduced and the underlying learning mechanism is described in more detail. Finally, the performance of the approach and the relevance learning, which were extensively tested on the COIL100 database [33], will be presented and discussed.

2 Related Work

Among the first authors to employ neural networks for visual object recognition were Poggio and Edelman [37], who used Generalized Radial Basis Function Networks (GRBFs) in order to represent object models. One of the main assumptions underlying this approach is that the 3D structure of an object could be specified by interpolating between stored 2D views of the respective item. As a result, the learned prototypes could be seen as representing the views of the object. In addition to being psychophysically [4,45] and physiologically [36] relevant, this basic assumption led to a great variety of approaches for 3D object recognition [39,42], including the current work.

Still, Poggio and Edelman dealt only with artificial data in the form of wire-frame 3D objects and a comparatively low complexity of only six dimensions. While keeping the main idea of view-based recognition, we are dealing with real-world objects and higher dimensionality, which considerably increases the task's complexity.

In the domain of computer vision with object recognition and content-based image retrieval in particular, the use of case-based reasoning (CBR) methods such as k-NN [5] and its variants is very common [6,25,28,35,43,49,50]. However, most of these approaches require storing all training patterns or at least a great number of them in order to achieve good performance [47]. This is especially problematic in the case of high-dimensional input spaces and online learning over extended periods of time. A solution to this problem is given by prototype-based algorithms, such as LVQ and its variants, because the number of prototypes needed is usually considerably smaller than the size of the original training set. Consequently, these methods are very appealing for efficiency reasons with respect to memory consumption and computational demands. For the task of object recognition, one of these methods was successfully applied in [21]. In that particular architecture, the standard learning vector quantization procedure (LVQ) was extended to better adapt to the task of online learning. This was achieved by applying an incremental mechanism (iLVQ) which enabled the system to dynamically add prototypes to objects which repeatedly caused misclassifications. Comparable approaches have also been proposed in combination with neural gas [7]. Concerning the input space, Kirstein et al. chose a biologically motivated approach resembling the ventral pathway of the human visual system. In their work, the input space was kept stable throughout the learning process and only the object prototypes were changed by the learning procedure. In our work, however, we address the change of feature space in addition to the prototypes. These two different ways of adjustments are applied in order to allow our system to achieve more dynamic and task-specific representations.

A number of methods for reducing the dimensionality of a feature set, also known as feature subset selection (FSS), have been published. Here, broadly three categories exist: Filter, wrapper and embedded methods [8]. Whereas filter methods, as for instance the one proposed in [23], are selecting features independently of the learning procedure, wrapper methods [22] treat the learning mechanism as a black box and use it to find the optimal feature subset. Although they are known to lead to promising results, wrapper methods are computationally very complex. Another disadvantage of the first two categories is that both have to be done prior to the actual learning procedure. Because of the high variability of visual data, it would be sensible to be able to select features directly during the learning process in order to be better able to account for changes in the data. Since embedded methods incorporate the selection as part of the training, they make better use of the available data

and do not require retraining as in the case of wrapper methods.

When dealing with artificial neural networks such as Multilayer Perceptrons (MLPs), techniques exist to reduce the number of parameters by dropping unimportant weights [27]. Additionally, Principal Component Analysis (PCA) [17] is a standard technique often used to prepare the input for MLPs [18,26,30]. Variants of PCA exist, which completely prune input dimensions (for a thorough discussion, please refer to chapter 6 of [18]), but may lead to very poor results, especially in the case of highly correlated dimensions in the original input space. The main problem of PCA is that it is not an integrated part of the actual learning process. Moreover, it does not consider the class labels but only variances in the input data. Despite these problems, PCA is still a very popular technique in computer vision applications. This is due to the fact that it can be applied directly to the image data and thereby *constructs* a set of features, also known as Eigenimages, instead of using PCA for feature *selection* only [31,20,48]. Performance of an up to date version of this method [14] will later be compared to our results.

Additionally, statistical methods exist which achieve very good results. Among the most prominent ones are the LASSO method [46] and Automatic Relevance Determination (ARD)[29,32]. Originally introduced in the context of least-square regression, the LASSO has the disadvantage that the number of selected variables is limited by the number of training instances [51]. In ARD, pruning is accomplished by using several weight decay constants, one for each input dimension. The general idea is now that the resulting hyperparameters or decay rates of unnecessary input are automatically inferred to be large. This way, relevances can be determined and used for dimension reduction. However, ARD can lead to overfitting [38]. For a more detailed description and overview of the available feature selection methods, please refer to [16] and [8].

3 FILOU - Feature and Incremental Learning of Objects Utility

In the following section, we will introduce a new architecture capable of recognizing 3D objects. In FILOU, each of the classes to be learned is represented by a number of adjustable prototypes or codebook vectors. Moreover, the input space initially used is formed by a generic set which offers a wide variety of standard visual features. On top of these features lies a selection mechanism for detecting relevant input dimensions. The codebook adjustments and feature selection are achieved automatically during learning by applying iGRLVQ. This extension is used for online feature pruning and incremental prototype recruitment. Because it is capable of dealing with these two main issues, it can be seen as the heart of FILOU. We will first explain the basic GRLVQ algo-

rithm before turning to a description of the use of prototype-based methods in the area of computer vision. Afterwards, details concerning the underlying feature space and the pruning mechanism will be given. Finally, we will motivate and describe iGRLVQ.

3.1 Prototype-Based Learning: GRLVQ

Extending the basic principles of LVQ, Sato and Yamada [40] introduced the generalized LVQ (GLVQ) algorithm, which is defined by a cost function including LVQ 2.1 as a special case. Thus, being based on stochastic gradient descent (the corresponding proof can be found in [11]), this method could further improve stability of the learning process. Using the idea of relevance terms, Hammer et al. [11] extended the algorithm to generalized relevance LVQ (GRLVQ) by incorporating an update rule that enables the system to assign a measure of importance to the input dimensions through relevances.

The algorithm can be described as follows. The data to be learned is given by a labeled training set in the form of feature vectors $x = \{(x^i, y^i) \in \mathbb{R}^n \times \{1, \dots, C\} \mid i = 1, \dots, m\}$, where each training instance belongs to one of C classes. Furthermore, let each vector x be of the form $x = (x_1, \dots, x_n)$; $x \in \mathbb{R}^n$. For every class in the training set, there exist vectors in \mathbb{R}^n corresponding to the prototypes of the respective class. Altogether, they form the codebook of the learning algorithm, i.e the codebook W is given by the prototype vectors (w^1, \dots, w^M) and the associated class labels $c^i \in \{1, \dots, C\}$. The underlying goal of the learning procedure is now to adjust the codebook vectors w^i of each class such that they represent it as accurately as possible, despite the closeness of data vectors belonging to different classes. By convention, the nearest prototype of the same class as a training vector x will be called positive and the nearest prototype of a different class will be called negative in the following. Basic LVQ offers an update rule only for the nearest positive codebook vector. The idea of generalized methods is to adjust the nearest positive ($w^J = \operatorname{argmin}_{w_{P=J}^j} (\|x - w^j\|^2)$) and the nearest negative ($w^K = \operatorname{argmin}_{w_{P \neq J}^j} (\|x - w^j\|^2)$) prototype when being presented with a training vector i of class P , x^i . This is done by using the training vector to attract w^J and push away w^K , according to $w_{new}^J = w_{old}^J + \Delta w^J$ and $w_{new}^K = w_{old}^K - \Delta w^K$. The update elements are defined as

$$\begin{aligned} \Delta w^J &:= \epsilon \cdot \frac{d_K}{(d_J + d_K)^2} (x^i - w^J) \\ \Delta w^K &:= \epsilon \cdot \frac{d_J}{(d_J + d_K)^2} (x^i - w^K) \end{aligned} \tag{1}$$

respectively. The distances $d_J = \|x, w^J\|$ and d_K are calculated according to the standard Euclidian metric. The resulting error measure is able to push the learned decision borders close to the Bayesian and thus optimal ones.

However, the resulting performance relies crucially on the commonly used Euclidian metric and hence relies on the fact that it is suitable for the respective learning task. Thus, the data must be preprocessed and scaled such that the input dimensions have approximately the same magnitude for the classification. In addition to the obvious problem of appropriateness of Euclidian space, superfluous data dimensions slow down training. Therefore, relevance learning uses input weights $\lambda = (\lambda_1, \dots, \lambda_n)$ where each λ_i is responsible for one dimension, in order to allow for automatic and variable scaling of the input space. Integrating λ into the standard Euclidean distance metric results in:

$$d_J = \|x - w^J\|_{\lambda}^2 = \sum_{i=1}^n \lambda_i (x_i - w_i)^2 \quad (2)$$

During learning, the relevance terms are adjusted according to a learning procedure which follows the main update principles used for prototype adjustments. Following [12], the resulting update rule for the relevance terms is defined as:

$$\lambda_m := \lambda_{m-1} - \epsilon_1 \cdot \left(\frac{d_K}{(d_J + d_K)^2} (x_m^i - w_m^J)^2 - \frac{d_J}{(d_J + d_K)^2} (x_m^i - w_m^K)^2 \right) \quad (3)$$

λ is normalized after each update step to ensure $\|\lambda\| = 1$. In order to keep the λ values positive, negative values and relevances for constant input dimensions are set to zero. The update rule can be interpreted as follows: In the case of correct classification, the update results in an increase in the weight terms standing for dimensions in which the training data is close to the positive prototype, whereas relevances for terms with greater distance are decreased. In contrast, in the case of false classification the more distant dimensions are increased and the closer ones weakened. Thus, this mechanism facilitates dimensions which contributed to the right classification and which did not contribute to a false one. A more detailed derivation of the above-mentioned formulas and procedures can be found in [10]. Again, the use of the generalized method proved to enhance convergence speed and stability compared to mere relevance learning [13].

3.2 Prototype-Based Methods for Object Recognition

The codebook vectors of the learning mechanism can be regarded as representing 2D views. This way, each prototype corresponds to a specific perspective of

the underlying object. All prototypes of one type taken together consequently form the overall representation.

When dealing with images, visual features have to be extracted in order to form the feature vectors of the training data. The feature extractions can therefore be seen as a mapping from the raw pixel space to a lower-dimensional feature space. Having transformed the images to feature vectors, a learning procedure is applied in order to find appropriate prototypes for later recognition. Generalized algorithms are known to improve stability of the learning process especially in the case of contradictory training patterns. Because the latter is particularly prominent in visual data and thus a relevant issue, the use of generalized learning is of great importance. Having decided on a particular type of learning mechanism, the underlying feature space remains to be defined, which will be done in the following section.

3.2.1 Generic Feature Set

In order to be able to use standard prototype-based algorithms for object recognition, which need a fixed dimensionality of input space, we decided to use global object features. Unfortunately, extractions of visual features are computationally very expensive, which is why the dimensionality of input space should be kept as small as possible, while maintaining high performance. Consequently, selecting the right amount and types of features is decisive for the later performance of the system, as well as highly task-specific and normally done by a human expert. Our approach is quite different. Initially offering a whole variety of standard measurements in the form of a generic feature set with 137 dimensions, we let the system automatically decide on relevant input dimensions and prune irrelevant ones. This does not only reduce the amount of external knowledge which has to be put into the system, but also leads to highly task-specific solutions. The initial feature space can broadly be put into two categories, i.e. appearance and shape selective.

The most common way of dealing with the first category is via color and luminance information. In detail, the resulting input space included a 32-dimensional luminance histogram, and 64 dimensions of YUV color histograms. Shape information was included by calculating the area of the object, its centroid, perimeter, eccentricity, circularity, compactness, and maximum and orthogonal diameter. To provide even more information, the Hu set of image moments [15] was used as well. These seven dimensions are calculated as particularly weighted averages of the object's pixel intensities and have the special property of being invariant under translation, scale and rotation. For instance, the first measurement can be interpreted as the moment of inertia around the centroid of the image. Finally, Gabor wavelets with three scales and four orientations were extracted. This type of feature can be seen as de-

tecting specially oriented and scaled bars of intensity in the image. Most often, they are used in biologically motivated approaches, which is due to the fact that simple and complex cells in area V1 of the human cortex are also selective to specially oriented bars of light. Since the resulting dimensionality is given by $2 \times \text{scales} \times \text{orientations}$, the input space included 24 dimensions corresponding to this feature. Because many of the extractions rely on a segmentation of the image to separate the object from the image background, we applied region-growing as a preprocessing step, as proposed in [1]. Although this selection of features is expected to be able to deal with various problems, it is of course possible to modify the set by integrating further features or changing existing ones. Moreover, it is possible to exchange the proposed set completely, such that every author can use his/her favorite features.

Altogether, the final feature space had 137 dimensions. Although GRLVQ was successfully used in high-dimensional input spaces before [3,10], the complex domain of visual data has not yet been examined.

3.2.2 Automatic Pruning

Having described the underlying feature space, we will now turn to the problem of feature selection, which is widely addressed in the machine learning literature. This aspect is especially valuable in the case of high-dimensional visual data where relevant dimensions might differ from task to task. Because of this variability, it is very hard even for humans to assess which measurements will be useful for a particular data set under a particular condition. Pruning allows for increased efficiency since the dimensionality of feature space as well as computational costs for future learning and extractions are reduced. The task of feature reduction is very extensive in standard techniques such as neural networks. In addition to the statistical approaches such as the ones described earlier, the connection weights of the input dimensions can be checked after training and then the low-weighted ones are pruned. Afterwards, the network has to be retrained [2]. This technique of assessing relevant dimensions after training and then relearning has the clear disadvantage of being very slow [27]. One of the main advantages of using relevance terms as the basis for pruning is that it can be done online, making it an integrated part of the overall learning process. As a result, feature selection and model learning cannot be seen as two distinct processes in the current approach. As will become more evident later, the algorithm is able to deal with online pruning of huge parts of the input space without the need of explicit retraining.

A further step in addition to only diminishing the influence of unimportant dimensions is to prune the weakest dimensions, i.e. the ones with the smallest λ -values, in order to completely exclude them from the following learning and recognition process and even more importantly from future calculations.

Having been successfully applied to time series by Hammer et al. [44], this can then be seen as real feature selection because only the remaining dimensions of input space have to be calculated in future. Instead of pruning all unneeded dimensions at once, we iteratively remove small parts of the feature space in every learning step, allowing the system to better adapt and reorganize the remaining relevances and prototypes. In addition to improved learning behavior, this method allows for an automatic stopping criterion, i.e. using the accuracy as measure, pruning is stopped when the loss on the validation data exceeds a certain threshold. This way, the optimal number of dimensions can also be found automatically.

3.2.3 *iGRLVQ*

In practice, one of the main issues of prototype-based algorithms is the selection of an appropriate number of prototypes which must be subject to extensive testing or human expertise. Standard methods normally use the same amount for every class. At times, this can lead to a codebook which is too large, especially in the case of less complex classes. Furthermore, the complexity of the task differs with the dimensionality of the input because the number of clusters is expected to be smaller in lower-dimensional input spaces. Consequently, the question regarding the appropriate amount of prototypes for every class becomes even more pronounced, if not impossible, in the light of relevance learning and automatic dimension pruning. The optimal number is highly dependent on the respective input features, which cannot be known a priori in an automatic selection process. In order to avoid these problems and to decrease the amount of external knowledge, we extended GRLVQ with an incremental part proposed in [21,3]. The key idea is to add prototypes to classes which cause subsequent errors during training. If, for a processed training instance, d^J is greater than d^K , the system would have misclassified the vector since the negative prototype would have been closer to it than the positive one. Whenever this happens, an error term g is increased and the corresponding erroneous feature vector is stored together with the corresponding d^K . If, at the end of the learning step, the number of misclassifications exceeds a threshold $gmax$, a prototype is added for each misclassified class. Here, we apply the idea of Kirstein et al. [21], who proposed that out of all misclassified feature vectors of the corresponding class, the one closest to the foreign codebook vector is used as a sample for the newly added prototype. The expected number of errors naturally varies with the size of the training set. Small sets normally cause less insertions of prototypes than bigger ones. For this reason, we set $gmax$ to two percent of the size of the training set.

Formally speaking, the incremental part can be seen as follows. After a misclassification of a vector x^i belonging to class c^p , x^i and the distance d^K are stored in the class-specific error set S_p . If $g \geq gmax$, the codebook vector with

minimal d^K is added for each $S_p \neq \emptyset$ and g is reset. The reasoning behind the selection of the training instance, which is closest to its negative codebook, as new prototype is that the newly added vectors should be near the decision boundary. g is only set to zero after the addition of new prototypes and not in every learning step. This is done because if we reset g in every iteration, cases in which only less than $gmax$ misclassifications occur in one epoch would not lead to any further additions. With the accumulation of errors through multiple epochs, the system is still able to increase the size of the codebook, even if only small amounts of errors remain. Nevertheless, additions become less often and the number of pure learning steps increases in later epochs. A pseudo-code of one iteration of the learning procedure is given in Algorithm 1. Generally, classes which cause the most errors and which are thus expected to be more complex are given more prototypes on demand, whereas simple classes will be able to be covered by smaller numbers of prototypes.

The current chapter introduced our object recognition system FILOU, which is a prototype- or view-based approach. It uses a generic set of visual features together with an automatic pruning algorithm. Because feature selection relies on the learned relevance terms of iGRLVQ, the system is able to reduce the input space as part of its learning process, speeding up future computations. Finally, an incremental mechanism is applied in order to be able to deal with varying numbers of clusters and to create sparse object models.

Algorithm 1 One iteration of iGRLVQ

```

for all  $(x^i, y^i) \in X$  do
  Find closest positive ( $w^J$ ) and negative ( $w^K$ ) codebook vector
  if  $d_J > d_K$  &  $epoch \geq incr\_start$  then
     $g := g + 1$  and store  $x_P^i$  and  $d_P^K$  in  $S_P$ 
  end if
  Adjust both codebook vectors according to equation 1
  if  $epoch \geq relevance\_start$  then
    adjust relevance terms acc. to eq.3
  end if
  if  $epoch \geq incr\_start$  &  $g \geq gmax$  then
    Find candidate error-vector for every  $S_P \neq \emptyset$ 
    Add candidate to codebook
    Reset  $g$  ( $g=0$ )
  end if
end for

```

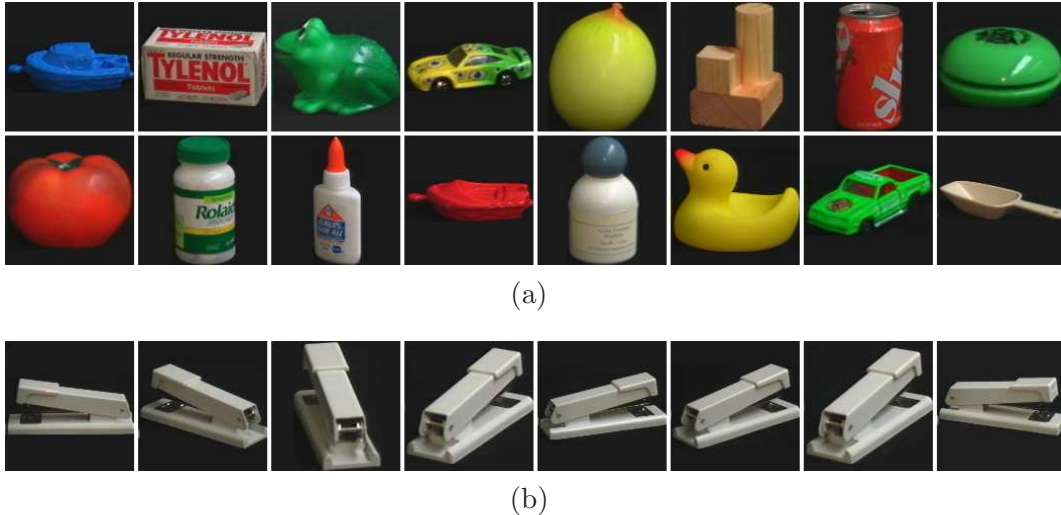


Fig. 1. (a) some examples from the COIL100 database (b) a sequence of images

4 Experimental Results

In order to assess the recognition performance of FILOU, several experiments were conducted using the COIL100 database. Figure 1 shows some of its stimuli. The data set includes 360° image sequences of 100 objects. In each subsequent picture of an object, it is turned by 5° such that there are 72 images for every object in total. The resulting task was thus to recognize 3D objects from various perspectives based on their corresponding 2D images. The most effective values for the learning rates for prototype and relevance updates ϵ and ϵ_1 were found experimentally and decrease during training in order to guarantee convergence.

In detail, we tested for accuracy, effects of dimension pruning and relevance learning. For comparability reasons, we applied the same method for selecting the training and test set as in [34,41] to assess recognition performance. For the experiments, 4 (8, 18 and 36) views out of the total 72 were used for training, and performance was tested on the remaining images forming the test set. These training instances were selected to best cover the 360°. For instance, the 18 training images were selected to have a distance of 20° between each other. The resulting training set included 1800 data points and the final performance was tested on the remaining 5400 vectors.

Since it was one of the main interests of the current work to assess how learning and accuracy would change after pruning the least relevant dimensions, we iteratively excluded all but 50, 20, 10 dimensions directly during learning in a second experiment.

Finally, in order to specifically test for the effects of relevance learning, two sets of objects were formed. The first included all green objects in the database

and the second contained objects with similar shape but different color. This enabled us to check in detail whether the automatic feature selection made sense from a semantic point of view because the feature types to be neglected could obviously be determined.

4.1 Recognition Performance

4.1.1 First Experiments: GRLVQ vs. iGRLVQ

First, we compared the performance of the incremental version to the standard algorithm. Both approaches were tested with learning rates varying from 0.001 to 0.98, and the best value for each method was used for the final experiments ($\epsilon = 0.01$ with *decay* = 0.7 and $\epsilon_1 = 0.005$ with *decay* = 0.88 for both methods). We assessed the recognition abilities when using either method as learning mechanism. Furthermore, we examined the effects of pruning, results are shown in Table 1. Because generalization abilities are of higher interest, all values given will correspond to the accuracies on the test set. As one might notice, the resulting performance of GRLVQ is highly dependent on the number of pruned features and the number of prototypes allocated by each class. With decreasing dimensionality, the performance dropped dramatically. The system was not able to compensate the reduction of input space.

method	# prototypes	# of dimensions kept			
		137	50	20	10
GRLVQ	3	94.7	93.9	91.3	85.0
GRLVQ	6	96.6	96.1	93.8	88.5
GRLVQ	9	97.4	96.8	95.2	92.0
GRLVQ	12	97.9	97.5	95.4	92.9
iGRLVQ	auto	97.9 (2.8)	97.7 (3.1)	96.3 (3.9)	93.8 (5.2)

Table 1

Resulting accuracy of the system when using either standard GRLVQ or iGRLVQ as learning method. Training was based on 18 views and the presented results are based on the accuracy on the remaining 54 object views used as test set. Shown are performances with the corresponding number of prototypes after pruning 0, 87, 117 and 127 dimensions of input space. Notice that iGRLVQ automatically selects the number of prototypes: The resulting average values are given in parentheses. In the case of GRLVQ, recognition performance was comparable to the one achieved by iGRLVQ only after adding up to 12 prototypes to the codebook, which corresponds to nearly the whole training set. Adding even more prototypes could not further improve performance.

A solution to these problems was found with iGRLVQ, which enabled the

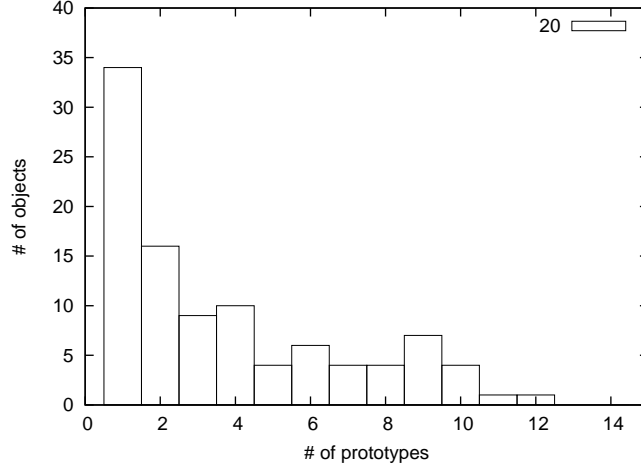


Fig. 2. This distribution shows how many objects were assigned the different numbers of prototypes by iGRLVQ. Results are shown for the case of reducing the feature space to 20 dimensions. Most of the objects were simple enough to be covered by small amounts of prototypes, whereas only some more complex ones needed higher numbers of prototypes.

system to dynamically introduce new prototype vectors to the codebook. The drop in accuracy was less pronounced and the overall performance could be increased significantly.

In detail, when using iGRLVQ, complex objects tend to get more resources than simple ones, which could often be classified by a single codebook vector (see Fig. 2). In the case of the standard method, the needed number of prototypes given to each class had to be equal to the maximum number from iGRLVQ in order to achieve the same performance. In other words, because standard GRLVQ does not differentiate between classes with regard to the number of prototypes, all objects had to get the same amount as needed for the most complex one in order to achieve comparable results. This naturally resulted in a higher average. As can be seen in Table 1, iGRLVQ only needed 2.8 prototypes on average instead of 12, while achieving comparable performance. A comparison of the resulting codebooks after letting GRLVQ and iGRLVQ learn an easy and a complex object is shown in Figure 3. The graph was created using only the two most important input dimensions. As can easily be seen, the instances of the simple object form a clear cluster. As a result, iGRLVQ allocated only one prototype for this class, whereas GRLVQ used the same amount as for the other object. The more complex item, however, needs more representations in order to cover all training instances. As a result, iGRLVQ clearly outperforms GRLVQ because it is able to attain higher accuracy with a smaller amount of prototypes.

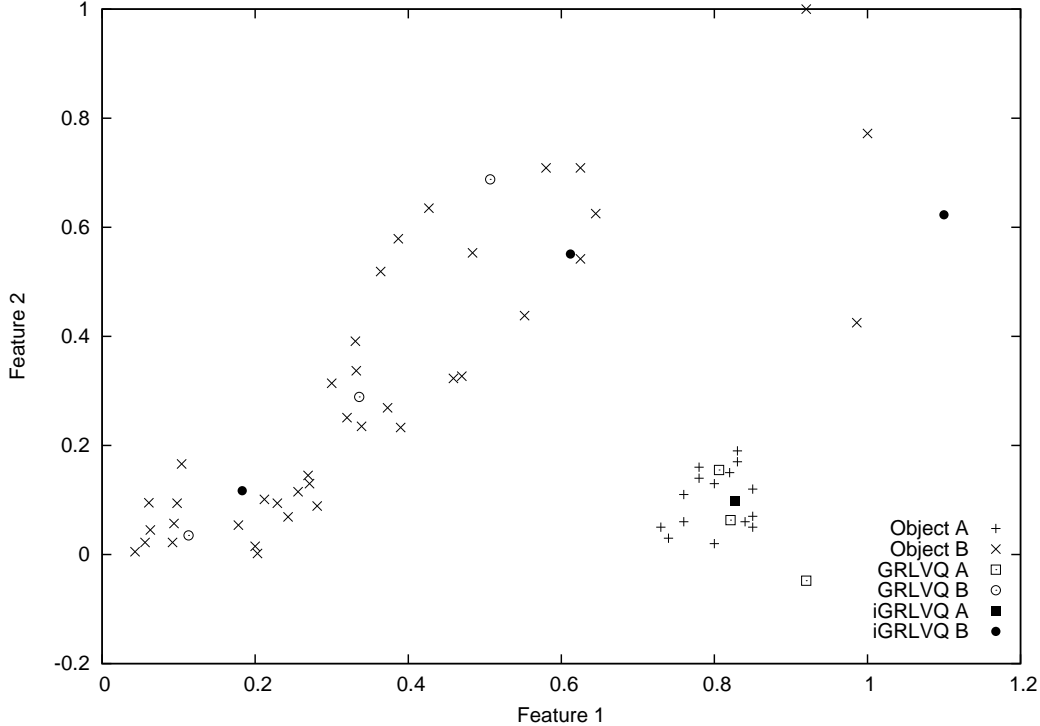


Fig. 3. Shown are the two most important dimensions of the training instances of a simple and a complex object together with the resulting codebook vectors of GRLVQ and iGRLVQ, respectively. Although one prototype suffices to represent the simple item, GRLVQ used three because it does not differentiate between objects with respect to numbers of prototypes. iGRLVQ, however, used only one representation for the simple and three for the complex case.

4.1.2 Recognition Performance of iGRLVQ

Figure 4 shows typical learning curves of iGRLVQ. The algorithm first started by adjusting only the currently stored prototypes without updating the relevance terms. After 30 epochs, the relevance updates began and could improve accuracy and stability of the learning process. However, only the addition of further codebook vectors to the erroneous classes after epoch 50 could further improve performance to achieve its final level. The recruitment of prototypes is highly task-specific. As shown in Figure 4(a), learning an easy task could be accomplished without recruiting additional prototypes.

The resulting accuracy of our approach and other comparable approaches on the test set when trained with different numbers of images are given in Table 2. Once more, it is important to consider that we used a generic set of standard features, whereas other approaches were mainly using handcrafted and task-specific features, which were specifically selected for the COIL100 database in order to optimize recognition performance. Here, no effort was put into the careful tuning of features because the system was able to do this automatically. Having ensured that the results of the architecture were absolutely competitive

to other state-of-the-art algorithms, we now turn to the the second central objective of the current work, i.e. a more detailed investigation of the effects of relevance learning and feature selection capabilities.

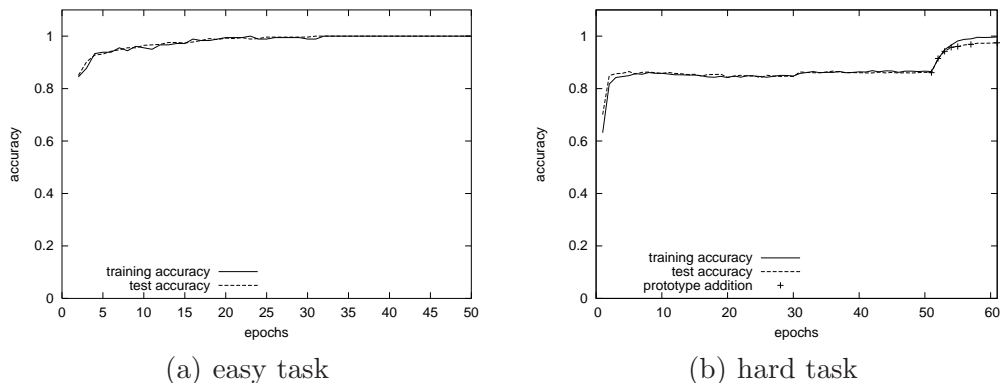


Fig. 4. Learning curves for easy and hard tasks. The crosses indicate additions of prototypes which were needed in the hard but not in the easy task.

Method	# training views			
	36	18	8	4
FILOU (137)	99.3%	97.9%	92.6%	85.4%
FILOU (50)	99.2%	97.7%	92.0%	82.9%
FILOU (20)	98.5%	96.3%	88.9%	76.3%
PCA & NN*	98.2%	96.5%	-	-
Spin-Glass MRF	-	96.8%	88.2%	69.4%
FILOU (10)	97.0%	93.8%	85.3%	69.2%
Linear SVM	-	91.3%	84.8%	78.5%
Nearest Neighbor	-	87.5%	79.5%	74.6%

Table 2

Comparison of generalization performance of various approaches using the COIL100 database. (* Results on 40 of the 100 objects using the first 12 PCAs). Results of the other approaches were taken from [34], the numbers in parentheses correspond to the resulting dimensionality of feature space after pruning.

4.2 Pruning Analysis

4.2.1 Performance

In order to test performance under the influence of pruning, various experiments were conducted with different values as target dimensionality for input space. The system was able to drastically reduce the feature space without

markedly degrading performance (see Table 1). For instance, we could reduce the dimensionality and thus computational costs to less than 1/6, while losing only about 1.6% in accuracy. For testing purposes, the system iteratively deleted features up to a predefined number, ignoring the aforementioned stopping criterion for now. Generally, the final loss in performance was greater with the increasing number of dropped dimensions.

As shown by the learning curves in Figure 5, the iterative deletion of unneeded input dimensions did not result in a significant drop in accuracy when performed down to 20 dimensions. However, when deleting all but 10 dimensions, the system was no longer able to directly recover and there was a temporary drop in accuracy of about 7%. Only the addition of new prototypes was able to restore performance. Interestingly, the resulting accuracy was comparable even when deleting all but the demanded number of features at once. In this case, accuracy always dropped and it took the system longer to recover. This period of worse performance does not exist in the iterative case.

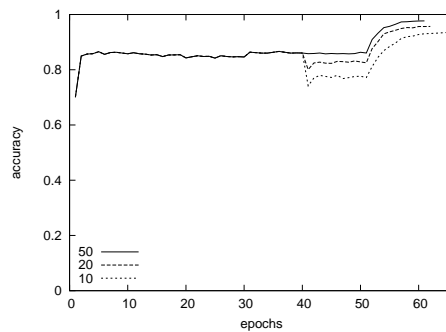


Fig. 5. Learning curves for pruning all but 10, 20 and 50 dimensions. In each iteration the four least relevant dimensions were deleted. There is no drop in accuracy when pruning input space down to 50 dimensions. Nevertheless, accuracy drops when keeping only 20 or even 10 dimensions. In these cases, only additions of further prototypes enabled the system to recover and achieve its final performance.

The average number of needed prototypes is increasing with a decreasing number of dimensions left. This can be seen as evidence for the fact that the number of clusters built by the training images changed with pruning. These exact effects cannot be known a priori, which provides even more support for the usage of the incremental method. The capability of allocating more resources on demand enabled the system to react to the need for more prototypes whenever the classes could not be discriminated sufficiently in the lower dimensional space. In other words, the use of the incremental method proved to be especially sensible when being applied in combination with online pruning.

4.2.2 Semantics

In addition to examining the changes in accuracy after dropping irrelevant dimensions, there are further interesting topics to be examined concerning relevance learning. Having demonstrated that performance is comparably stable to dimension reduction, we conducted further experiments in order to assess the semantics behind the selection. Here, the influence of the task in the form of the data presented in the training set and the resulting kind of selected features were investigated in more detail. For this reason, we tested two special cases. The first data set included objects with similar color but different shape. The second one used objects with similar shape and different color. Examples of the two sets are given in Figure 6.

The outcome of the first training set was expected to place emphasis on features selective to the shape of the object because color information would clearly fail in this special case. In the second case, resulting relevances should be the other way around. Because the objects differed only in color, the shape selective features should not help the discrimination as opposed to e.g. color-histograms, which should clearly form the most important features. Results of both tests are shown in Table 3. The system worked as expected, i.e. task-relevant features were emphasized and irrelevant features diminished.



Fig. 6. Instances of the two selected training sets. Objects in the top row belong to the same color condition, objects in the bottom belong to the same form/different color condition.

	C	P	D	Hu
similar color	0.041	0.082	0.167	0.22
similar shape	0.15	0.03	0.02	0.03

Table 3

The resulting λ -values on two specific tasks. The features shown are color histograms (C), perimeter (P), diameter (D) and Hu-Moments (Hu). The same color condition gave different class names to objects with similar color but different shape. The same shape condition gave objects with comparable shape but different color different class names. The resulting relevance values show that color information was found to be much more important in the same shape condition, whereas shape selective features were decisive in the same color condition.

4.3 Prototype Analysis

As described earlier, manually selecting an appropriate number of prototypes for every object class in the training set is very complicated without extensive testing prior to learning. As described, when taken together with the dynamics of relevance learning, this task becomes nearly impossible. Being able to let the system automatically deal with this issue was expected to improve efficiency and performance since the recruitment of new prototypes would be done during learning and would directly depend on the current training set and the complexity of the objects. Table 4 shows some examples of the COIL100 database together with the resulting number of prototypes used by our architecture. Because prototypes do not necessarily lie directly on data points and the abstract features used do not allow a reconstruction, the images shown do not directly resemble the prototype, but rather the image which caused its highest activity. As can readily be seen, a very simple object (e.g. onion) required only a single prototype, whereas with increasing complexity the needed number of prototypes increased as well. In the case of vaseline, the shape in the first and last image is comparably similar. However, the label differs significantly, which was the reason for an additional prototype. This behavior of the system clearly shows the effectiveness of the incremental method and its ability to form extremely sparse object representations directly dependent on the complexity of the underlying object and task.


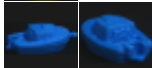
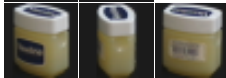

Object	# Prototypes	Images
Onion	1	
Boat	2	
Vaseline	3	
Hook	4	

Table 4

Examples of objects together with the resulting number of prototypes. Homogenous and thus simple objects needed only small amounts, whereas more complex objects had to be represented by more prototypes in order to achieve good recognition performance.

5 Discussion

FILOU has proven to be a reliable approach for the complex task of object recognition. We have clearly shown that beyond good recognition performance,

the incremental relevance learning could also be used as a reliable feature selection mechanism.

The ability to select relevant visual features is clearly one of the most important advantages of the system because computational costs can be reduced during learning without the necessity of relearning. As mentioned before, feature extractions are very time and memory consuming in the domain of computer vision. Thus, being able to reduce the number of extractions needed for future recognitions is very sensible from an efficiency point of view. This makes our approach particularly reasonable in autonomous robotics, where real-time performance is essentially necessary. Possible applications include rapid object detection and object tracking.

We could reduce the amount of knowledge and preliminary work being put into the system when using handcrafted features by applying a generic set of standard features. Out of this, the system could automatically select relevant elements during learning. In the case of back-propagation in neural nets, pruning low-weighted input dimensions after learning needs to be followed by a new learning process in order to let the system adapt to the new structure. By contrast, iGRLVQ is able to rely on the available structure, the adjustments of the codebook vectors are already mostly based on the dominant dimensions before the actual pruning takes place. This way, FILOU can drop irrelevant features and thus increase efficiency while keeping performance comparably stable.

Considering the amount of human expertise needed, our approach has further advantages. First of all, an a priori selection of the right amount of prototypes for every class is very tedious if not impossible because it is dependent on the object's complexity. As shown in Table 4, the number of prototypes needed increases with the complexity of the object represented. Additionally, the combination with relevance learning and feature pruning makes a manual identification of the right amount especially difficult because the numbers of prototypes needed also depend on the underlying feature space. However, both problems could be solved in an elegant manner by applying iGRLVQ.

The system's behavior is quite intuitive regarding the semantics behind the feature selection. It proved to be highly task-specific and made sense from a human point of view. Nevertheless, the relevance learning algorithm is not able to directly detect certain redundancies in the data. If, for instance, two dimensions of input space were equal, the resulting relevances would give both the same importance instead of deleting one of them. Being able to deal with these kinds of redundancies and interdependencies between the features is expected to further reduce the dimensionality of input space and thus to increase the efficiency of the overall system.

Summing up, FILOU was able to show excellent recognition performance while automatically selecting the number of prototypes for every class and relevant features out of a generic set. We described a new combination of variants of LVQ, which proved to be especially effective and synergetic when used in combination. Moreover, the resulting input space and codebook size were shown to be task-specific. This makes the current approach even more suitable when applied to a domain known to be highly dynamic, such as the area of computer vision. For these reasons, the current work can be regarded as a step in the direction of real, unattached machine learning, independent of the expertise of its human creator.

References

- [1] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647, 1994.
- [2] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc. New York, NY, USA, 1995.
- [3] T. Bojer, B. Hammer, and C. Koers. Monitoring technical systems with prototype based clustering. *European Symposium on Artificial Neural Networks*, pages 433–439, 2003.
- [4] H.H. Bülthoff and S. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proc Natl Acad Sci US A*, 89(1):60–64, 1992. 32.
- [5] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.
- [6] J.T. Favata and G. Srikantan. A multiple feature/resolution approach to handprinted digit and character recognition. *International Journal of Imaging Systems and Technology*, 7(4):304–311, 1996.
- [7] B. Fritzke. Growing cell structures - A self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.
- [8] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1157–1182, 2003. 44.
- [9] B. Hammer, A. Rechten, M. Strickert, and T. Villmann. Rule extraction from self-organizing networks. In *International Conference on Artificial Neural Networks*, 2002.
- [10] B. Hammer, M. Strickert, and T. Villmann. Prototype based recognition of splice sites. *Bioinformatics using Computational Intelligence Paradigms*, pages 25–56, 2005.

- [11] B. Hammer, M. Strickert, and T. Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, 21(1):21–44, 2005.
- [12] B. Hammer and T. Villmann. Estimating relevant input dimensions for self-organizing algorithms. *Advances in Self-Organizing Maps*, pages 173–180, 2001. 41.
- [13] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8):1059–1068, 2002. 40.
- [14] G. Heidemann, D. Lucke, and H. Ritter. A system for various visual classification tasks based on neural networks. *Proc. 15th Int’l Conf. on Pattern Recognition ICPR*, pages 9–12, 2000. 37.
- [15] M.K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IEEE Transactions on*, 8(2):179–187, 1962.
- [16] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994. 36.
- [17] I. T. Jolliffe. Principle component analysis. *Spring-Verlag, New York*, 1986.
- [18] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.
- [19] K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134, 1992.
- [20] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1):103–108, 1990.
- [21] S. KIRSTEIN, H. WERSING, and E. KORNER. Rapid online learning of objects in a biologically motivated recognition architecture. *27th Pattern Recognition Symposium DAGM*, pages 301–308, 2005. 4.
- [22] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
- [23] D. Koller and M. Sahami. Toward optimal feature selection. *International Conference on Machine Learning*, 1996. 26.
- [24] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *European Conference on Machine Learning*, pages 171–182, 1994.
- [25] F. Korn, N. Sidoropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. *Fast Nearest Neighbor Search in Medical Image Databases*. University of Maryland, 1996.
- [26] W. J. Krzanowski. Selection of variables to preserve multivariate data structure, using principal component analysis. *Applied Statistics- Journal of the Royal Statistics Society Series C*, 36:22–33, 1987.

- [27] Y. Le Cun, J.S. Denker, S.A. Solla, et al. Optimal brain damage. *Advances in Neural Information Processing Systems*, 2:598–605, 1990.
- [28] T.M. Lehmann, M.O. Güld, T. Deselaers, D. Keysers, H. Schubert, K. Spitzer, H. Ney, and B.B. Wein. Automatic categorization of medical images for content-based retrieval and data mining. *Computerized Medical Imaging and Graphics*, 29(2-3):143–155, 2005.
- [29] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.
- [30] G. P. McCabe. Principal variables. *Technometrics*, 26:127–134, 1984.
- [31] H. Murase and S.K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.
- [32] R.M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.
- [33] S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (COIL-100). *Techn. Rep. No. CUCS-006-96, dept. Comp. Science, Columbia University*, 1996.
- [34] S. Obdrzalek and J. Matas. Object recognition using local affine frames on distinguished regions. *BMVC 2002*, pages 113–122, 2002.
- [35] J. Peng, B. Bhanu, and S. Qing. Probabilistic feature relevance learning for content-based image retrieval. *Computer Vision and Image Understanding*, 75(1):150–164, 1999.
- [36] D.I. Perrett, A.J. Mistlin, and A.J. Chitty. Visual neurones responsive to faces in the monkey temporal cortex. *Trends Neurosci.*, 10:358–364, 1987.
- [37] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, 1990. 34.
- [38] Y.A. Qi, T.P. Minka, R.W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. *ACM International Conference Proceeding Series*, 2004.
- [39] D. Roobaert and MM Van Hulle. View-based 3d object recognition with support vector machines. *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 77–84, 1999.
- [40] A.S. Sato and K. Yamada. Generalized relevance learning vector quantization. *Advances in Neural Information Processing Systems*, 7:423–429, 1995.
- [41] G. Schneider, H. Wersing, B. Sendhoff, E. Korner, G. Schneider, and H. Wersing. Evolution of hierarchical features for visual object recognition. *Third Workshop on SelfOrganization of AdaptiVE Behavior (SOAVE 2004) Ilmenau*, pages 104–113, 2004. 9.
- [42] A. Shokoufandeh, I. Marsic, and S.J. Dickinson. View-based object recognition using saliency maps. *Image and Vision Computing*, 17(5):445–460, 1999.

- [43] K. Sims and H.L. Voorhees. Handwritten character classification using nearest neighbor in large databases. *Proc. 8th SCIA. Trom. c. Norway*, 1993.
- [44] M. Strickert, T. Bojer, and B. Hammer. Generalized relevance lvq for time series. *Artificial Neural Networks - ICANN'2001*, pages 677–683, 2001.
- [45] M.J. Tarr and HH Bülthoff. Is human object recognition better described by geon-structural-descriptions or by multiple-views. *Journal of Experimental Psychology: Human Perception and Performance*, 21(6):1494–1505, 1995.
- [46] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.
- [47] G.T. Toussaint. Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining. *International Journal of Computational Geometry and Applications*, 15(2):101–150, 2005.
- [48] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
- [49] S.K. Warfield. Fast k-nn classification for multichannel image data. *Pattern Recognition Letters*, 17(7):713–721, 1996.
- [50] P. Wu and B.S. Manjunath. Adaptive nearest neighbor search for relevance feedback in large image databases. *Proceedings of the ninth ACM international conference on Multimedia*, pages 89–97, 2001.
- [51] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 2004.