3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

Friedhelm Schwenker   Hans A. Kestler   Günther Palm

Department of Neural Information Processing
University of Ulm
D-89069 Ulm

In this chapter we present a 3-D visual object recognition system for an autonomous mobile robot. This object recognition system performs the following three tasks: Object localisation in the camera images, feature extraction, and classification of the extracted feature vectors with hierarchical radial basis function (RBF) networks.

1 Introduction

The recognition of 3-D objects from 2-D camera images is one of the most important goals in computer vision. There is a large number of contributions to this field of research from various disciplines, e.g. artificial intelligence and autonomous mobile robots [1, 2], artificial neural networks [3–5], computer vision and pattern recognition [6–10], psychophysics and brain theory [11–13]. Due to the increasing performance of current computer systems and the increasing development of computer vision and pattern recognition techniques several 3-D object recognition systems have been developed [14–17]. Among these many different approaches to 3-D object recognition two main streams can be detected: structural-based or primitives-based approaches and view-based methods.

In primitives-based approaches the 3-D objects are modelled using a small set of 3-D volumetric primitives (cubes, cylinders, cones, etc) in a CAD-like model. In the recognition phase the most important step is to identify the primitives that are visible in the camera image. This approach is derived from the *recognition-by-components* theory developped by Biederman in [18, 19]. It seems that this approach is reasonable for CAD applications, but has its limitations for the recognition of free-form objects, for example in face recognition.
Psychophysical results achieved during the last years have shown that humans are able to learn to recognize 3-D objects from different characteristic 2-D views. In these view-based approaches a set of 2-D views of each object is stored or learned in order to build an internal object representation of the 3-D object. In the recognition phase of such a view-based system a single 2-D view of an object is compared to the learnt 2-D views. This processing step is related to methods like template matching and nearest neighbor classification. One of the main tasks in these view-based approaches is the selection of characteristic object views. The objects have to be recorded from various viewpoints, in different poses and with different illumination in order to build a recognition system which is robust under all such transformations.
Artificial neural network models can be used to learn to recognize 3-D objects on the basis of a small set of 2-D camera images which are recorded from distinct view points [4]. Based on a training set of feature vectors, the network learns a discrimination function in the highdimensional feature space. For this kind of classification task supervised network training procedures must be utilized.
Often synthetic images or well prepared data sets ignoring problems which are present at lower processing levels have been used in order to simplify the recognition problem, e.g. the 3-D objects are always in the center of the camera images. We attempt to solve a more realistic problem and use camera images recorded from real 3-D objects for training and testing the recognition system. In the recognition phase scenes with multiple 3-D objects may be presented to the recognition system.
The recognition of a 3-D object consisted of the following three subtasks which will be discussed throughout this chapter:

1. **Localization of objects in the camera image.**

   In this processing step the entire camera image is segmented into regions. Each region should contain exactly one single 3-D object. Only these marked regions, which we call the regions of interest (ROI), are used for further image processing. Colour-based approaches for the ROI-detection are used.

2. **Extraction of characteristic features.**

   From each ROI within the camera image a set of features is computed. For this, the ROIs are divided into subimages and for each subimage an orientation histogram with eight orientation bins is calculated from the gray valued image. The orientation histograms of all subimages are concatenated into the characterizing feature vector.

3. **Classification of the extracted feature vectors.**

   The extracted feature vectors together with the target classification are used in a supervised learning phase to build the neural network classifier. After network training novel feature vectors are presented to the classifier which outputs the estimated class labels.

We address all these three topics in this chapter, but we will focus on the classification task. The chapter is organized in the following way: In Section 2 the methods for 3-D object localization and feature extraction are described. RBF networks and support vector learning in RBF networks including multiclass SVM classifiers are discussed in Section 3. The construction of binary classifier trees is the topic of Section 4. In Section 5 we present some numerical classification results for the recognition system and finally a conclusion is given.

2 Object Localisation and Feature Extraction

Visual Attention—Regions of interest in the camera image.

The classification of visual 3-D objects in camera images requires their reasonable delimitation from background. A possible way to achieve this is to locate regions of interest within the camera image. For this visual attention task biologically motivated models, like saliency-based methods have been intensively investigated during the last years [20–22]. In our case the regions of interest should contain areas of single objects. From the subimages enclosed by these regions, features are then extracted for the object classification. Because computational requirements are tight, currently a colour-based approach for the object detection is used, see [23] and Figure 1. After downsampling the original camera image the resulting image is transformed from RGB (red, green, blue) to HSV-colour space (hue, saturation, value) [24]. HSV colour space is used, because this colour model makes it very easy to describe colour ranges independent of saturation and lightness. Downsampling has the effect of a low pass filtering and also reduces the computational complexity. After downsampling and colour space transformation, each pixel inside a valid colour range is labeled, and sets of connected labeled pixels (so-called colour blobs) are calculated. For every found colour-blob the colour range and number of pixels is determined. If the number of pixels of a colour-blob is larger then a predefined threshold value, a bounding box for this blob is calculated.

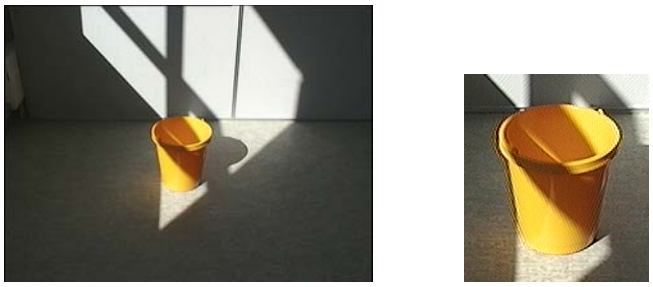


Figure 1. Examples of class `bucket` of the data set (left) and the calculated region of interest (right)

In addition to these processing steps, the following heuristics for merging regions are applied:

1. In the first step, bounding boxes which are contained in larger ones are merged. This is typically applied to objects, whose colour range does not match a single valid range, but matches two or more valid colour ranges.

2. After this pre-processing step, the distances between centers of regions within the same colour range are calculated. If the distance between two regions is less than a predefined value, they will be

merged. This is useful for merging small colour blobs, for example bottles, whose colour regions are usually separated by the label of the bottle.

3. In a last processing step, the bounding boxes determined on the downsampled image are rescaled to the original image size and are slightly enlarged.

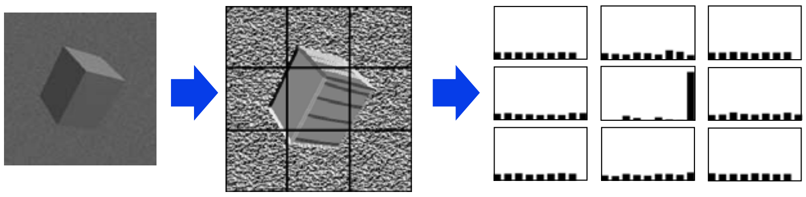


Figure 2. Elements of the feature extraction method. The grey valued image (left) is convolved with the masks S_x and S_y (see text) resulting in the gradient image (center; absolute value of the gradient). Orientation histograms (right) of non–overlapping subimages constitute the feature vector.

Feature Extraction

The image within the region of interest is divided into $n \times n$ non–overlapping subimages and for each subimage the orientation histogram of eight orientations (range: $0 - 2\pi$, dark/light edges) is calculated [25] from the gray valued image. The orientation histograms of all subimages are concatenated into the characterizing feature vector.
The gradient of an image $f(x, y)$ at location (x, y) is the two dimensional vector

$$\begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \approx \begin{pmatrix} f * S_x \\ f * S_y \end{pmatrix}$$

($*$ denotes the convolution operation). Gradient directions (S_x, S_y) were calculated with 3×3 Sobel operators. The gradient directions are calculated with respect to the x-axis:

$$\alpha(x, y) = \text{atan2}\,(f * S_y, f * S_x)$$

The *atan2* function corresponds to the *atan* but additionally uses the sign of the arguments to determine the quadrant of the result. The eight bins of the histogram all have equal size ($2\pi/8$). The histogram values are calculated by counting the number of angles falling into the respective bin. Histograms are normalized to the size of their subimages.

3 Learning in RBF Networks

RBF networks were introduced into the neural network literature by Broomhead and Lowe in 1988 [26]. The RBF network model is motivated by the locally tuned response observed in biologic neurons. Neurons with a locally tuned response charateristic can be found in several parts of the nervous system, for example cells in the auditory system selective to small bands of frequencies or cells in the visual cortex sensitive to bars oriented in a certain direction. These locally tuned neurons show response characteristics bounded to a small range of the input space.
The theoretical basis of the RBF approach lies in the field of interpolation of multivariate functions. We consider multivariate functions $f : \mathbb{R}^d \to \mathbb{R}^m$. Without loss of generality we may assume that m is equal to 1. The goal of interpolating a set of tupels $(x^\mu, y^\mu)_{\mu=1}^M$ is to find a function $F : \mathbb{R}^d \to \mathbb{R}$ with $F(x^\mu) = y^\mu$ for all $\mu = 1, \dots, M$, where F is an element of a predefined set (often a vector space) of functions. In the RBF approach the interpolating function F is a linear combination of basis functions:

$$F(x) = \sum_{\mu=1}^{M} b_\mu h(\|x - x^\mu\|) + p(x) \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm, $b_1, \dots, b_M$ are real numbers, h a real valued function, and p a polynomial $p \in \Pi_n^d$ (polynomials of degree at most n in d variables). The degree of the polynomial term has to be fixed in advance. The interpolation problem is to determine the real coefficients $b_1, \dots, b_M$ and the polynomial term $p := \sum_{l=1}^{D} a_l p_j$ where $p_1, \dots, p_D$ is a basis of Π_n^d and $a_1, \dots, a_D$ are real numbers. The function F has to satisfy the conditions:

$$F(x^\mu) = y^\mu, \quad \mu = 1, \dots, M$$

and

$$\sum_{\mu=1}^{M} b_\mu p_j(x^\mu) = 0, \quad j = 1, \dots, D.$$

Sufficient conditions for the unique solvability of the interpolation problem were given by several authors e.g. see results due to Micchelli, Powell, or Light [27–29]. The function h is called a *radial basis function* if the interpolation problem has a unique solution. In some cases the polynomial term in equation (1) can be omitted, then the interpolation problem is equivalent to the matrix equation

$$Hb = y \tag{2}$$

where $b = (b_1, \ldots, b_M)$, $y = (y^1, \ldots, y^M)$, and $M \times M$ matrix H defined by

$$H = (h(\|x^\nu - x^\mu\|))_{\mu,\nu=1,\ldots,M}.$$

Provided the inverse of H exists, the solution of the interpolation problem has the form:

$$b = H^{-1}y. \tag{3}$$

Examples for radial basis functions h often used in applications are:

$$\begin{aligned} h(r) &= e^{-r^2/\sigma^2} \\ h(r) &= (r^2 + \sigma^2)^{1/2} \\ h(r) &= (r^2 + \sigma^2)^{-1/2} \end{aligned}$$

Here, σ is a positive real number which we call the scaling parameter or the width of the radial basis functions. The most popular and widely used radial basis function is the Gaussian function $h(\|x - c\|) = exp(-\|x - c\|^2/\sigma^2)$ with peak at center $c \in \mathbb{R}^d$ and decreasing as the distance from the center increases.

The solution of the exact interpolating RBF mapping passes through every data point (x^μ, y^μ). In the presence of noise in the data the exact solution of the interpolation problem is typically a function badly oscillating between the given data points. An additional problem with the exact interpolation procedure is that the number of basis functions is equal to the number of data points and so calculating the inverse of the $M \times M$ matrix H becomes intractable in practice. In applications, where we have to deal with many thousands of noisy data points an approximative solution to the data is more desirable than an interpolative one. Broomhead and Lowe [26] first proposed to reduce the number of basis functions in order to reduce the computational complexity. This technique produces a solution approximating instead of interpolating the data points. Furthermore, in [26] an interpretation of the RBF method as an artificial neural network model is given. It consists of three neural layers: a layer of input neurons feeding the feature vectors into the network, a hidden layer of RBF neurons, calculating the outcome of the basis functions, and a layer of output neurons, calculating a linear combination of the basis functions. Under some additional conditions to the basis function h the set of RBF networks with free adjustable prototype vectors are shown to be universal approximators, so that any continous function can be approximated with arbitrary precision [30]. This implies that RBF-networks with adjustable prototypes can also be used for classification tasks [31].

Support Vector Learning

Three different training procedures to train an RBF network are known: Two-stage training, Backpropagation training, and Support vector training. We concentrate on support vector learning [32–34]. The support vector machine (SVM) was initially developed by Vapnik to classify data points of a linear separable data set. In this case a training set consisting of M examples (x^μ, y^μ), $x^\mu \in \mathbb{R}^d$, and $y^\mu \in \{-1, 1\}$ can be divided up into two sets by a separating hyperplane. Such a hyperplane is determined by a weight vector $b \in \mathbb{R}^d$ and a bias or threshold $\theta \in \mathbb{R}$ satisfying the separating contraints

$$y^\mu \;\; [\langle x^\mu, b\rangle + \theta] \geq 1 \quad \mu = 1, \dots, M.$$

The distance between the separating hyperplane and the closest data points of the training set is called the margin. Intuitively, the larger the margin, the higher the generalization ability of the separating hyperplane. The optimal separating hyperplane with maximal margin is unique and can be expressed by a linear combination of those training examples lying exactly at the margin. These data points are called the support vectors. The separating hyperplane has the form

$$H(x) = \sum_{\mu=1}^{M} \alpha_\mu^* y^\mu \langle x, x^\mu \rangle + \alpha_0^*$$

where $\alpha_1^*, \dots, \alpha_M^*$ is the solution optimizing the functional

$$Q(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu=1}^{M} \alpha_\mu \alpha_\nu y^\mu y^\nu \langle x^\mu, x^\nu \rangle$$

subject to the constraints $\alpha_\mu \geq 0$ for all $\mu = 1, \ldots, M$ and $\sum_{\mu=1}^{M} \alpha_\mu y^\mu = 0$. Then a training vector x^μ is a support vector if the corresponding coefficient $\alpha^*_\mu > 0$. Then the weight vector b is determined, $b = \sum_{\mu=1}^{M} \alpha_\mu x^\mu$ and the bias term α^*_0 is defined by a single support vector (x^s, y^s):

$$\alpha^*_0 = y^s - \langle x^s, b \rangle.$$

The SVM approach has been extended to the nonseparable situation and to the regression problem. In most applications (regression or pattern recognition problems) linear solutions are insufficient. For example, in real world pattern recognition problems it is common to define an appropriate set of nonlinear mappings $g := (g_1, g_2, \ldots)$, each g_j defined as a real valued function, transforming the input vectors x^μ to a vector $g(x^\mu)$ which is element of a new feature space $\mathcal{H}$. Then the separating hyperplane can be constructed in the feature space $\mathcal{H}$ and can be expressed by

$$H(x) = \sum_{\mu=1}^{M} \alpha_\mu y^\mu \langle g(x), g(x^\mu) \rangle + \alpha_0.$$

Provided $\mathcal{H}$ is a Hilbert space, the explicit mapping $g(x)$ does not need to be known since it can implicitly be defined by a kernel function

$$K(x, x^\mu) = \langle g(x), g(x^\mu) \rangle$$

representing the inner product of the feature space. With a suitable choice of a kernel function the data can become separable in feature space despite being not separable in the input space. Using a kernel function K satisfying the condition of Mercer's theorem (see [34] for details), the separating hyperplane is given by

$$H(x) = \sum_{\mu=1}^{M} \alpha_\mu y^\mu K(x, x^\mu) + \alpha_0.$$

The coefficients α_μ can be found by solving the optimization problem

$$Q(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu=1}^{M} \alpha_\mu \alpha_\nu y^\mu y^\nu K(x^\mu, x^\nu)$$

subject to the contraints $0 \leq \alpha_\mu \leq C$ for all $\mu = 1, \ldots, M$ and

$$\sum_{\mu=1}^{M} \alpha_\mu y^\mu = 0$$

where C is a predefined positive number. An important kernel function satisfying Mercers condition is the Gaussian kernel function

$$K(x, y) = e^{-\frac{\|x-y\|_2^2}{\sigma^2}}.$$

The separating surface obtained by the SVM approach is a linear combination of Gaussian functions located on the support vectors. The SVM reduces to an RBF network. In contrast to RBF networks trained by backpropagation learning the centers are located at the data points of the training set. Furthermore, the number of centers is automatically determined in the SVM approach.

Multiclass Classification

In many real world applications, e.g. speech recognition or optical character recognition, a multi-class pattern recognition problem has to be solved. The SVM classifiers as previously described are formulated as binary classifiers. Various approaches have been developed in order to deal with multi-class classification problems. The following strategies can be applied to build N-class classifiers utilizing binary SVM classifiers:

1. N ***one-against-rest*** **classifiers.**

   In this approach N different classifiers are constructed, one classifier for each class. Here the l-th classifier is trained on the whole training data set in order to classify the members of class l against the rest. For this, the training examples have to be re-labeled: Members of the l-th class are labeled to 1; members of the other classes to -1. In the classification phase the classifier with the maximal output defines the estimated class label of the current input vector.

2. $N(N-1)/2$ ***one-against-one*** **classifiers.**

   For each possible pair of classes a binary classifier is calculated. Each classifier is trained on a subset of the training set containing only training examples of the two involved classes. As for the *one-against-rest* strategy the training sets have to be re-labeled. All $N(N-1)/2$ classifiers are combined through a majority voting scheme to estimate the final classification [35, 36]. Here the class with the maximal number of votes among all $N(N-1)/2$ classifiers is the estimation.

3. **Extension of the original SVM formulation to the N-class classification problem.**

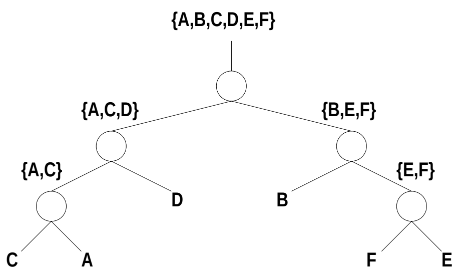
   Weston and Watkins proposed in [37] a natural extension to the binary SVM approch to solve the N-class classification problem directly. Here re-labeling of the training data is not necessary. All the N classes are considered at once, and the separating conditions are integrated into the same optimization problem. As for the *one-against-rest* classifiers, the result is a N-class classifier with N weight vectors and N threshold values. The recall phase is organised as for the *one-against-rest* classifier strategy.

4. **Hierarchies or trees of binary SVM classifiers.**

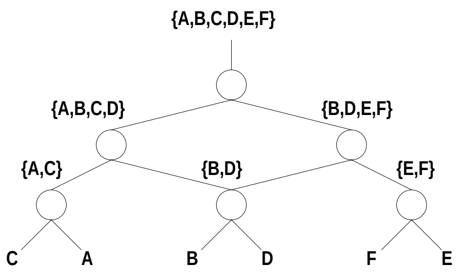
   Here the multi-class classification problem is decomposed into a series of binary classification sub-problems organized in a hierarchy, see Figure 3. We discuss this approach in the next section.

4 SVM Classifier Trees

One of the most important problems in multi-class pattern recognition problems is the existence of confusion classes. A confusion class is a subset of the set of the classes $\{1, \ldots, N\}$ in which very small differences in the feature vectors may lead to misclassifications. For example, in OCR the measured features for members of the classes `o`, `O`, `0` and `Q` are typically very similar, so $\{$`o, O, 0, Q`$\}$ defines a confusion class. The major idea of hierarchical classification is to make a coarse discrimination between confusion classes first and a finer discrimination within the confusion classes later [38].
In Figure 3 examples of hierarchical classifiers are depicted. Each node within the graph represents a SVM classifier discriminating feature vectors of a confusion class into one of two smaller confusion classes or possibly into indiviual classes. The terminal nodes of the graph (called leaves) represent these individual classes, and the other nodes are classifiers performing a binary decision task, thus these nodes have exactly



(a) Binary tree classifier



(b) General hierarchical classifier

Figure 3. Two examples of hierarchical classifiers. The graphs are directed acyclic graphs with a single root node at the top of the graph and with terminal nodes (leaves) at the bottom. Individual classes are represented in the leaves, the other nodes within the graph are classifiers performing a binary decision task, which is defined through the annotations at the incoming and the outgoing edges.

two children. Nodes within the graph may have more than one incoming edge. Figure 3a shows a tree-structured classifier, where each node has exactly one incoming edge. In Figure 3b a more general classifier structure defined through a special directed acyclic graph is depicted. In the following we restrict our considerations to SVM trees.
The classification subtask is defined through the annotations at the incoming and outgoing edges of the node. Let us consider for example the SVM classifier at the root of the tree in Figure 3a. The label of the incoming edge is $\{A, \ldots, F\}$, so for this (sub-)tree a 6-class classification task is given. The edges to the children are annotated with $\{A, C, D\}$ (left child) and $\{B, E, F\}$ (right child). This means that this SVM has

to classify feature vectors into confusion class $\{A, C, D\}$ or $\{B, E, F\}$. To achieve this, all members of the six classes $\{A, \ldots, F\}$ have to be re-labeled: Feature vectors with class labels A, C, or D get the new label -1 and those with class label B, E, or F get the new label 1. After this re-labeling procedure the SVM is trained as described in the previous section. Note, that re-labeling has to be done for each classifier training.
So far we have not answered the question how to construct the hierarchy of subsets of classes. In some cases it may be a priori defined. For example in applications where the set of classes is hierarchically arranged based on some kinds of symbolic properties which are in general different from the measured features. In OCR for example, the set of characters can be divided into digits and letters, etc, and further the letters into upper and lower caser letters.
An attempt to design a hierarchy of confusion classes is to consider the confusion matrices. Let K be a set of classes that should be divided into two disjoint subsets K_1 and K_2. A partition has to be evaluated by some kind of error function counting the number of errors made through the current class assignments of K_1 and K_2:

$$E(K_1, K_2) := \sum_{\substack{(i \in K_1 \wedge j \in K_2) \vee \\ (j \in K_1 \wedge i \in K_2)}} q_{ij}$$

here q_{ij} is the frequency of feature vectors from class i classified to class j. To design the whole hierarchy this procedure has to be applied until all confusion classes have exactly a single class.
This approach has two main disadvantages: a) The whole N-class classification problem has to be solved to determine the confusion matrix. To determine the confusion matrices, either classifiers have to be trained by supervision or the confusion matrices have to be calculated by k-nearest-neighbor procedures. b) The exact evaluation of all possible partitions is computational expensive for large N, because the number of possible binary partitions is equal to $2^{N-1} - 1$. For large N an exact enumeration of all possible binary partitions is intractable, and so the confusion classes have to be calculated heuristically [23].
Another approach is the partition of classes K into subsets K_1 and K_2 by clustering or vector quantisation methods. In clustering and vector quantisation a set of representative prototypes $\{c_1, \ldots, c_k\} \subset \mathbb{R}^d$ is determined by unsupervised learning from the feature vectors $x^\mu, \mu =$

$1, \ldots, M$ of the training set. For each prototype c_j the Voronoi cells R_j and clusters C_j are defined by

$$R_j = \{x \in \mathbb{R}^d \ : \ j = \text{argmin}_i \|c_i - x\|\}$$

and

$$C_j = R_j \cap \{x^\mu \ : \ \mu = 1, \ldots, M\}.$$

The relative frequency of members of class i in cluster j is

$$p_{ij} = \frac{|\Omega_i \cap C_j|}{|C_j|}.$$

For class i the set Ω_i is defined by

$$\Omega_i = \{x^\mu \ : \ \mu = 1, \ldots, M, \ y^\mu = i\}$$

where y^μ denotes the teacher signal of feature vector x^μ. The k-means clustering with $k = 2$ cluster centers c_1 and c_2 defines a hyperplane within the feature space $\mathbb{R}^d$ separating two sets of feature vectors. From the corresponding clusters C_1 and C_2 a partition of the set of classes K into two subsets K_1 and K_2 can be achieved through the following assignment:

$$K_j = \{i \in K \ : \ j = \text{argmax}\,\{p_{i1}, p_{i2}\}\}, \quad j = 1, 2.$$

Recursively applied, this procedure leads to a binary tree of confusion classes as depicted in Figure 3a, where each node within the tree is labeled with a confusion class and defines a partition into two sub-confusion classes. This assignment scheme can be extended in several directions, e.g. for schemes with $k > 2$ cluster centers or for non-binary trees.

5 Data and Classification Results

Data Sets

Two different data sets were used in performance evaluation of the classifier, artificially generated and a data set of real camera images.

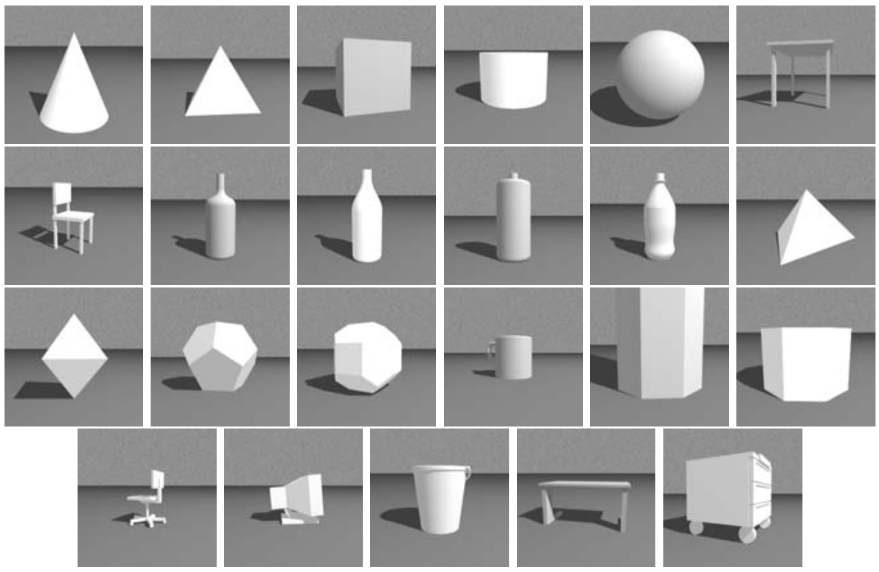


Figure 4. Examples of all 23 classes. Labels are from top left to right bottom: cone, pyramid, cube, cylinder, bowl, table, chair, bottle 1 to 4, tetrahedron, octahedron, dodecahedron, gem, coffee mug, column, clipped column, office chair, monitor, bucket, office table and drawer.

Artificial data

Images of 23 different objects were generated with the raytracing software PoV–Ray. Examples of the 5920 objects (class 0 to 5: 250 images; class 6 to 22: 260 images) are given in Figure 4. Images (256×256 pixel) were created through a virtual helical movement of the observer around the object (constant distance). The anzimut angle, was varied in steps of $36°$($0°$to $360°$) with reference to the object. The declination angle was varied in steps of $1.5°$($0°$to $15°$). A point light source made the same circular movement $45°$ahead of the observer but at a constant declination angle of $45°$. During this ”movement” the objects were rotated at random (uniform distribution) around their vertical axis, the scaling of the objects (x,y,z -axis) was varied uniformly distributed in the range of 0.7 to 1.3 and the ambient light was varied at random (uniform) in the range of 0.2 to 0.6. An example of the variation within one class is given in Figure 5. From all images, 5×5 histograms (see Section 2 and Figure 2) were concatenated into a feature vector that serves as input to the classifier.

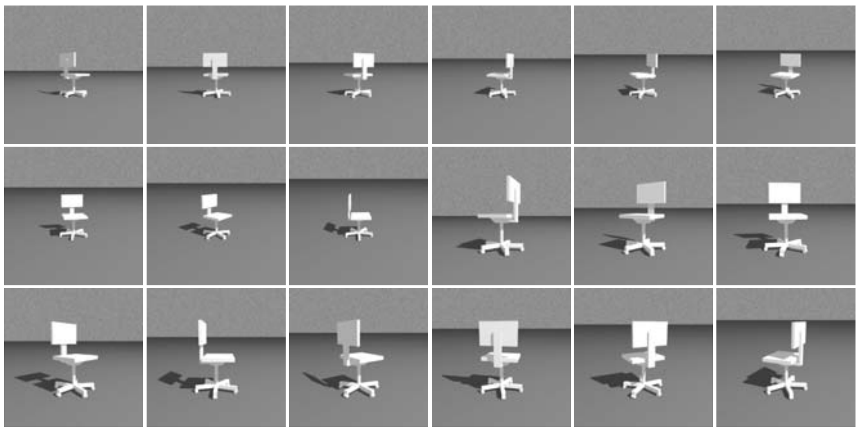


Figure 5. Examples illustrating the variation inside the class `office chair`.

Real–world data set of camera images

Camera images were recorded for six different 3-D objects (orange juice bottle, small cylinder, large cylinder, cube, ball and bucket) with an initial resolution of 768×576 pixels. To these five objects nine classes were assigned (bottle lying/upright, cylinders lying/upright). The test scenes were acquired under mixed natural and artificial lighting. Regions of interest where calculated from 1800 images using the colour-blob detection method. These regions where checked and labeled by hand, 1786 images remained for evaluation. Regions of interest are detected using three colour ranges, one for red (bucket, cylinder, ball), blue (cylinder) and yellow (cylinder, bucket, orange juice). The image in Figure 1 gives an example of the automatically extracted regions of interest. Feature vectors were calculated concatenating 5×5 orientation histograms calculated through 3×3 Sobel operator, see Section 2 and Figure 2.

Results

In order to get an overview over the underlying structure of the data, two data analysis tools are briefly mentioned. They are useful to explore large sets of highdimensional feature vectors:

1. **Clustering** of the high dimensional feature vectors utilizing for example the k-means clustering algorithm in order to get a smaller set

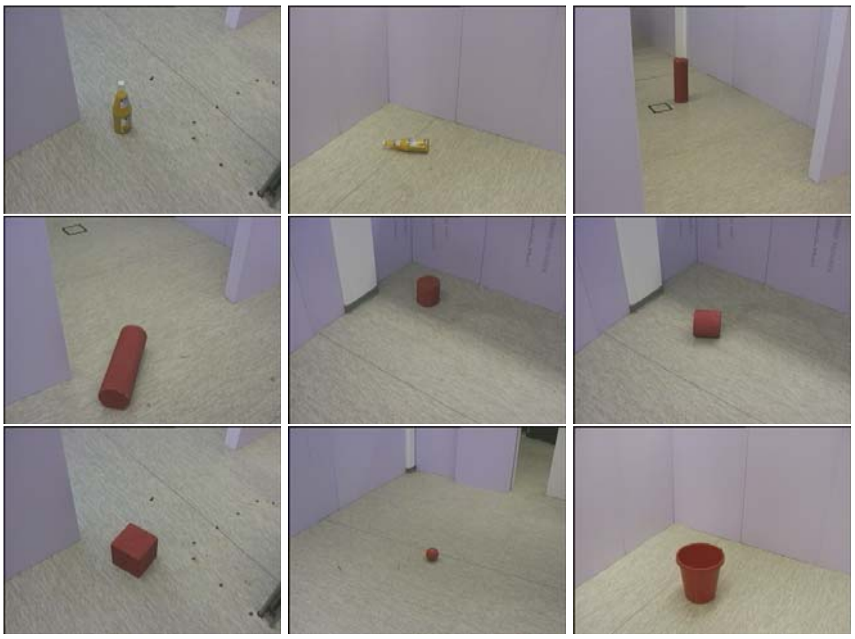


Figure 6. Examples illustrating the real–world data set. Labels are from top left to right bottom: class 0/1: orange juice bottle upright/lying, class 2/3: large cylinder upright/lying, class 4/5: small cylinder upright/lying, class 6: cube, class 7: ball, class 8: bucket.

of representative prototypes. The feature vectors from each class are clustered separately leading to a set of prototypes, each representing a certain class. For the union of all prototypes a distance matrix is calculated. This distance matrix can be plotted as a matrix of gray values (see Figure 7) and used for further data analysis.

The distance matrix of $6 \times 9 = 54$ k-means prototypes of the recorded camera images is shown in Figure 7. Small distances can be observed between prototypes within the classes $\{2, 3, 4, 5, 6, 8\}$ and within $\{0, 1\}$. The prototypes of class 7 seem to be separated from the others, but some smaller distances to prototypes of classes $\{0, 1, 4, 5, 6\}$ can be detected. These smaller distances between prototypes of different classes typically lead to misclassifications.

2. **Nonlinear distance preserving projections** of the data points into a lowdimensional projection space (typically $\mathbb{R}^2$) may be used to explore the data set. The projection of large data sets is computa-

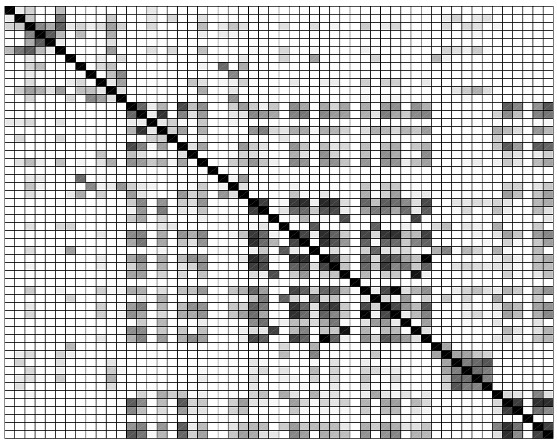


Figure 7. A 54×54 (Euclidean) distance matrix for 54 k-means prototyes (6 prototypes per class) of the recorded camera images. The 54 prototypes are sorted by its class memberships in such a way that the prototypes $c_1, \ldots, c_6$ are members of class 0, $c_7, \ldots, c_{12}$ have class label 1, etc. Distances between prototypes have been encoded into gray values (small distances black, large distance white).

tional expensive and intractable, therefore only the prototypes are projected which gives a rough overview over the data (see [39] for some more details on clustering and visualization of large and highdimensional data sets). In Figure 8 the 54 prototypes of the k-means cluster procedure are shown as projections (with class labels) to $\mathbb{R}^2$. Here, in the lower left part of Figure 8 the projections of the prototypes of classes $0, 1$ and 7 are located, the prototypes of the other classes cover the rest of the projection area. This is a similar observation as already seen for distance matrix of the cluster centers. This result is not surprising because Spearman's rank order coefficient of the two sorted distance sequences (distances in feature space vs. distances in projection space) has been determined to $r_s = 0.98$. The value of Spearman's rank order coefficient r_s is within the interval $[-1, 1]$, where larger values close to 1 indicate similar rank orders of the sequences, so that the calculated value of $r_s = 0.98$ shows that the order of the distances between the cluster centers in the feature space are fairly well represented by the distances of the corresponding projections in $\mathbb{R}^2$. For a more detailed visualization of the data set clustering and pro-

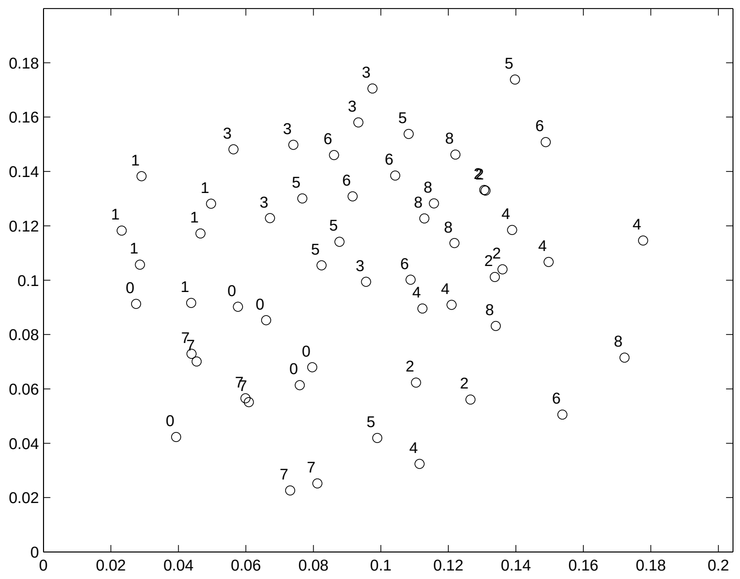


Figure 8. A distance preserving 2-D projection of the 54 prototypes (see text). The class membership of a prototype is used to annotate of its 2-D projection.

jection algorithms have to be applied to the feature vectors of the confusion classes.

In Figure 9 the structure of the confusion classes for the real-world data set of recorded camera images is given. The binary tree of the confusion classes has been determined by 2-means clustering followed by the assignment scheme previously described. For a partition of a certain confusion class, 5 clustering runs have been applied to the feature vectors of this confusion class. Each clustering run has been initialized with a randomly selected subset of the data set serving as initial setting of the cluster centers. Except for the confusion class $\{4, 5, 6\}$, always the same split of the confusion classes has been found. For confusion class $\{4, 5, 6\}$ the partition $\{4, 5\}$ and $\{6\}$ which is shown in Figure 9 has been found once, the trivial partition $\{4, 5, 6\}$ and $\emptyset$ has been found in the other four clustering runs. In the nodes the classification accuracies of the particular classification task are given. As already observed for the distance matrix of the class specific prototypes and in the 2-D projection of the prototypes the two confusion classes $\{0, 1, 7\}$ and $\{2, 3, 4, 5, 6, 8\}$ may be defined and have been determined by the 2-means clustering pro-

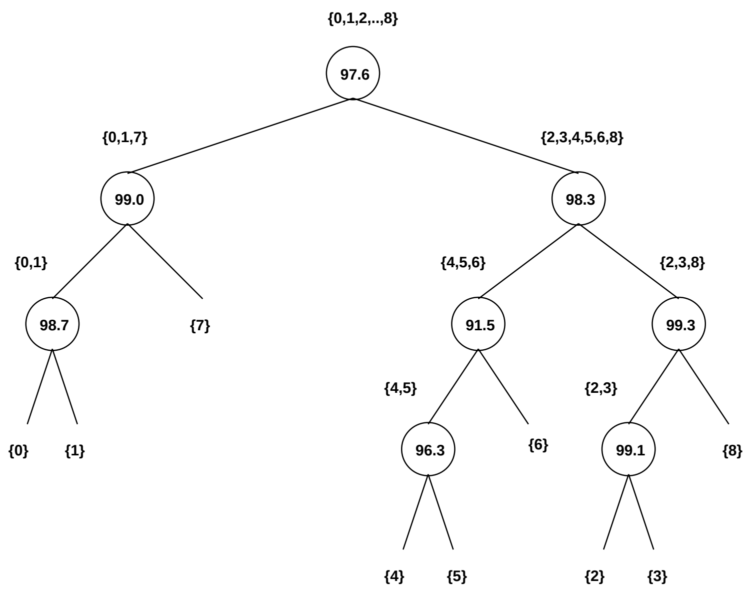


Figure 9. Binary tree classifier for the set of feature vectors derived from the ROIs of the camera images from the real-world data set. Each node within the tree is labeled with a confusion class and the classification rate of the defined sub-classification problem (mean of three 5-fold crossvalidation runs). In the leaves the original individual classes are represented.

cedure at the root level.
The classification results of four different classifiers are presented for both data sets:

1NN: 1-nearest neighbor classifier with Euclidean distance.

LVQ: 1-nearest neighbor classifier trained by Kohonen's software package OLVQ1 and LVQ3 training each algorithm for 50 training epoches; 10 prototypes per class.

RBF-SVM: A set of support vector networks each with Gaussian kernel function; N-against-rest learning/voting strategy; NAG library for optimization.

RBF-SVM-HC: A binary tree of support vector networks each with Gaussian kernel function; a binary tree of confusion classes is determined with 2-means clustering; NAG library for optimization.

Classifier	**Camera Images**	**PoV-RAY Data**
1NN	91.04	92.45
LVQ	92.44	92.80
RBF-SVM	93.89	93.32
RBF-SVM-HC	94.28	93.06

Table 1. Classification accuracies of the real–world camera images and the artifically generated PoV-RAY data set for the 1-nearest-neighbor classifier, an LVQ-classifier trained by OLVQ1 and LVQ3 with $k = 90$ prototypes (for the camera images) respectively $k = 230$ (for the POV-RAY data set) prototypes, an RBF network trained by support vector learning (N-against-rest strategy), and a binary tree of RBF classifiers each trained by support vector learning. The mean of three 5-fold crossvalidation runs is given.

The classification results given in Table 1 are the means of three 5–fold crossvalidation runs. The classification rates of the SVM classifiers are significantly higher than the accuracies for the **1NN** and **LVQ** classifiers. For the real world data set the results of the SVM tree classifier are remarkably good, slightly better as for the flat SVM classifier with the N-against-rest strategy. It can be observed in Figure 9 that many misclassifications appear by discriminating between the classes $\{4, 5, 6\}$, because the extracted feature vectors looked very similar. In such cases different features have to be extracted, e.g. edges, colours or texture.

6 Conclusion

In this chapter, we presented a 3-D visual object recognition system for a mobile robot. The components of this system perform the subtasks: (1) Localization of the 3-D objects in the camera images based on colour blob detection, the ROIs in the camera images, (2) extraction of low-level features within the determined ROIs—a set of orientation histograms calculated from the ROIs, and (3) classification of the extracted feature vectors utilizing a tree of RBF-classifiers, each classifier trained by support vector learning. Classification results based on crossvalidation have been presented for 1-NN, LVQ, SVM, and SVM-trees. We have assessed the potential of nonlinear SVMs and trees of nonlinear SVMs in the problem of recognizing 3-D objects from a single object view. In comparison to other algorithms, it appears that the classification results of SVM tree classifiers are remarkably good. The proposed SVM tree classifier is a flexible classifier architecture which is able to deal with different sets of features which may be useful in applications with many thousands of different objects to classify. Future work in this area includes the integration of additional features like edges, colours, texture, etc. where different features should be used in the visual attention module and in particular for the hierarchical classifier.

Acknowledgments

The research described here was part of the Collaborative Research Center (SFB 527) and is supported by the German Science Foundation (DFG).

References

[1] Brooks, R. (1983). Model-based three-dimensional interpreations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 140–149.

[2] Lowe, D. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, vol. 31, pp. 355–395.

[3] Little, J., T. Poggio, and E. Gamble (1988). Seeing in parallel: The vision machine. *International Journal of Supercomputing Applications*, vol. 2, pp. 13–28.

[4] Poggio, T. and S. Edelman (1990). A network that learns to recognize tree-dimensional objects. *Nature*, vol. 343, pp. 263–266.

[5] Schiele, B. and J. Crowley (1996). Probabilistic object recognition using multidimensional receptive field histograms. In *Proc. of the 13th Int. Conf. on Pattern Recognition*. IEEE Computer Press, pp. 50–54.

[6] Marr, D. and H. Nishihara (1978). Representation and recognition of the spatial organization of three dimensional structure. *Proceedings of the Royal Society of London B*, vol. 200, pp. 269–294.

[7] Marr, D. (1982). *Vision*. Freeman, San Fransisco.

[8] Ullman, S. (1996). *High-level Vision. Object Recognition and Visual Cognition*. The MIT Press, Cambridge.

[9] Basri, R. (1996). Recognition by Prototypes. *International Journal of Computer Vision*, vol. 19, pp. 147–168.

[10] Edelman, S. and S. Duvdevani-Bar (1997). A model of visual recognition and categorization. *Phil. Trans. R. Soc. London B*, vol. 352, pp. 1191–1202.

[11] Edelman, S. and H. Bülthoff (1992). Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. *Vision Research*, vol. 32, pp. 2385–2400.

[12] Bülthoff, H., S. Edelman, and M. Tarr (1995). How are three-dimensional objects represented in the brain? *Cerebal Cortex*, vol. 5, pp. 247–260.

[13] Logothetis, N. and D. Scheinberg (1996). Visual object recognition. *Annual Review of Neuroscience*, vol. 19, pp. 577–621.

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

[15] Zhu, S. and A. Yuille (1996). FORMS: A Flexible Object Recognition and Modeling System. *International Journal of Computer Vision*, vol. 20, pp. 1–39.

[16] Murase, H. and S. Nayar (1995). Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, vol. 14, pp. 5–24.

[17] Lades, M., J. Vorbrüggen, J. Buhmann, J. Lange, C. v.d. Malsburg, R. Würtz, and W. Konen (1993). Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, vol. 42, pp. 300–311.

[18] Biederman, I. (1987). Recognition by components: a theory of human image understanding. *Psychol. Review*, vol. 94, pp. 115–147.

[19] Biederman, I. (1985). Human image understanding: recent research and a theory Computer Vision. *Graphics and Image Processing*, vol. 32, pp. 29–73.

[20] Itti, L., C. Koch, and E. Niebur (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis*, vol. 20(11), pp. 1254–1259.

[21] Koch, C. and S. Ullman (1985). Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, vol. 4, pp. 219–227.

[22] Niebur, E. and C. Koch (1994). A model for the neuronal implementation of selective visual attention based on temporal correlation among neurons. *Journal of Computational Neuroscience*, vol. 1, pp. 141–158.

[23] Kestler, H., S. Simon, A. Baune, F. Schwenker, and G. Palm (1999). Object Classification Using Simple, Colour Based Visual Attention and a Hierarchical Neural Network for Neuro–Symbolic Integration. In W. Burgard, T. Christaller, and A. Cremers, eds., *KI–99: Advances in Artificial Intelligence*, Springer Verlag. pp. 267–279.

[24] Smith, A. R. (1978). Color gamut transform pairs. In R. L. Phillips, ed., *5th annual conf. on Computer graphics and Interactive Techniques*. ACM, New York, pp. 12–19.

[25] Roth, M. and W. Freeman (1995). Orientation histograms for hand gesture recognition. Technical Report 94-03, Mitsubishi Electric Research Laboratorys, Cambridge Research Center.

[26] Broomhead, D. and D. Lowe (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, vol. 2, pp. 321–355.

[27] Micchelli, C. (1986). Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions. *Constructive Approximation*, vol. 2, pp. 11–22.

[28] Light, W. (1992). Some Aspects of Radial Basis Function Approximation. In S. Singh, ed., *Approximation Theory, Spline Functions and Applications,* Kluwer, vol. 365 of *Kluwer Mathematical and Physical Sciences Series*. pp. 163–190.

[29] Powell, M. J. D. (1992). The Theory of Radial Basis Function Approximation in 1990. In W. Light, ed., *Advances in Numerical Analysis,* Oxford Science Publications, vol. II. pp. 105–210.

[30] Park, J. and I. W. Sandberg (1993). Approximation and Radial Basis Function Networks. *Neural Computation*, vol. 5, pp. 305–316.

[31] Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE*, vol. 78, pp. 1481–1497.

[32] Cristianini, N. and J. Shawe-Taylor (2000). *An introduction to support vector machines*. Cambridge University Press.

[33] Schölkopf, B., C. Burges, and A. Smola (1998). *Advances in Kernel Methods — Support Vector Learning*. MIT Press.

[34] Vapnik, V. (1998). *Statistical Learning Theory*. John Wiley and Sons.

[35] Friedman, J. (1996). Another approach to polychotomous classification. Tech. rep., Stanford University, Department of Statistics.

[36] Kreßel, U. (1999). Pairwise Classification and Support Vector Machines. In B. Schölkopf, C. Burges, and A. Smola, eds., *Advances in Kernel Methods*, The MIT Press, chap. 15. pp. 255–268.

[37] Weston, J. and C. Watkins (1998). Multi-Class support vector machines. Tech. Rep. CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science.

[38] Nadler, M. and E. Smith (1992). *Pattern Recognition Engineering*. John Wiley and Sons

[39] Schwenker, F., H.A. Kestler, and G. Palm (2000). An algorithm for adaptive clustering and visualisation of highdimensional data sets. In G. Riccia, R. Kruse, and H.-J. Lenz, eds., *Computational Intelligence in Data Mining*, Springer. pp. 127–140.