Active Computer Vision System

D. Paulus, Chr. Drexler, M. Reinhold, M. Zobel, J. Denzler

Lehrstuhl für Mustererkennung (LME, Informatik 5)
Martensstr. 3, Universität Erlangen–Nürnberg, 91058 Erlangen
http://www5.informatik.uni-erlangen.de
Ph: +49 (9131) 85–27894 Fax: +49 (9131) 303811
email: paulus@informatik.uni-erlangen.de

Submitted to

CAMP 2000

(9 pages (limit: 12 pages))

Abstract: We present a modular architecture for image understanding and active computer vision which consists of the following major components: Sensor and actor interfaces required for data–driven active vision are encapsulated to hide machine–dependent parts; image segmentation is implemented in object–oriented programming as a hierarchy of image operator classes, guaranteeing simple and uniform interfaces. We apply this architecture to appearance-based object recognition. This is used for an autonomous mobile service robot which has to locate objects using visual sensors.

Keywords: Computer vision system, object–oriented design, object recognition, scene analysis, mobile systems, object recognition

printed as
D. Paulus, C. Drexler, M. Reinhold, M. Zobel, and J. Denzler. Active computer vision system. In V. Cantoni and C. Guerra, editors, *Computer Architectures for Machine Perception*, pages 18–27, Los Alamitos, California, USA, 2000. IEEE Computer Society.

Contents

List of Figures

Active Computer Vision System

D. Paulus, Chr. Drexler, M. Reinhold, M. Zobel, J. Denzler
Lehrstuhl für Mustererkennung (LME, Informatik 5)
Martensstr. 3, Universität Erlangen–Nürnberg, 91058 Erlangen
http://www5.informatik.uni-erlangen.de
Ph: +49 (9131) 85–27894 Fax: +49 (9131) 303811
email: paulus@informatik.uni-erlangen.de

Abstract

We present a modular architecture for image understanding and active computer vision which consists of the following major components: Sensor and actor interfaces required for data–driven active vision are encapsulated to hide machine–dependent parts; image segmentation is implemented in object–oriented programming as a hierarchy of image operator classes, guaranteeing simple and uniform interfaces. We apply this architecture to appearance-based object recognition. This is used for an autonomous mobile service robot which has to locate objects using visual sensors.

1 Introduction

Conventional autonomous robots can operate and perform their tasks in many cases without visual and audio capabilities. They can navigate using their dedicated sensors and built–in plans. In contrast, service robots which operate in environments where people are present need capabilities to communicate with trained and untrained persons. This is essential for safety reasons as well as for increasing the acceptance of such technical products by the users. Two major modes of human communication are speech and visual interpretation of gestures, mimics, and possibly lip movements. In [1] we described an architecture for knowledge-based recognition of speech as well as images in the context of robotics tasks. In this contribution we elaborate on the visual recognition tasks.

Autonomous mobile systems with visual capabilities are a great challenge for computer vision systems since they require skills for the solution of complex image understanding problems, such as driving a car [27] or exploring a scene [30]. In this contribution we present a vision system that provides mechanisms active computer vision and robotics. The major goal here is to explore a scene with an active camera device which is one task that autonomous mobile systems have to solve. They employ active camera devices to focus on objects or on details. Object recognition is one of the major tasks to be solved in this context. In this contribution we concentrate on appearance-based object recognition methods which have regained attention because of their ability to deal with complex shaped objects with arbitrary texture under varying lighting conditions. While segmentation-based approaches [8, 29, 18] suffer from difficult model generation and unreliable detection of geometric features [17, 18], appearance-based methods are solely based on intensity images without the need for segmentation, neither for model generation nor during the classification stage. In contrast to segmentation which tries to extract only the important information needed for the recognition task, appearance-based approaches retain as much information as possible by operating directly on local features of the image. The input is either the unprocessed image vector or the result of a local feature extraction process. Our work extends the approach of [15, 4, 14] which allows for robust object recognition in the presence of noise and occlusion.

A software system for image understanding and robotics usually has a considerable size. The major problem in software design of general imaging systems is that on the one hand highly run–time efficient code and low–level access to hardware is required, and that on the other hand a general and platform–independent implementation is desired which provides all data types and functions also for at least intermediate–level processing, such as results of segmentation. Today's software engineering is closely coupled with the ideas of object–orientation and genericity which can help simplifying code reuse; if applied properly, object–orientation unifies interfaces and simplifies documentation by the hierarchical structure of classes. Genericity provides an alternative

This work was funded partially by the *Deutsche Forschungsgemeinschaft* (DFG) under grant SFB 603 and Graduiertenkolleg 3D Bildanalyse. Only the authors are responsible for the contents. This work was funded partially by the Bayerische Forschungsstiftung (project DIROKOL)

solution to software engineering problems [13]. Both concepts are available in C++ [25]. Object–oriented programming has been proposed for image processing and computer vision by several authors, in particular in the context of the image understanding environment [9]; this approach is mainly used to represent data. We also use object–oriented programming for operators and devices.

In Sect. 2 we describe techniques to appearance-based object recognition, which is introduced formally. In Sect. 3.1 we outline the general structure of our system and the object–oriented implementation. In Sect. 3.2 we combine object recogntion, robotics tasks, and our software design and outline the object–oriented implementation; results for object recognition are shown in Sect. 4 using an example of a service robot designed for fetch–and–carry services in a hospital. We conclude with a summary and future directions in Sect. 5.

2 Appearance-Based Object Recognition

The most challenging problems in computer vision which are still not solved entierly are especially related to object recognition [28]. Up to now, there have been no general algorithms that allow the automatic learning of arbitrary 3-D objects and their recognition and localization in complex scenes. The term *object recognition* denotes two problems [28]: classification of an object and determination of its pose parameters. By definition, the recognition requires that *knowledge* or *models* of the object are available. The key idea is to compare the image with a model. The key issues thus are the choice of the representation scheme, of the selection of models, and the method for comparison.

We assume that N_{K} object classes Ω_κ $(1 \leq \kappa \leq N_{\mathrm{K}})$ are known and represented as "knowledge" (i.e., models) in an appropriate manner. The representation of the object can be in two dimensions, it may use a full 3–D description, or it can contain a set of 2–D views [26] of a 3–D object. The representaiton of such models is one of the major problems in computer vision and will be discussed in the following. The object models use an object coordinate system and a reference point (mostly on the object) as its origin.

We also assume that an image is given which may contain data in 2–D, $2\frac{1}{2}$–D or 3–D. For intensity images in 2–D it may be either monochrome, color or a multi–channel image. A digital image is mathematically considered as a matrix of discrete values $\boldsymbol{f} = [f_{i,j}] (1 \leq i \leq M, 1 \leq j \leq N)$.

Appearance based object recognition uses non–geometric models representing the intensities in the projected image. Rather than using an abstract model of geometries and geometric relations, images of an object taken from different viewpoints and under different lighting conditions are used as object representation. Figure 1 shows a set of such images. To beat the curse of dimensionality, the images used for object representation are transformed to lower–dimensional feature vectors. This overcomes several problems related to standard approaches as, for example, the geometric modeling of fairly complex objects and the required feature segmentation. Comparative studies prove the power and the competitiveness of appearance based approaches to solve recognition problems [23].

In the following we concentrate on approaches using Eigenspaces. As the given image and the model share the same representation, the choice of the distance function for matching images with models is simpler than for geometric models. We rearrange the image pixels $f_{i,j}$ in an image vector

$$\boldsymbol{f}' = (f_{1,1}, \ldots, f_{1,N}, \ldots, f_{M,1}, \ldots, f_{M,N})^{\mathrm{T}} \quad (1)$$

where the prime character denotes image *vectors* in the following. The elements of $\boldsymbol{f}'$ are denoted by f'_i with $1 \leq i \leq MN$. The comparison of two normalized images $\boldsymbol{f}'_1$ and $\boldsymbol{f}'_2$ with $||\boldsymbol{f}'_i|| = 1$ by correlation simply reduces to the dot product of the image vectors $\boldsymbol{f}'_1$ and $\boldsymbol{f}'_2$

$$s = {\boldsymbol{f}'_1}^{\mathrm{T}} \cdot \boldsymbol{f}'_2 \quad ; \quad (2)$$

the bigger s gets, the more similar are the images $\boldsymbol{f}'_1$ and $\boldsymbol{f}'_2$.

Obviously, high dimensional feature vectors such as this image vector will not allow the implementation of efficient recognition algorithms [17]. The vectors have to be transformed to lower dimensions. Commonly used transforms are the principal component analysis [16, 12, 6] or in more recent publications the Fisher transform [3]. In the following we motivate a linear tranformation $\boldsymbol{\Phi}$ which maps the image vector $\boldsymbol{f}' \in \mathbb{R}^{N \cdot M}$ to a feature vector $\boldsymbol{c} = (c_1, \ldots, c_{L_a})^{\mathrm{T}} \in \mathbb{R}^{L_a}$ with $L_a \ll N \cdot M$ by

$$\boldsymbol{c} = \boldsymbol{\Phi} \boldsymbol{f}' = \boldsymbol{A} \boldsymbol{\Phi}_{\mathbf{t}} \boldsymbol{f}' \qquad \boldsymbol{b} = \boldsymbol{\Phi}_{\mathbf{t}} \boldsymbol{f}' \quad (3)$$

where the linear tranformation $\boldsymbol{\Phi}_{\mathrm{t}}$ maps the image vector $\boldsymbol{f}' \in \mathbb{R}^{N \cdot M}$ to a feature vector $\boldsymbol{b} = (b_1, \ldots, b_{L_a}, \ldots, b_{NM})^{\mathrm{T}} \in \mathbb{R}^{NM}$ and does not reduce the dimension; the matrix $\boldsymbol{A}$ selects the first L_a columns from $\boldsymbol{\Phi}_{\mathbf{t}}$.

If we choose $\boldsymbol{\Phi}$ such that the distance of all features is maximized, this reduces to a problem of eigenvalue computation. From N_a given images written as vectors $\boldsymbol{f}'_1, \ldots \boldsymbol{f}'_{N_a}$ of an object we compute the mean vector

$$\boldsymbol{\mu} = \frac{1}{N_a} \sum_{k=1}^{N_a} \boldsymbol{f}'_k$$

and from this we create a matrix $\boldsymbol{V}$ whose columns are the image vectors

$$\boldsymbol{V} = \left[(\boldsymbol{f}'_1 - \boldsymbol{\mu}) | \ldots | (\boldsymbol{f}'_{N_a} - \boldsymbol{\mu}) \right] \quad . \quad (4)$$

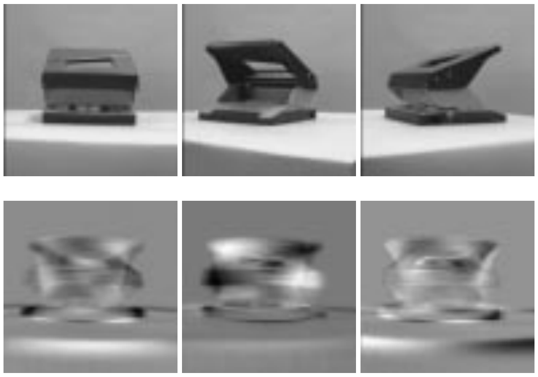

Figure 1. Three different views of an object (upper row), mean vector (lower row, left), and eigenvectors v_0, v_{15} (second row). For the computation 72 views and 360^o rotation in 5^o–steps were used.

Eigenvalue analysis of the matrix $\boldsymbol{K} = \boldsymbol{V}\boldsymbol{V}^{\mathrm{T}}$ yields the eigenvectors $\boldsymbol{v}_1, \ldots \boldsymbol{v}_{N_a}$ sorted by magnitude of the corresponding eigenvalues. A fundamental fact from linear algebra states that an image vector $\boldsymbol{f'}_l$ can be written as a linear combination of the mean image vector and the eigenvectors as

$$\boldsymbol{f'}_l = \boldsymbol{\mu} + \sum_{\nu=1}^{N_a} b_\nu^{(l)} \boldsymbol{v}_\nu \quad .$$

An approximation of $\boldsymbol{f'}$ can be obtained if instead of N_a eigenvectors we select only the first $L_a \leq N_a$ vectors $\boldsymbol{v}_1, \ldots \boldsymbol{v}_{L_a}$. The image vector $\boldsymbol{f'}_l$ is represented by a feature vector

$$\boldsymbol{c}^{(l)} = \left(c_1^{(l)}, \ldots, c_{L_a}^{(l)} \right)^{\mathrm{T}} = \boldsymbol{\Phi} \left(\boldsymbol{f'}_l - \boldsymbol{\mu} \right) \qquad (5)$$

and the columns of the matrix $\boldsymbol{\Phi}$ are the vectors $\boldsymbol{v}_1, \ldots \boldsymbol{v}_{L_a}$.

In the experiments for $N_a = 100$ images we choose only the first $L_a = 15$ eigenvectors. For each object class κ we now record images from different viewpoints and under changing lighting conditions, perform the transformation to eigenspace to obtain a set of vectors

$$\left\{ \boldsymbol{c}^{(\kappa,\nu)}; \nu = 1 \ldots N_{a\kappa} \right\} \qquad (6)$$

and a class–specific matrix $\boldsymbol{\Phi}_\kappa$ for $N_{a\kappa}$ images captured. The recording conditions including the camera position are assumed to be known; they can be set accurately by a camera mounted to a robot or by placing the object on a turn table. The vectors $\boldsymbol{c}^{(\kappa,j)}$ of an object of class κ are a manifold in eigenspace. They are used and stored as the object model C_κ. The processing steps of this approach are exemplified in Figure 1. For pose estimation the ground truth pose parameters of the training images are stored together with the feature vectors. In [16], for example, parametric curves for interpolating the sparse data are used for this. Figure 2 shows an example of a manifold projected onto the first three eigenvectors. Besides manifolds, other object models like Gaussian densities are possible and are currently examined carfully [7].

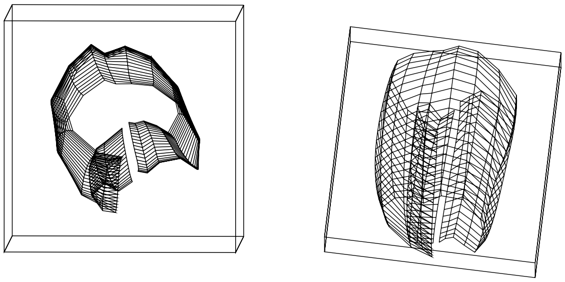

Figure 2. Example of a manifold model with two degrees of freedom generated from views of the punch (Figure 1).

The correlation of two normalized images $\boldsymbol{f'}_i$ and $\boldsymbol{f'}_j$ can now be approximated by the Euclidian distance of two weight vectors $\boldsymbol{c}^{(i)}$ and $\boldsymbol{c}^{(j)}$ which yields a huge gain in computation speed:

$$\| \boldsymbol{f'}_i{}^{\mathrm{T}} \boldsymbol{f'}_j \| \approx 1 - 0.5 \| \boldsymbol{c}^{(i)} - \boldsymbol{c}^{(j)} \| \quad .$$

For the recognition of an object on a given image which has not to be part of the image set used for training, we compute its eigenspace representation to create a vector $\boldsymbol{c}$ using (5). From the manifolds representing the objects we choose the one which has minimal distance $d(C_\kappa, c)$ to the computed vector $\boldsymbol{c}$. Object recognition is thus reduced to the problem of finding the minimum distance between an object and a model. Classification of an image vector $\boldsymbol{f'}$ is then performed according the mapping

$$\delta(\boldsymbol{f'}) := \mathrm{argmin}_\kappa d(C_\kappa, \boldsymbol{\Phi}_\kappa, \boldsymbol{f'}) \quad , \qquad (7)$$

where the function d was chosen here to have three arguments in order to gain flexibility for the distance measure. A rejection class Ω_0 can be introduced by defining a upper bound Θ for the accepted distance. If the distance of a vector $\boldsymbol{c}$ is larger than this threshold for each class, then the vector is assigned to the class Ω_0.

In order to generate an image from a vector $\boldsymbol{c}$, we use the pseudo–inverse $\boldsymbol{\Phi}^+$ of $\boldsymbol{\Phi}$

$$\boldsymbol{\Phi}^+ = \boldsymbol{\Phi}^{\mathrm{T}} \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \right)^{-1} \qquad (8)$$

to create

$$\tilde{\boldsymbol{f'}} = \boldsymbol{\Phi}^+ \boldsymbol{c} + \boldsymbol{\mu} \quad , \qquad (9)$$

which is an approximation of $\boldsymbol{f'}$.

The key to success in this approach is not to create the matrix

$$\boldsymbol{K} = \boldsymbol{V}\boldsymbol{V}^{\mathrm{T}}$$

explicitly when the eigenvectors are computed. For a typical image f of size $N = 256$ and $M = 256$, the image vector $\boldsymbol{f}'$ has length 2^{16}; for $N_a = 100$ images, the matrix $\boldsymbol{V}$ has size $2^{16} \times 100$; the matrix $\boldsymbol{K}$ would thus be of size $2^{16} \times 2^{16}$ and computation of the eigenvectors would be unfeasible. Instead, we use either iterative methods to compute the eigenvectors [19] or we use a result from singular value decomposition. We compute the eigenvalues λ_i and eigenvectors $\boldsymbol{v}'_j$ of the so–called implicit matrix

$$\boldsymbol{K}' = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}$$

which is much smaller than $\boldsymbol{K}$. In our example, the size would be 100×100. We note that

$$\boldsymbol{K}'\boldsymbol{v}'_\nu = \boldsymbol{V}^{\mathrm{T}}\left(\boldsymbol{V}\boldsymbol{v}'_\nu\right) = \lambda_j \boldsymbol{v}'_\nu \quad . \qquad (10)$$

We multiply (10) from left by $\boldsymbol{V}$ and get

$$\boldsymbol{V}\left(\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}\right)\boldsymbol{v}'_k = \left(\boldsymbol{V}\boldsymbol{V}^{\mathrm{T}}\right)\boldsymbol{V}\boldsymbol{v}'_k = \lambda_j\left(\boldsymbol{V}\boldsymbol{v}'_k\right) \quad . \qquad (11)$$

which shows that the eigenvalues of $\boldsymbol{K}'$ are also eigenvalues of $\boldsymbol{K}$ and that the eigenvectors are related by $\boldsymbol{V}$. We use these results to compute the eigenvectors for $\boldsymbol{K}$.

The problem of calculating the feature vector $\boldsymbol{c}_\kappa$ for an image vector $\boldsymbol{f}'$ via (5) is, that elements belonging to occluded or noisy image parts lead to arbitrary errors [14]. The idea is to reformulate the projection problem so that no longer all elements are used but only a subset.

Therefore the pseudo–inverse matrix of $\boldsymbol{\Phi}^{+}_{\kappa}$ introduced in (8) resulting in an equation system of $m = MN$ equations for the L_a unknowns $c_1^{(\kappa)}, \ldots, c_{L_a}^{(\kappa)}$ of $\boldsymbol{c}^{(\kappa)}$

$$\begin{array}{rcl} f'_1 & = & \varphi^{+}_{\kappa,11} c_1^{(\kappa)} + \ldots + \varphi^{+}_{\kappa,1n} c_n^{(\kappa)} + \mu_1 \\ & \vdots & \\ f'_m & = & \varphi^{+}_{\kappa,m1} c_1^{(\kappa)} + \ldots + \varphi^{+}_{\kappa,mn} c_n^{(\kappa)} + \mu_m \end{array} \qquad (12)$$

with $\boldsymbol{\Phi}^{+}_{\kappa} = [\varphi^{+}_{\kappa,\sigma\tau}](1 \leq \sigma \leq m, 1 \leq \tau \leq L_a)$.

Based on the observation that in the absence of interferences it would be sufficient to choose $r_{\min} = L_a$ independent equations out of the m from this equation system to compute a solution for the L_a components of the feature vector $\boldsymbol{c}_\kappa$, an approximation $\tilde{\boldsymbol{c}}_\kappa$ can be calculated by choosing a set $\mathcal{S} = \{s_1, \ldots, s_r\}$ with $L_a \leq r \ll m$ and solving

$$\begin{array}{rcl} f'_{s_1} & = & \varphi^{+}_{\kappa,s_1 1} \tilde{c}_1^{(\kappa)} + \ldots + \varphi^{+}_{\kappa,s_1 n} \tilde{c}_n^{(\kappa)} + \mu_{s_1} \\ & \vdots & \\ f'_{s_r} & = & \varphi^{+}_{\kappa,s_r 1} \tilde{c}_1^{(\kappa)} + \ldots + \varphi^{+}_{\kappa,s_r n} \tilde{c}_n^{(\kappa)} + \mu_{s_r} \end{array} \qquad (13)$$

in the least square sense for $\tilde{\boldsymbol{c}}_\kappa$ using singular value decomposition (SVD).

The set of chosen equations for $\boldsymbol{f}'_{s_\iota}, s_\iota \in \mathcal{S}$ can be partitioned into $\mathcal{S}_o$, for which $f'_{s_\iota}, s_\iota \in \mathcal{S}_o$ are undisturbed object pixels, and $\mathcal{S}_b$, which represents background pixels and outliers. The approximation for $\tilde{\boldsymbol{c}}_\kappa$ according to (13) can only be adequate if $|\mathcal{S}_o| > |\mathcal{S}_b|$ holds. To achieve this, [15] suggests to generate a number H of hypotheses ${}^t\mathcal{S}, 1 \leq t \leq H$ for each class Ω_κ by generating the elements ${}^t s_\iota$ on a random basis and to compute

$${}^t\tilde{\boldsymbol{f}}' \quad = \quad {}^t\boldsymbol{\Phi}_\kappa {}^t\tilde{\boldsymbol{c}}_\kappa + \boldsymbol{\mu} \qquad (14)$$

for each hypothesis. For noisy images, the simple distance measure defined by (2) turns out to be insufficient because all components of the feature vector are weighted equally, whereas the components belonging to vectors with smaller eigenvalues are more sensitive to noise. This is the reason, why we choose three arguments in (7). Any distance to the feature vector can be chosen here.

While this random selection scheme works fine for compact objects, e.g. those for which the ratio of object to background pixels within the bounding box is considerably high, it fails for objects which occupy only a small part of the bounding box in the image as the probability of getting a sufficient amount of good object points for the generation of hypotheses is low. By incorporating additional knowledge about object properties the initial selection scheme can be improved if only pixels are regarded as possibly good candidates if object specific conditions like local texture features or color, are fulfilled. Up to know, only the average object intensity is used for restricting the point selection.

3 Implementation

Various modules for common computer vision algorithms are provided our software environment. These modules were implemented for several applications. This flexibility first requires additional effort for portable software. On the long run it reduces the effort of software maintenance.

3.1 System Architectures

The need for a flexible, knowledege-based computer vision system with real–time capabilities at least for low–level processing lead to "**An im**age **a**na**l**ysis **s**ystem" (ANIMALS, [20, 22, 21]) which is implemented in C++. It provides modules for the whole range of algorithms from low–level sensor and actor control up to knowledge-based analysis.

The general problem of image analysis is to find the *optimal* description of the input image content which is appropriate to the current problem. Sometimes this means that the most precise description of the image data has to be found, in other cases a less exact result which can be computed faster will be sufficient. In many image analysis problems, objects have to be found in the images and described by terms that fit to the application.

These general problems can be divided into several sub–problems. After an initial preprocessing stage, images are usually segmented into meaningful parts. Various segmentation algorithms create initial symbolic descriptions of the input data [18] which we call *segmentation objects* [20]. A segmentation object contains sets of features, such as points, lines, or more complex geometric structures; it can also contain and administrate relations between such features such as parallelism of lines or adjacency of points. Models in a knowledge base containing expectations on the possible scene in the problem domain are matched with segmentation objects in order to provide a final symbolic description. This is achieved best if the representation for the models is similar to the structure of segmentation results. If no segmentation is required or desired, the segmentation stage is replaced by a feature extraction algorithm; the data representation for the resulting feature sets is easily managed by the segmentation object as well.

Modern architectures for image analysis incorporate active components such as pan/tilt devices or cameras on a robot. Such devices lead to feed–back loops in the course from image data to image descriptions. A top–level view of the main components in our image analysis system is shown in Figure 3; data is captured and digitized from a camera and transformed to a description which may cause changes in camera parameters or tuning of segmentation parameters. Models which are collected in the knowledge base are created from segmentation results or at least have similar structure.These models are used for the analysis. Image processing tasks are shown in oval boxes; data is depicted as rectangles.

The dotted lines in Figure 3 indicate that a control problem has to be solved in active vision or active exploration resulting in a closed loop of sensing and acting. Information is passed back to the lower levels of processing and to the input devices; this way, parameters of procedures can be changed depending on the state of analysis, or the values of the camera and lens can be modified.

The algorithms and data structures of our system are implemented in a **Hi**erarchy of **P**icture **P**rocessing **O**bject**S** (HIPPOS, written as ἵππος [20, 22]), an object–oriented class library designed for image analysis. In [22], the data interfaces were defined as classes for the *representation* of segmentation results. The *segmentation object* [22] provides a uniform interface between low–level and high–level processing; its components can be arbitrary objects resulting from point, line, region or surface segmentation. Images, segmentation objects, and additional classes for representation of segmentation results such as e.g. chain codes, lines, polygons, circles, etc. are derived from one common base class which unifies all interfaces. In [10, 21]this system is extended to a hierarchical structure of *image processing and analysis classes* and *objects* (cmp. [5]). Objects are the actual algorithms with specific parameter sets which are also objects. Classes as implementation of algorithms are particularly useful, when operations require internal tables which increase their efficiency since tables can then be easily allocated and handled.

For appearance-based object recognition we need camera classes with attached actors in order to change the camera position. As shown below, the segmentation objects are relatively simple in this case; they contain merely vectors of real numbers. The structure of models, in this case, is similar to segmentation objects. The description after recognition contains the detected object class as well as the pose estimation. Feedback is required if the recogniton is not successful and a change in the viewing direction is required in order to initiate a new recognition sequence.

For robotics tasks we need other interfaces which are also defined by classes. Care has to be taken that actors which are commonly related to the vision hierarchy, such as pan/tilt axes of an active camera, are equipped with similar syntax as axes on the robot. Only then, imaging algorithms can be easily integrated into the robotics application: a pan movement can then be performed alternativley by panning the camera or turning the robot in place.

The general class structure of the system provides disjoint packages for commandline and graphical interfaces, matrix and vector classes, image and image related data, image processing and image analysis classes, sensors such as cameras, actors such as camera stepper motors, and robotics such as motion commands. For efficiency reasons, all implementation is done in C and C**++**. All software has been tested under Linux, IRIX, and HPUX. Various modules for common computer vision algorithms are provided in ANIMALS. These modules were implemented for several applications. Since all segmentation algorithms use the common interface for data exchange provided by the segmentation objects, the compatiblity is high. This flexibility first requires additional effort for portable software. On the long run it reduces the effort of software maintenance.

3.2 Image Data and Algorithms

The description of appearance-based methods requires that a two–dimensional image is accessed as a one–dimensional vector (1). Whereas conventional image processing systems only use the latter notion, our image classes mentioned in Sect. 3.1 provide both, access by one or by two indices, which can be either checked for the validity of the range, or not (if high execution speed is required). The relatively simple idea for the implementation is to provide parametric vector and matrix classes. Matrices are composed of vector classes which do not allocate their memory themselves, but reuse the already allocated continuous memory from the matrices; details are given in [20, Chap. 11].

By inheritance, these generic vector classes are equipped with numeric operations such as addition and scalar product. Matrices define methods for multiplication by vectors and matrices. Matrix and vector objects can thus be used for the implementation of most of the equations in Sect. 2.

Several methods and programs can be used to determine the eigenvectors of the matrix $\boldsymbol{V}$ in (4). We equip the numeric matrix object with a common interface for eigen analysis and internally switch to different algorithms for the solution of the eigen system.

In the experiments it is required to select an optimal distance measure between model (in this case manifold) and feature set. This is denoted by d in (7). Naturally, in C++ we implement this by a virtual function and do experiments with different distance measures without chaning the classification scheme. Other classification algorithms could be selected as well from a hierarchy of classifcation classes [11].

3.3 Sensors and Actors

Due to the variability of the hardware connected with an image analysis software system, interfaces to several types of frame grabbers, cameras, lenses, stereo devices, etc. have to be provdided by the system. In order not to burden the programs by a large number of switches, all these special devices are encapsulated as classes which share their common behaviour in a base class. Whereas the internal implementation may be sophisticated and thereby provides the required performance, the user interfaces for these classes are simple in C++.

The computation of (6) requires that a set of images is recorded with known parameters for the viewing direction. This requires the notion of rotation axes which yields the idea of an axis class. Several techical realizations are used to record such a set of images. Either we rotate a turn table, or we move a robot arm connected to a camera, or we move the autonomous robot on which we place the camera. From the algorithmic view, the problem remains the same. We simply need transformations of the image and camera coordinate systems.

Using a class hierarchy as outlined in Figure 4, not only the algorithms are similar, but also the implementation.[1] As axes, motors, and geomeric transformations are derived from a common base class, they are *forced* into a common interface syntax. Only so it is possible to have an implementation which is almost independent of the actors used. The syntax of those motors and axes has also be used to access robot motors.

4 Results

The methods have been tested on typical objects from office environments and on objects commonly used in hospitals; examples drawn from the training set are depicted in Figure 5. Object scale, translation and one rotational degrees of freedom has been estimated during the tests. Only those images have been used for the tests, which have not been included into the training set. In addition, Gaussian noise was added to the images for one test set. Another test was performed where parts of the objects were randomly occluded; a large area of the image was masked out for that purpose. Examples for test images are shown in Figure 6. The test images were of size 256^2 whereas the bounding boxes of the objects differed from $122x87$ to $185x131$, which corresponds to size of the training images.

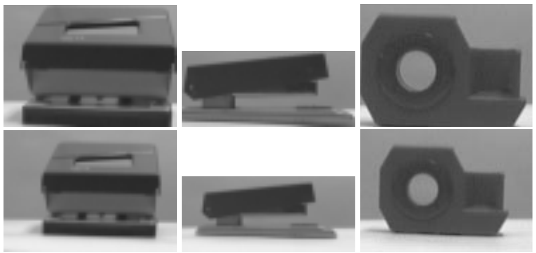

Figure 5. Objects from the sample set (different scales)

Number of training images	355
Number of test images	60
Number of classes	3
Number of eigenvectors used	20

Table 1. Sample set of office data, also used in [24]

For the estimation of the object translation within the scene, a hierarchical approach is used. The 256^2 scenes have been subsampled two times by a factor of three resulting in images of size 28^2 at the lowest resolution. For each imagepoint at this resolution, a confidence value for beiing an object location is calculated. This is done by using subsampled version of the

[1] Currently, we use a TRC head mounted to our XR 4000 robot. For training, Canon, or Sony cameras and a turn table or camera on a hand of a stationary robot are used.

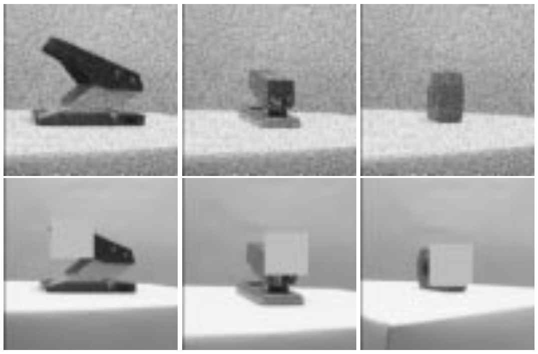

Figure 6. Expamles from the test set. Top line: Gaussian noise added, bottom line: synthetic occlusion of large parts of the objects

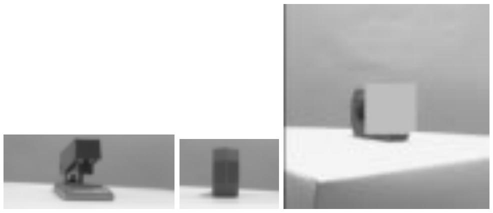

Figure 7. Examples of objects subject to confusion in the recognition module. The object in the middle shows a tape roll. The same object from another view is occluded on the right side. Due to the lack of information, the stapler (left) has a higher confidence.

un– disturbed	Gaussian noise added	occlusion
91.67%	83.33%	50.00%

Table 2. Averaged classification results.

	reject	0	1	2
0	0	16	4	0
1	3	0	11	6
2	5	0	12	3

Table 3. Absolute confusion matrix for occluded test images (left column: number of actual class, top row: number of assigned class, or reject).

eigenspace models. Only areas with high confidence values are searched on the higher resolution levels.

After calculating a confidence map for each class, the class and location with the highest confidence is choosen. The pose and scale estimation is given implicetly by nearest model point of the choosen class. Table 2 gives the results achieved for the selected objects.

The main problem is the mixing up between objects 1 and 2 (duct tape disposer and stapler). Rejected cases result from correct classification but a wrong pose estimation. The high number of wrong assignments of class 2 to 1 is due to fact that for small objects (like the duct tape disposer) in some cases almost the complete object was occluded, as can be seen in Figure 7.

The search and classification time depends on the number of hypotheses generated per class, the number of points initially used for each hypothesis, and the number of object classes. For measuring computation times, 5 hypotheses have been generated per image, 70 initial points were selected per hypothesis, and 3 object classes.

On a personal computer with a PentiumII processor running at 700 MHz, it took about 8 seconds for the search, confidence calculation and pose estimation of one object with a 185×131 bounding box within a 256^2 image using a resolution hierarchy (2 level deep, subsampling factor of 3). Most of the time is spent on calculating the SVD for approximating $\tilde{c}_\kappa$ (about 55%). The other time consuming part is the pointwise search at the lowest resolution hierachy level for possible object locations. Faster “region–of–interest” operators are currently studied, like “histogram backprojection”, for circumventing this search.

5 Conclusion and Future Work

We described our system architecture for active image understanding which is implemented in an object–oriented fashion. Classes and objects encapsulate data, devices, and operators. We argued that this programming paradigm simplifies solving image analysis tasks and gave examples for the expressional power of the proposed approach, especially when applied to unusual areas such as device interfaces. Object–oriented programming is preferred to genericity for hierarchies of operator classes; assuming that the task of an operator is not trivial, the overhead imposed by this implementation scheme is negligible. Genericity is used for regular data structures such as for pixel access in image matrices.

In our application for appearance-based object recognition we successfully applied the architecture. The use of ANIMALS as the implementation basis allows easy incorporation of new algorithms in cases where the appaerance based approach fails, e.g an aktive viewpoint selection scheme will be incorporated for disambiguating object views. Up to now, only graylevel images are used for the classification task. The object–oriented design makes it possible to augment the object models with information about the object color and this models may be used in the existing algorithms without any changes.

References to other applications have been given.

References

[1] U. Ahlrichs, J. Fischer, J. Denzler, Ch. Drexler, H. Niemann, E. Nöth, and D. Paulus. Knowledge based image and speech analysis for service robots. In *Proceedings Integration of Speech and Image Understanding*, pages 21–47, Corfu, Greece, 1999. IEEE Computer Society.

[2] R. B. Arps and W. K. Pratt, editors. *Image Processing and Interchange: Implementation and Systems*, San Jose, CA, 1992. SPIE, Proceedings 1659.

[3] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.

[4] H. Bischof and A. Leonardis. Robust recognition of scaled eigenimages through a hierarchical approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 664–670, June 1998.

[5] I. C. Carlsen and D. Haaks. IKS^{PFH} — concept and implementation of an object–oriented framework for image processing. *Computers and Graphics*, 15(4):473–482, 1991.

[6] Y.T. Chien and K.S. Fu. Selection and ordering of feature observations in a pattern recognition system. *Information And Control*, 12:395–414, 1968.

[7] F. Deinzer, J. Denzler, and H. Niemann. Classifier Independent Viewpoint Selection for 3-D Object Recognition. In G. Sommer, editor, *Mustererkennung 2000*, September 2000. accepted.

[8] O. Faugeras. *Three–Dimensional Computer Vision – A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[9] R. M. Haralick and V. Ramesh. Image understanding environment. In Arps and Pratt [2], pages 159–167.

[10] M. Harbeck. *Objektorientierte linienbasierte Segmentierung von Bildern*. Shaker Verlag, Aachen, 1996.

[11] J. Hornegger. *Statistische Modellierung, Klassifikation und Lokalisation von Objekten*. Shaker Verlag, Aachen, 1996.

[12] K. Karhunen. Über lineare Methoden in der Wahrscheinlichkeitsrechnung. *Ann. Acad. Sci. Fenn.*, Ser. A I:37, 1947.

[13] U. Köthe. Reusable components in computer vision. In B. Jähne, H. Haussecker, and P. Geissler, editors, *Handbook of Computer Vision and Applications* , pages 103–132. Academic Press, London, 1999.

[14] A. Leonardis and H. Bischof. Dealing with occlusion in the eigenspace approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 453–458, 1996.

[15] A. Leonardis and H. Bischof. Robust recovery of eigenimages in the presence of outliers and occlusion. *Internationl Journal of Computing and Information Technology*, 4(1):25–38, 1996.

[16] 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, January 1995.

[17] H. Niemann. *Klassifikation von Mustern*. Springer, Heidelberg, 1983.

[18] H. Niemann. *Pattern Analysis and Understanding*, volume 4 of *Springer Series in Information Sciences*. Springer, Heidelberg, 1990.

[19] E. Oja and J. Parkkinen. On Subspace Clustering. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 692–695. San Diego, 1984.

[20] D. Paulus and J. Hornegger. *Applied pattern recognition: A practical introduction to image and speech processing in C++*. Advanced Studies in Computer Science. Vieweg, Braunschweig, 2 edition, 1998.

[21] D. Paulus, J. Hornegger, and H. Niemann. Software engineering for image processing and analysis. In B. Jähne, P. Geißler, and H. Haußecker, editors, *Handbook of Computer Vision and Applications*, volume 3, pages 77–103. Academic Press, San Diego, 1999.

[22] D. Paulus and H. Niemann. Iconic–symbolic interfaces. In Arps and Pratt [2], pages 204–214.

[23] J. Ponce, Zisserman, and M. Hebert, editors. *Object Representation in Computer Vision*, volume 1144 of *Lecture Notes in Computer Science*, Heidelberg, 1996. Springer.

[24] J. Pösl and H. Niemann. Wavelet features for statistical object localization without segmentation. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 3, pages 170–173, Santa Barbara, California, USA, October 1997. IEEE Computer Society Press.

[25] B. Stroustrup. *The C++ Programming Language, 3^{nd} edition*. Addison-Wesley, Reading, MA, 1997.

[26] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

[27] F. Thomanek and E.D. Dickmanns. Autonomous road vehicle guidance in normal traffic. In *Second Asian Conference on Computer Vision*, pages III/11–III/15, Singapore, 1995.

[28] E. Trucco and A. Verri. *Introductory Techniques for 3–D Computer Vision*. Prentice Hall, New York, 1998.

[29] F. C. D. Tsai. Using line invariants for object recognition by geometric hashing. Technical report, Courant Institute of Mathematical Sciences, New York, February 1993.

[30] L. Wixson. Gaze Selection for Visual Search. Technical report, Department of Computer Science, College of Arts and Science, University of Rochester, Rochester, New York, 1994.

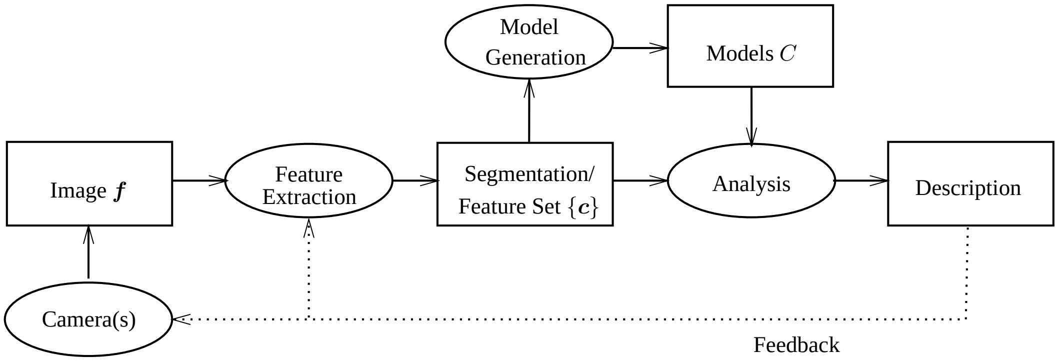

Figure 3. Data flow for apperance-based object recognition

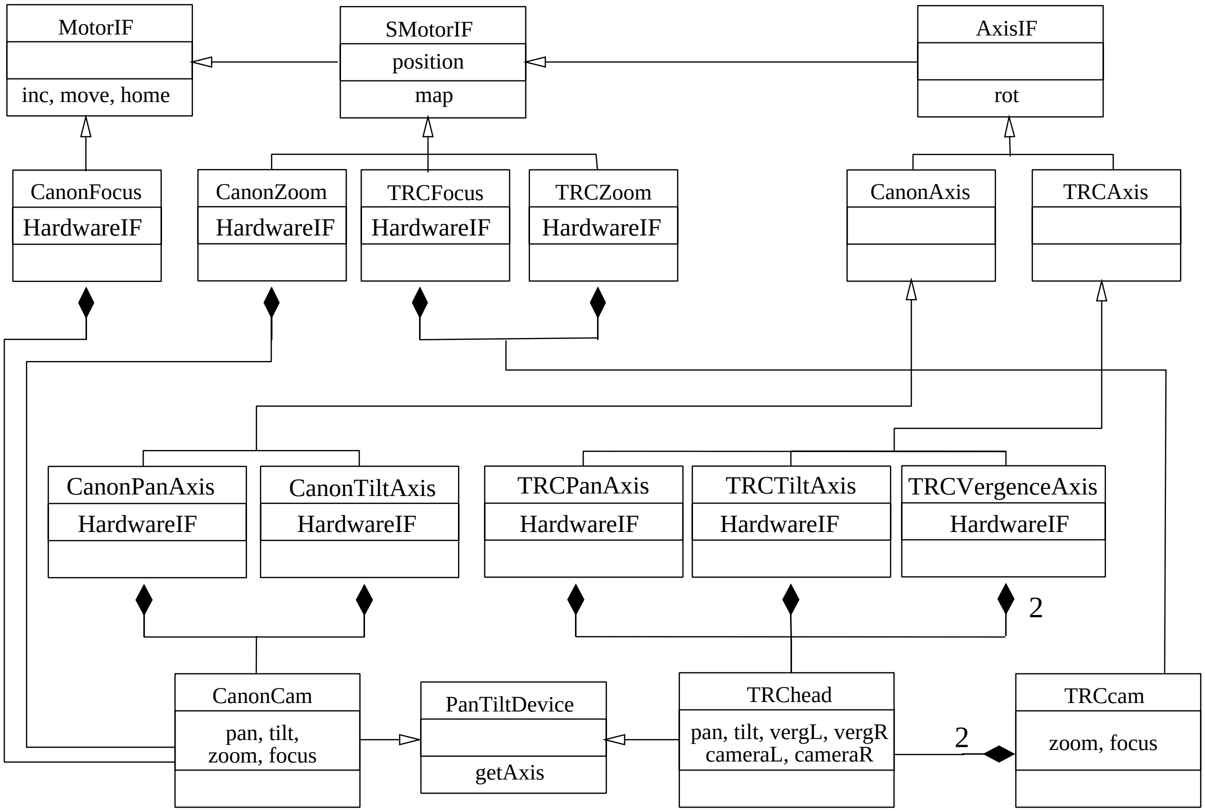

Figure 4. Class hierarchy for motors used in active vision systems