Learning Manipulation Skills by Combining PCA Technique and Adaptive Interpolation

Jianwei Zhang and Alois Knoll

Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany

ABSTRACT

We propose an approach for learning manipulation skills of robot arms based on complex sensor data. The first component of the used model serves as a projection of high-dimensional input data into an eigenspace by using Principal Component Analysis (PCA). Complex sensor data can be efficiently compressed if robot movements achieving optimal manipulation tasks are constrained to a local scenario. The second component is an adaptive B-spline model whose input space is the eigenspace and whose outputs are robot motion parameters. In the offline learning phase, an appropriate eigenspace can be built by extracting eigenvectors from a sequence of sampling sensor patterns. The B-spline model is then trained for smooth and correct interpolation. In the online application phase, through the cascaded two components, a sensor pattern can be mapped into robot action for performing the specified task.

This approach makes tasks such as visually guided positioning much easier to implement. Instead of undergoing cumbersome hand-eye calibration processes, our system is trained in a supervised learning procedure using systematical perturbation motion around the optimal manipulation pose. If more sensors or some robust geometric features from the image processing are available, they can also be added to the input vector. Therefore, the proposed model can integrate multiple sensors and multiple modalities.

Our experimental setup is a two-arm robotic system with "self-viewing" hand-eyes and force/torque sensors mounted on each parallel jaw gripper. Implementations with one-hand grasping and two-hand assembly based on visual and force sensors show that the method works even when no robust geometric features can be extracted from the sensor pattern.

Keywords: sensor-based skills, hand-eye, task based mapping, learning

1. INTRODUCTION

Grasping and assembly are two of the most important and most demanding sensor-based manipulation skills. While multifinger grasping can be viewed as a complex planning problem, two-finger grasping of rigid objects poses problems of hand-eye coordination and sensor-guided positioning. Given an object whose model is unknown, the fine-motion of a gripper should be exactly controlled based on dynamic sensor data so that it can move from any location in the vicinity of the object to the optimal grasping position. The assembly skills, e.g. inserting and screwing, can be de decomposed into force control techniques and vision-based fine-positioning.

The traditional methods of sensor-guided fine-positioning are based on hand-eye calibration. Such methods work well if the hand-eye configuration is strictly fixed and the geometric features of the grasping position can be extracted robustly. However, the hand-eye calibration matrix cannot be interpreted as an adequate cognitive model of human grasping.

Recently, neural network based learning has also found applications in grasping,[4,8,3] which use geometric features as input to the controller. Since the image processing procedures such as segmentation, feature extraction and classification are not robust in real environments and since these processing algorithms are computationally expensive, some of the work has to use marked points on the objects to be grasped. It is desirable that a general control model can be found which transforms raw image data directly to action values. If such a model can be found it may be studied whether it bears similarities with the cognitive abilities of humans.

Learning of vision-based positioning based on visual appearance information was introduced.[5] A parametric eigenspace representation is used for describing the different objects as well as object locations. The positioning problem is thus transformed into finding the minimum distance between a point and a manifold in the eigenspace.

E-mail: zhang|knoll@techfak.uni-bielefeld.de

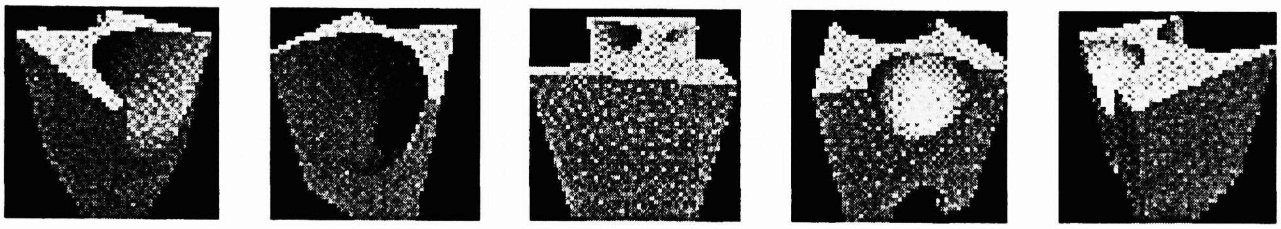
In this paper, we propose a solution by combining the PCA (*principal component analysis*) technique with an adaptive fuzzy controller. A "self-viewing" hand-mounted camera provides visual feedback to the fine-motion control. In the following sections, we first introduce the models we employed and explain the relation with human sensorimotor control. In the experimental parts two applications are presented: one is grasping based on the hand-eye data, the other one is assembly using a two-arm system.

2. CONCEPT OF BUILDING THE SENSING-ACTION MODEL

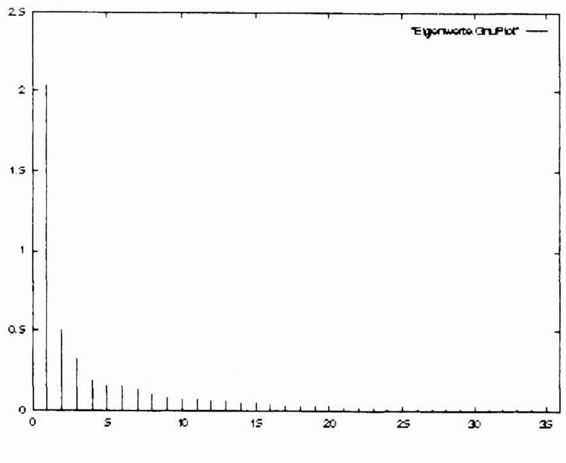
2.1. Dimension Reduction by Using Observation Data

It is well-known that general neural or fuzzy systems with a large number of input variables suffer from the problem of the "curse of dimensionality". For instance, an image has 192×144 pixels, each pixel is measured by brightness level. If no extra image processing is performed, then a control system with about 26,000 input variables needs to be modelled. The outputs of the system are the motion values for a robot (mobile robot as well as robot arm). Is it possible to build a neuro-fuzzy model to map the inputs to the outputs?

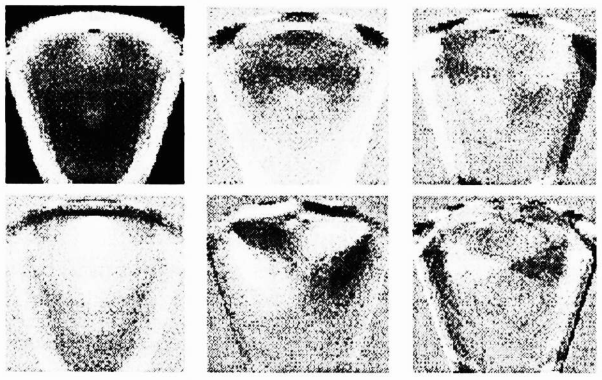
If the input vectors are considered as a stack of vectors of training data originating from a continuously varying process, statistical indices can be extracted. These indices describe data distribution, variances, and the most interesting features for control – eigenvectors (see Fig. 1 for an example) and principal components.



(a) A sequence of brightness images taken by rotating the viewing point of the camera.



(b) Eigenvalues.



(c) The first six eigenvectors.

Figure 1. An example of extracting eigenvectors from a sequence of training images.

For an input space $X_1 \times X_2 \times \cdots \times X_m$, if all the variables x_1 to x_m may vary in all their universes during the observed sampling procedure, input data will be scattered over the whole input space. Nevertheless, in a high-dimensional input space, if the observed process runs continuously, the input vector varies only gradually. Under an appropriately separated observation, e.g. within a local scenario of the robot environment, the input vectors possess a large grade of similarity. In other words: the observed input data are correlated to a large degree in the high-dimensional input space.

If certain correctly selected original input variables happen to be the axes along which the sampling data aggregate they can be directly used as fuzzy controller inputs. Otherwise, the information from the unselected variables will be lost.

2.2. Projection in Eigenspace

A well-known technique for dealing with multivariate problems in statistics is PCA. Until now, it is mainly applied in data compression and pattern recognition.[6] This technique is also suitable for reducing the dimension of the input space of a general control problem.

An eigenvector, noted as $\vec{a}_i$, is computed as $[a_{1,i}, a_{2,i}, \ldots, a_{m,i}]^T$. The eigenvectors form an orthogonal basis for representing the original individual sensor patterns. Assume that the eigenvectors $\vec{a}_1, \vec{a}_2, \ldots$ are sorted according to their eigenvalues in a descending order. An eigenspace with a reduced dimension n can be formed with the first n eigenvectors. $\vec{a}_i$ accounts for the ith dimension in the eigenspace. The projection of an input vector $X = [x_1, x_2, \ldots, x_m]$ on eigenvector $\vec{a}_i$, called the ith principal component, is $p_i = a_{1,i}x_1 + a_{2,i}x_2 + \cdots + a_{m,i}x_m$. The complete projection can be represented as:

$$[\vec{a}_1, \ldots, \vec{a}_n]^T \cdot X = [p_1, \ldots, p_n]^T$$

All projections of the sample data sequence form a manifold in the eigenspace.

Depending on how "local" the measuring data are and therefore how similar the observed sensor patterns look like, a small number of eigenvectors can provide a good summary of all input variables. It is possible that two or three eigenvectors supply the most information indices of the original input space. In very high-dimensional cases, an efficient dimension reduction can be achieved by projecting the original input space into the eigenspace. This step is illustrated in the left part of Fig. 2.

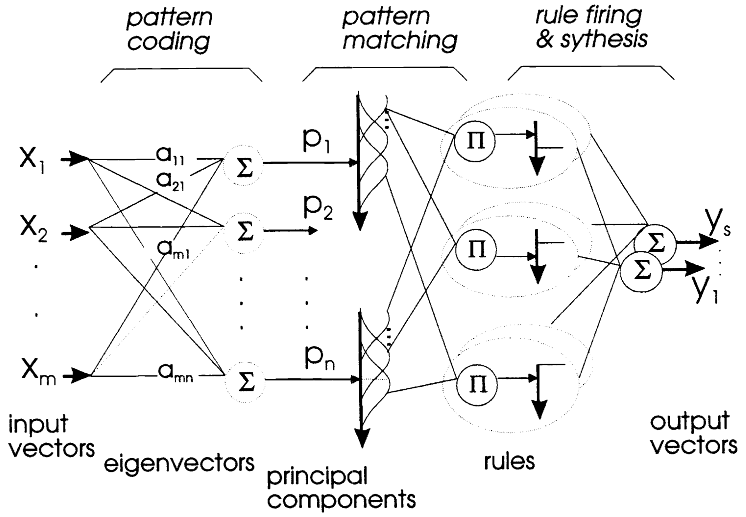


Figure 2. The task based mapping can be interpreted as a neuro-fuzzy model. The input vector consists of pixels of a brightness image.

The eigenvectors of a covariance matrix can be efficiently computed by the Jacobian[2] or perceptron approach.[7] The perceptron approach has three advantages over the Jacobian method if the number of the eigenvectors are pre-selected. Firstly, the Jacobian method calculates all eigenvectors of the covariance or explicit covariance matrix of the input data; the perceptron method only calculates the first n eigenvectors, thus saves computation time. Secondly, Jacobian requires the construction of the covariance matrix of the input data. With large input data sets this is often critical regarding the amount of memory needed. Thirdly, the eigenvectors found with the perceptron method are already in descending order.

2.3. Adaptive Interpolation

If the dimension is high, the whole sensor space cannot be directly partitioned into fuzzy regions. However, with PCA, a fuzzy partition is possible in the eigenspace.

Therefore, a control rule looks like:

"If the current input is similar with $Sensor\_pattern_i$ Then the output value should be y_i"

The $Sensor\_pattern_i$ can be approximately reconstructed using the principal components. This procedure is comparable with the *vector quantisation.*

Partitioning of eigenvectors can be done by covering eigenvectors with linguistic terms, see the right part of Fig. 2. In the following implementations, fuzzy controllers constructed according to the B-spline model are used.[9] This model provides an ideal implementation of CMAC proposed by Albus.[1] We define linguistic terms for input variables with B-spline basis functions and for output variables with singletons. Such a method requires less parameters than the other set functions such as trapezoid, Gaussian function, etc. The output computation becomes very simple and the interpolation process is transparent. We also achieved good approximation capability and rapid convergence of B-spline fuzzy controllers.[9] An illustrative example is shown in Fig. 3.

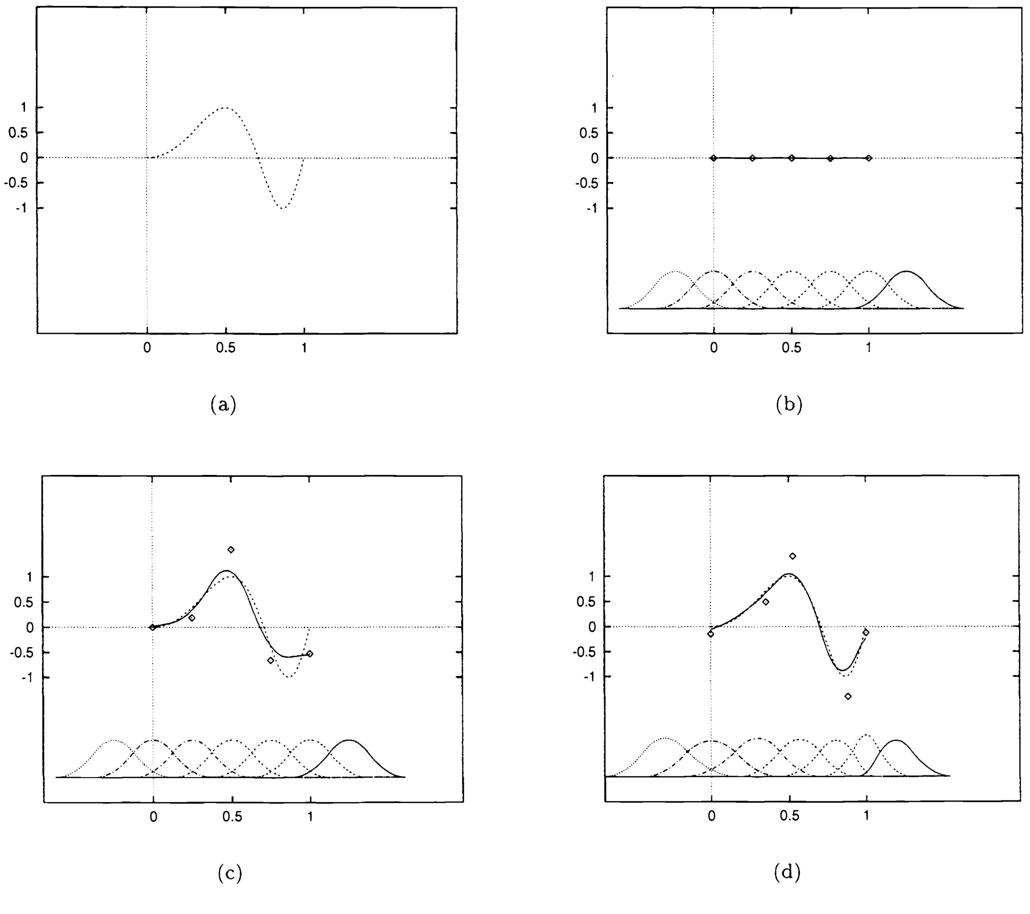


Figure 3. Approximation of the function $y = \sin(2\pi x^2)$. The horizontal axis represents x, covered with B-spline basis functions of order 3, the vertical axis represents the output value y. (a) The function $\sin(2\pi x^2)$. (b) The initial controller output set as zero. (c) The output curve after 2000 epochs without knot adaptation. (d) The output curve after 2000 epochs with knot adaptation.

In the online application, the input data are first projected into the eigenspace, then mapped to output based on the fuzzy control model, Fig. 2.

3. THE APPROACH

By summarising the above ideas and formulating them stepwise, it turns out that constructing a controller for a multivariate problem consists of two phases. The first phase comprises sampling and analysing training data. The second phase is a supervised learning algorithm for a fuzzy controller (if the desired output data as well as the input data are available). Once the fuzzy controller has been constructed the outputs can be calculated through eigenspace projection and fuzzy rule synthesis.

3.1. Phase 1: Sampling Training Data and Analysis

This step aims at evaluating the statistical indices and at performing the dimension reduction. It is desirable that all representative values of input data be available for the sampling process.

1. Sample input data, record the desired output values (if available).
2. Pre-process the input data: normalise and subtract the mean value.
3. Stack the input variables into vectors.
4. Compute the eigenvectors and eigenvalues, e.g. with the Jacobian method.

To improve the results of the Eigenspace transformation, all images are normed, so that their energy is equal. This preprocessing is done by the following formula:

$$g_i(j) = c\frac{\sqrt{dim}}{\sqrt{\sum_{k=1}^{dim}(f_i(k))^2}}f_i(j)$$

where

- $f_i(j)$ is the intensity of the j-th pixel in the i-th image.
- $g_i(j)$ is the intensity of the j-th pixel in the corresponding normed image.
- dim is the number of pixels in the image.
- c is a constant which is usually so chosen that $g_i(j)$ can be represented in integer arithmetic.

Interpreting the k training images as vectors $\vec{b_i}$ that come from a pattern-generating process, we can compute the mean value of these vectors:

$$\vec{m} = \frac{1}{k}\sum_{i=1}^{k}\vec{b_i}$$

3.2. Phase 2: Training the Fuzzy Controller

By adapting the supervised learning algorithm presented in our earlier work,[9] the training procedure looks as follows:

1. Select the n eigenvectors with the largest n eigenvalues, noted as $\vec{a}_1, \dots, \vec{a}_n$.
2. Select the order of the B-spline basis functions for each eigenvector.
3. Determine the knots for partitioning each eigenvector.
4. Initialise the control vertices for the output.
5. Learn the control vertices using the gradient descent method.
6. If the results are satisfactory: terminate.
7. Modify the knots for eigenvectors, go to 4.

3.3. Online Application

In the case of supervised learning, each learning datum corresponds to a supporting point in the control space. If a sensor pattern is taken online and its eigenvalues are calculated, the computation of the controller outputs may then be regarded as the "blending" of all the firing rules.

The following steps are necessary:

1. Update the input data $(x_1, x_2, \ldots, x_m)$;
2. Pre-process the data - normalise and subtract mean;
3. Project the input into the eigenspace $(p_1, \cdots, p_n)$;
4. Compute the output by feeding the projection vector (principal components) into the fuzzy controller trained in section 3.2.

The offline and online phases can be illustrated by Fig. 4. The following grasping and hole-searching experiments are based on such a flowchart.

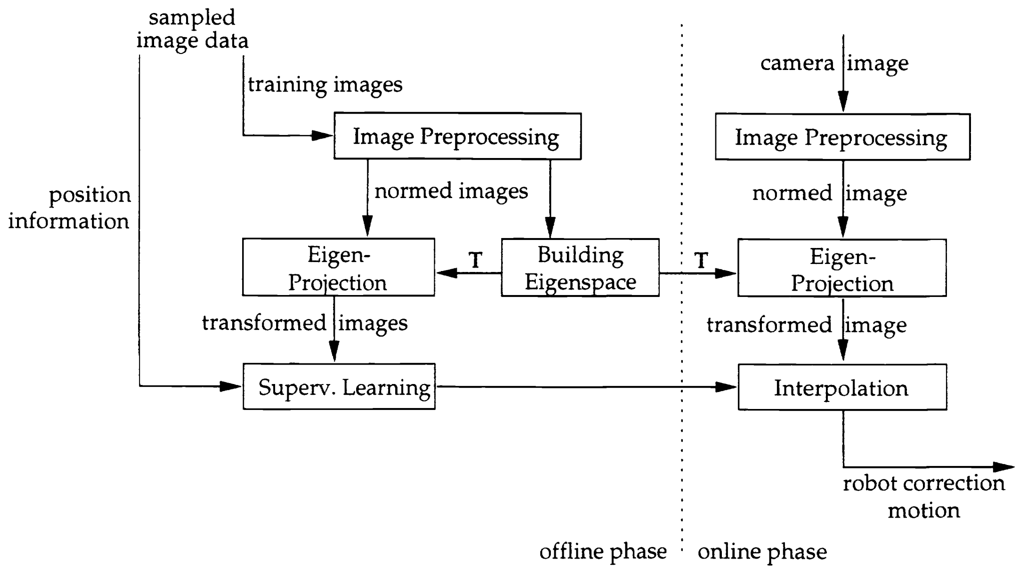


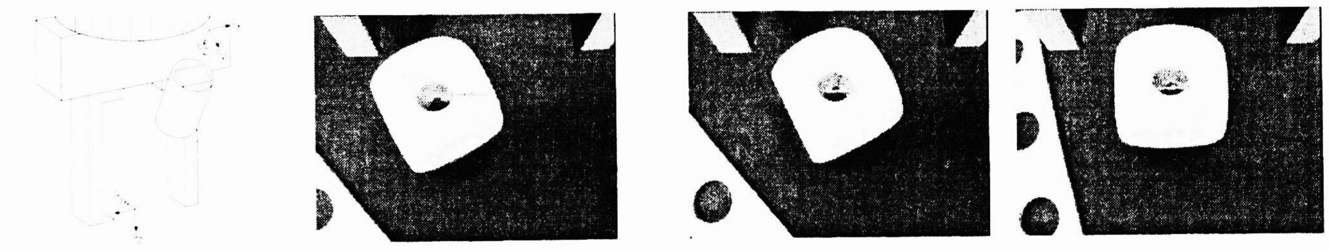
Figure 4. The offline training and online application phase.

4. GRASPING EXPERIMENTS

In the following implementations, linguistic terms of the eigenvectors are defined by B-spline basis function of order three. The output value of each rule is represented as a singleton which is called control vertex in the B-spline fuzzy controller. The control vertices are adaptively determined by minimising the RMS error for the supervised learning.

4.1. Experimental Setup

This approach is applied to find exact grasp positions by a robot parallel gripper equipped with a hand-eye system, Fig. 5.



(a) "Self-viewing" hand-camera. (b) The start position. (c) Improved position. (d) The optimal position for grasping.

Figure 5. Fine-positioning using a "self-viewing" hand-camera.

4.2. Training

An offline learning approach starts with positioning the robot gripper over the desired position. The robot hand is then moved incrementally, e.g. in the freedom of displacement x, y and/or the orientation θ. At each new location, an image is produced while the motion parameters are recorded, i.e. a sequence of images is obtained. To a large degree each pair of two adjacent images is similar, see Fig. 6.

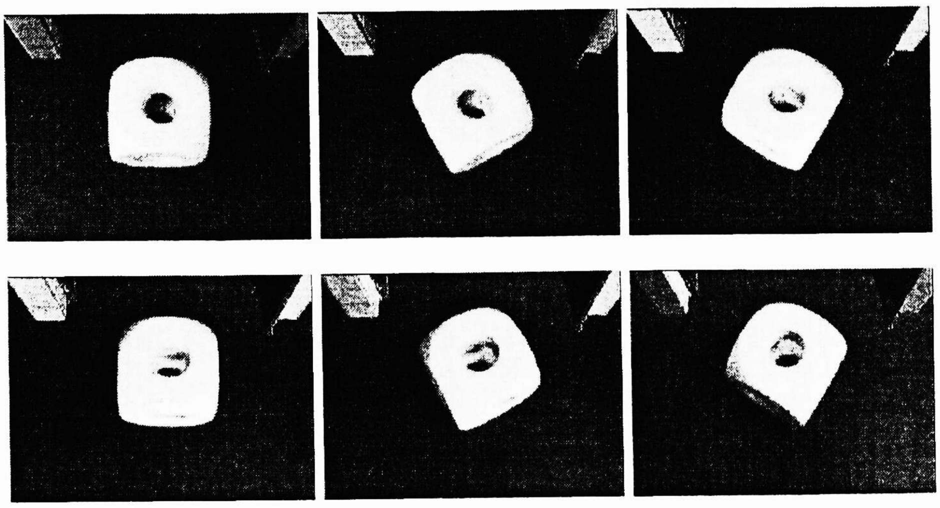


Figure 6. Six training images (192×144 pixels, with orientation variation $0°$, $30°$, $60°$, $90°$, $120°$, $150°$) for automatic grasping, taken from about 300 training images.

The large amount of brightness data of the image pixels can be significantly compressed by finding the eigenvectors of their covariance matrix. If the eigenvectors are ordered according to the magnitude of their eigenvalues, it can be easily found that only a limited number of eigenvectors need to be considered while the others can be ignored.

4.3. Control Rules

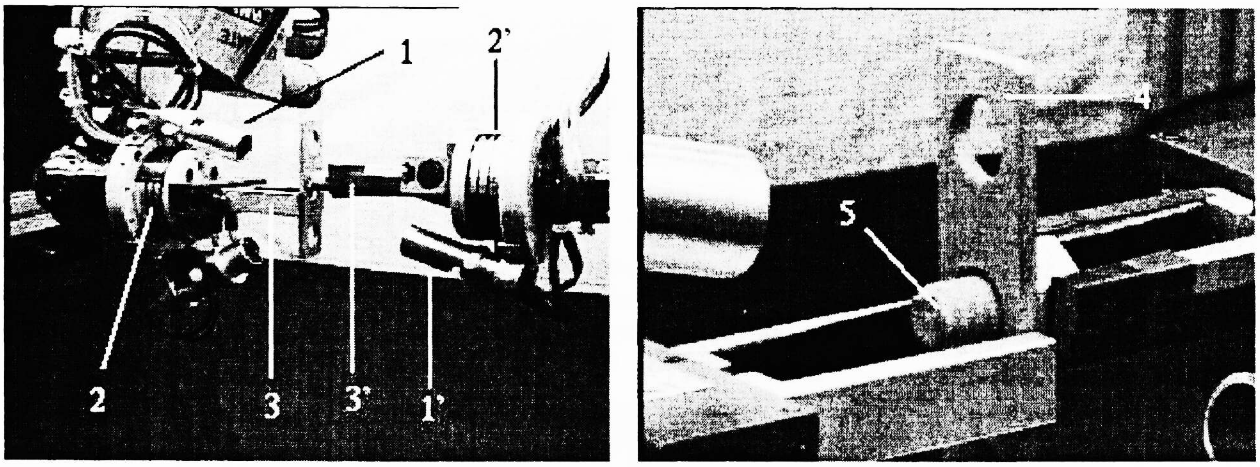
In this way, the dimension of the local perceptual space is reduced to a manipulable size of a subspace. If the axis of a principal component, denoted as p_j ($j = 1, \ldots, n$), is covered with B-spline basis functions, denoted as $X^j_{i_j,k_j}$, the rule for determining the relative location of the robot gripper to the object can be written in the form:

IF (p_1 IS $X^1_{i_1,k_1}$) and ... and (p_n IS $X^n_{i_n,k_n}$)
(*meaning:* *the input image matches pattern*$_{i_1 i_2 \ldots i_n}$)
THEN (x is $X_{i_1 i_2 \ldots i_n}$) and (y is $Y_{i_1 i_2 \ldots i_n}$)
and (θ is $\Theta_{i_1 i_2 \ldots i_n}$)

Each rule corresponds to a supporting control vertex for the interpolation in the eigenspace. with less than 6 eigenvectors a sufficient grasping precision of the separated objects using the robot hand can be obtained.

5. VISUALLY GUIDED HOLE-SEARCHING

In the assembly scenario (Fig. 7), two cooperating robots are to insert a screw (5) into a wooden slat (4). The manipulators are installed overhead and can grasp the required assembly components from the assembly table. Each robot is equipped with a force sensor (2,2') on which a pneumatic parallel-jaw gripper (3,3') is mounted. A small camera (1,1'), which observes the scene, is mounted over the gripper at an angle of approximately 30°. The manipulators are two Puma 260.

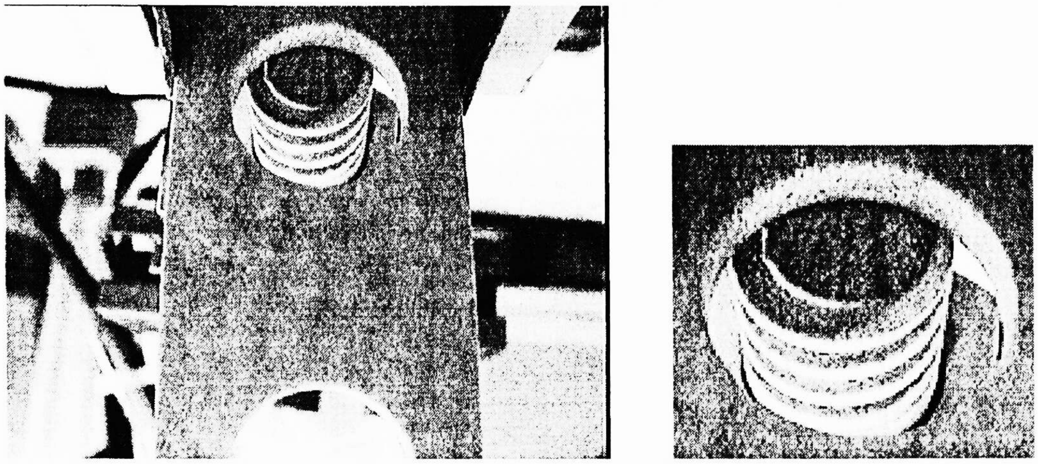


(a) (b)

Figure 7. The experimental setup for assembly. 1,1': hand-camera; 2,2': force/torque sensor; 3,3': parallel jaw-gripper; 4: slat; 5: screw-head.

5.1. Selecting Region of Interest

For this task, focusing attention is transformed into crop the camera image to find out the interesting part, namely the hole in the slat, see Fig. 8.



(a) The camera image. (b) The cropped image.

Figure 8. One important pre-processing is to select the region of interest.

General solution of this problem requires complex image processing algorithms. Fortunately, one trick can be used to robustly find the desired result. Before the slat is grasped, it lies on the desk-top and therefore what the camera sees through the hole, in the moment the slat is grasped, is the desk-top itself, which in our case is black. The position of the hole now can be remembered the whole time the gripper grasps the slat. As long as the slat stays between the gripper fingers, the position of the hole in the images is constant and so pictures can easily be cropped to a rectangle region around the hole.

5.2. The Problem of Finding a Hole

The search for a hole based only on "blindly" force control is slow and prone to failure. It is safer to monitor the scene through the hole with a camera. The screw (or the slat) may thus be guided to the correct position.

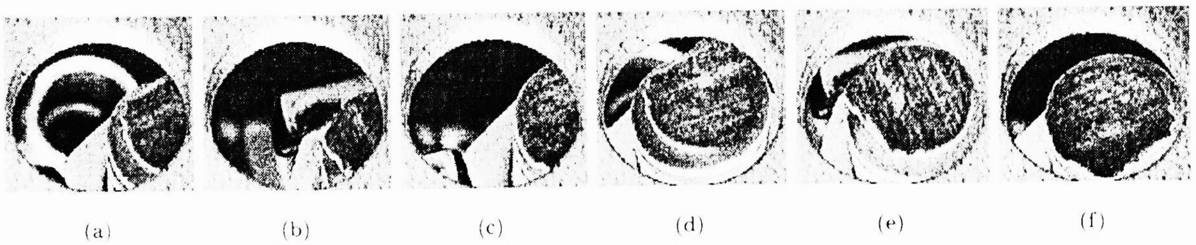


Figure 9. Typical images taken by the hand-camera (Image size: 111 × 103 pixels). Notice that the background changes continuously and the shape of the screw varies during the gripper rotation.

Fig. 9 shows a sequence of typical views of the scene. It is obviously difficult to recognise geometric features in the images.

5.3. Eigenspace Projection

We examined our system with both 200 and 400 training images. The resulting Eigenvectors and the projection on the first three most important eigenvectors are shown in Fig. 10.

5.4. Numerical Results

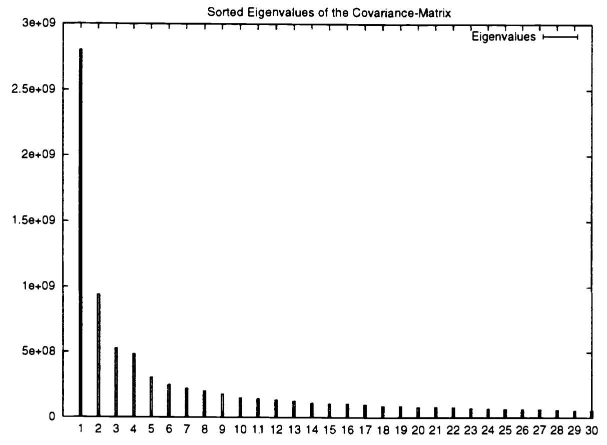
Fig. 11(a) shows the learned control vertices for the first two eigenvectors and 11(b) the resulting control surface.

Table 1 shows the average number of the required correction steps in O-direction depending on the controller parameters for some typical displacements.

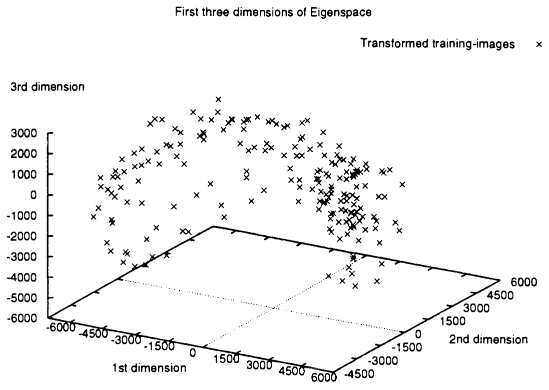
Number of used EV's	3	5	7
Number of used B-Splines	$11 \times 7 \times 7$	$11 \times 7 \times 7 \times 5 \times 5$	$11 \times 5 \times 7 \times 7 \times 5 \times 5 \times 3 \times 3$
Robot displacement			
above	5.0	1.4	1.09
above right	6.86	2.29	1.18
below	×	0.2	0
to the left	×	×	3.64
above left	4.86	2.2	2.7

Table 1. Required correction steps in O-direction for two different controllers.

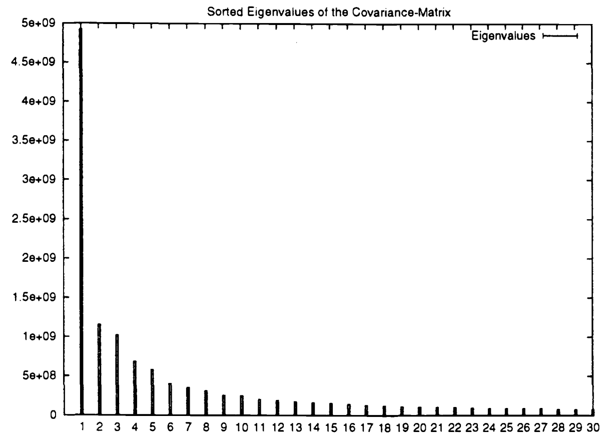
Our experiment shows that with an increasing number of eigenvectors fewer steps for correcting the position of the slat are required. With three and five eigenvectors not all situations can be separated. If, for example, the screw



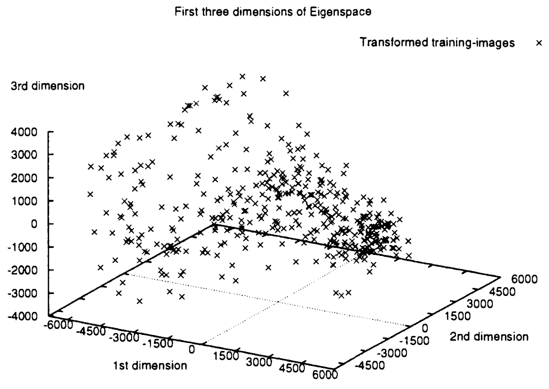
(a) Eigenvalues (200 training images).



(b) Projected data (200 training images).

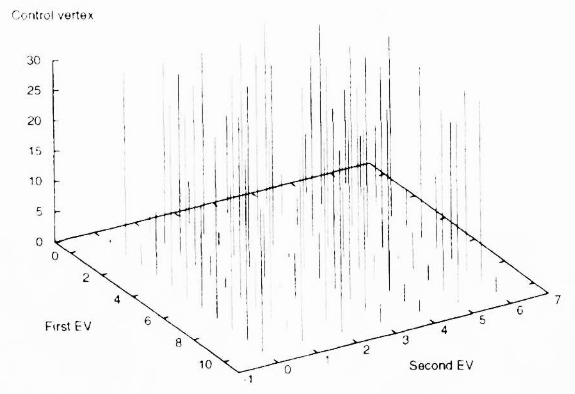


(c) Eigenvalues (400 training images).

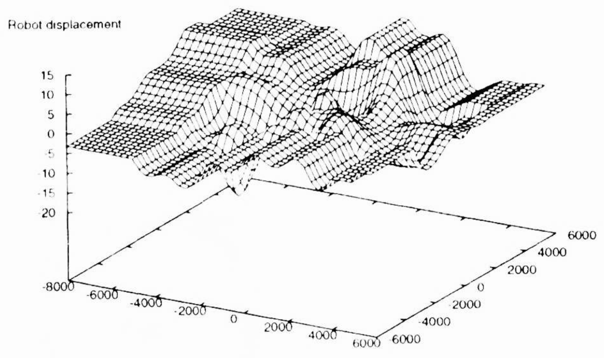


(d) Projected data (400 training images).

Figure 10. Eigenvectors, eigenvalues and eigenspace projections.



(a) Control vertices on the plane of the first two principal components.



(b) Control surface with the first two principal components as input and the robot displacement as output.

Figure 11. Control vertices and surface.

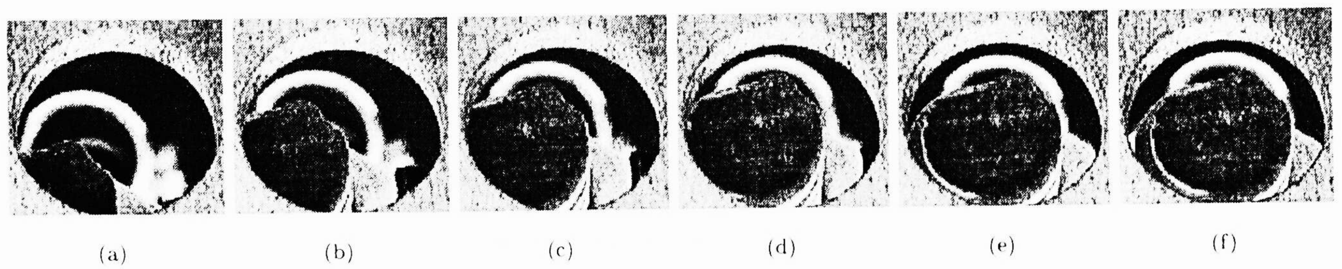


Figure 12. Correction sequence output by the proposed task-mapping model.

is to the left of the hole. the controller cannot correct the displacement. In this case the motion was lead to the wrong direction. Fig. 12 shows a sequence of movements generated by the fuzzy controller.

Force control based on the force/torque sensor readings is applied to establish contact between the screw and the slat before the fine-positioning. and to verify if the screw sits correctly in the hole after the fine-positioning.

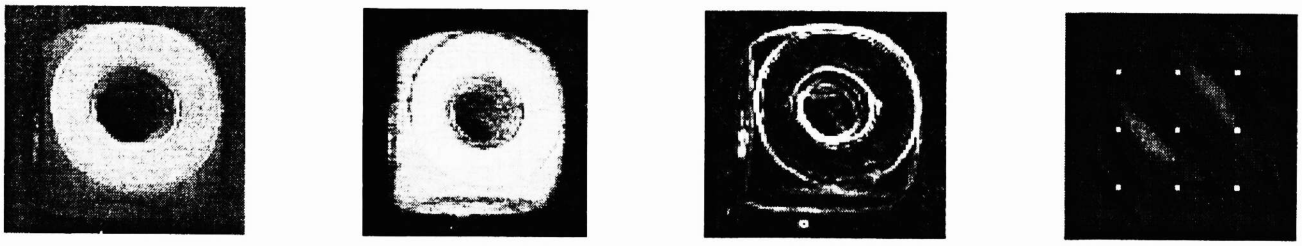
6. DISCUSSION

We have shown that high-dimensional problems such as visually guided fine-motion can be solved with a neuro-fuzzy model. The B-spline model serves as an efficient interpolator which can be interpreted as fuzzy control rules. The advantages of this approach are:

- By projecting the high-dimensional input space into a reduced eigenspace, the most significant information for control is maintained. A limited number of transformed inputs can be partitioned with the B-spline model and a sufficient precision can be obtained for determining the robot position correction.
- The statistical indices used in the approach provide a suitable solution to describe the information in images with a lot of uncertainties.
- A vector in the eigenspace is directly mapped onto the controller output based on the B-spline model. This makes real-time computation possible.

- Training motion can be programmed so that representative images can be generated automatically. *The only work for the user is to place the object or robot so that the grasping or inserting pose is correct.* Obviously, this procedure is much simpler for a user in comparison with the hand-eye calibration.

If observed scenarios are not "local" enough, i.e. the images possess less similarity, then it could be happen that the grasp precision cannot be satisfied with too few (e.g. with six) eigenvectors. For these cases, we are investigating methods to classify the image sequence into more local scenarios by using some simple criteria, e.g. objects are "left", "right", "top" and "down".



(a) I-component. (b) S-component. (c) After a Sobel filter. (d) After a Gabor filter.

Figure 13. Different modalities of a colour image input can be integrated for PCA.

If more sensors or different representations from the image pre-processing are available, they can also be added to the input vector. Therefore, without modification, the proposed model can integrate multiple sensors and multiple modalities, see Fig. 13 for an example. Since each filter works robustly again certain noise factors, integration of many results after different filters can enhance the robustness of the approach. The proposed solution can still work because the efficiency of the PCA still depends on the variance of the input data although the dimension of the input information increases. Our future work will be investigating the capability of the approach to grasp more complicated assembled aggregates by using multiple cameras and laser sensors.

Acknowledgement

This work is partly funded by DFG project SFB360 "Situated Artificial Communicators". The authors thank Y. von Collani, R. Schmidt, A. Zielke, H. Wirausky for implementing the methods with real robots.

REFERENCES

1. J. S. Albus. A new approach to manipulator control: The Cerebellar Model Articulation Contorller (CMAC). *Transactions of ASME, Journal of Dynamic Systems Measurement and Control*, 97:220–227, 1975.
2. J. Greenstadt. The determination of the chararcteristic roots of a matrix by the Jacobi method. *IBM Report*, 1961.
3. I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2470–2476, 1996.
4. W. T. Miller. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on System, Man and Cybernetics*, 19:825–831, 1989.
5. S. K. Nayar, H. Murase, and S. A. Nene. Learning, positioning, and tracking visual appearance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3237–3244, 1994.
6. E. Oja. *Subspace methods of pattern recognition*. Research Studies Press, Hertfordshire, 1983.
7. T. Sanger. *An optimality principle for unsupervised learning*. Advances in neural information processing systems 1. D. S. Touretzky (ed.), Morgan Kaufmann, San Mateo, CA, 1989.
8. G.-Q. Wei, G. Hirzinger, and B. Brunner. Sensorimotion coordination and sensor fusion by neural networks. In *Proc. IEEE Int. Conf. Neural Networks, San Francisco*, pages 150–155, 1993.
9. J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems*, 13(2/3):257–285, FEB./MAR. 1998.