**To appear in** *Robotics and Autonomous Systems*, **Elsevier, 1999**

# A Neuro-Fuzzy Control Model for Fine-Positioning of Manipulators

Jianwei Zhang, Alois Knoll, R. Schmidt

*Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany*
*Tel: ++49–521–106–2951/2952*
*Fax: ++49–521–106–6440*
*E-mail: zhang | knoll | rschmidt @techfak.uni-bielefeld.de*

**Abstract**

We propose an approach for learning the fine-positioning of a parallel-jaw gripper on a robot arm using visual sensor data as controller input. The first component of the model can be viewed as a perceptron network that projects high-dimensional input data into a low-dimensional eigenspace. The dimension reduction is efficient if the movements achieving optimal positioning are constrained to a local scenario. The second component is an adaptive fuzzy controller serving as an interpolator whose input space is the eigenspace and whose outputs are the motion parameters. Instead of undergoing cumbersome hand-eye calibration processes, our system is trained in a self-supervised learning procedure using systematic perturbation motion around the optimal position. The approach is applied to tasks of three degrees of freedom, e.g. translating the gripper in the $x$-$y$-plane and rotating it about the $z$-axis.

## 1 Introduction

Fine-positioning is one of the most important and most demanding sensor-based manipulation skills. Even relatively simple tasks such as grasping rigid objects with two-fingered grippers based on an image taken by a hand-camera presuppose an effective sensorimotor feedback. This entails the implementation of the whole perception-action cycle including image acquisition with the hand-camera, image processing, generation of manipulator actions for approaching the grasping position, etc. Additional levels of complexity are added if the system is to be designed so as to work under variable lighting conditions, moving or occluded objects. It is also desirable that the system be able to control the manipulator from *any* location in the vicinity of the object to the optimal grasping position regardless of perspective distortions (if the object

is seen from "remote" points), specular reflections and the like. 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, completely known (including camera parameters) and if the geometric features for detecting the grasping position can be extracted robustly from the camera image. We note, however, that even if these conditions are met, the hand-eye calibration matrix cannot be interpreted as an adequate cognitive model of human grasping (and hence probably never become just as powerful). Our idea to solve the fine-positioning control problem is to use a direct, linear method to reduce the input dimension and then apply the non-linear B-spline model to map the projection on the subspace further to the control output. Since the a robot arm can measure all its movements quite precisely using internal joint position sensors, we can easily program the robot to generate training data by itself. The whole controller is actually constructed by a neural network of five layers. The B-spline model can be interpreted as fuzzy "IF-THEN" rule system whose input is the whole image scenario data and output is the correction motion of the gripper.

Recently, neural network-based learning has also found applications in grasping: [5,9,3,6] use geometric features as input to the position controller. Since the image processing procedures such as segmentation, feature extraction and classification are generally not robust in real environments and since these processing algorithms are computationally expensive, some of the work resorts to marking points on the objects to be grasped. By contrast, for dealing with the general case of handling objects whose geometry and features are not precisely modelled or specially marked, it is desirable that a general control model can be found which, after an initial learning step, robustly transforms raw image data *directly into action values*. In [7] Murase and Nayar used PCA (principal component analysis) for object classification and for solving a one-dimensional position-reconstruction-problem. In [2] Black and Jepson presented an approach they called *eigentracking,* which can be used to track objects in picture sequences. However, the experimental results did not prove that PCA-based tracking is capable of controlling a robot.

## 2 Experimental Environment

Our robot system [4] aims at assembling a toy-aircraft from basic objects like screws, ledges or nuts (Fig. 1). Within this scope different tasks have to be performed: determine which basic objects are needed, identify a single object, position the gripper above it, grasp it, assemble it with others. The task discussed here is the fine-positioning of a manipulator after a coarse positioning has been completed. The object to be grasped is visible in the image of a "self-viewing" eye-in-hand camera (Fig. 2), which sees an area of

Fig. 1. The working cell of our robot system. The "Baufix"-parts on the assembly table are to be grasped by the robot gripper.

about 11 cm × 9 cm of the $x$-$y$-plane (not exactly the PAL image ratio). The aim is to move the robot hand from its current position (Fig. 3 left) to a new position so that the hand-camera image matches the optimal grasping position (Fig. 3 right). In our setting there are 23 different objects to be handled. Some of the objekts in the image have the same shape but different colors. It is therefore mandatory that a general image processing technique be applied, which needs *no specialised algorithm for each object* and shows stable behaviour under varying object brightness and color.

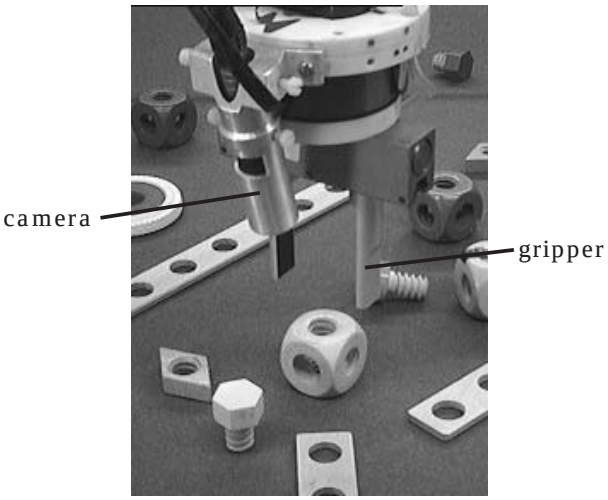



Fig. 2. The end-effector of the manipulator with a hand-camera (positioned optimally over the yellow cube).

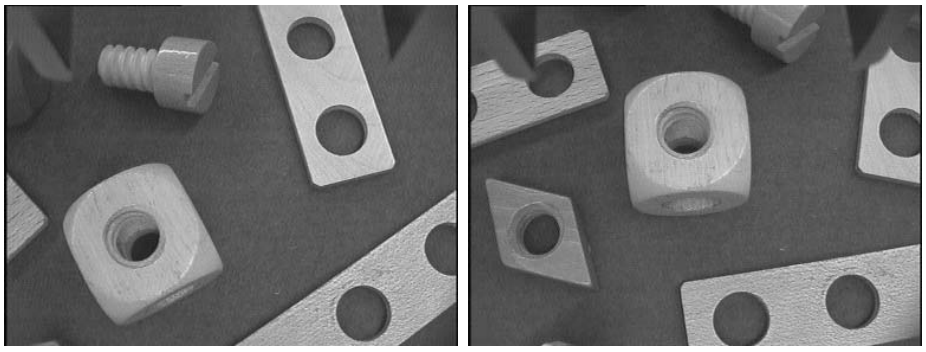


Fig. 3. A cube viewed from the hand-camera – before and after fine-positioning.

## 3 The Perception-Action Transformation

### *3.1 The Neuro-Fuzzy Model*

Fig. 4 depicts the feed-forward neuro-fuzzy structure which fulfils three subtasks subsequently: *pattern coding*, *pattern matching* and *rule firing and output synthesis*.

The pattern coding aims at reducing input dimension based on the eigenspace projection. Depending on how "local" the measuring data are and, therefore, how similar the observed sensor patterns appear, a more or less small number of eigenvectors can provide a sufficient summary of the state of all input variables (see the left part of Fig. 4). Our experimental results show under the most diverse conditions that it is very likely that three or four eigenvectors provide all information indices of the original input space necessary for the positioning task. Therefore, in the case of very high input dimensions, an effective dimension reduction can be achieved by projecting the original input space into the eigenspace.

The pattern matching, rule firing and output synthesis are realised with an adaptive B-spline fuzzy controller. Eigenvectors can be partitioned by covering them with linguistic terms (the right part of Fig. 4). In the following implementations, fuzzy controllers constructed according to the B-spline model are used [10]. 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 fewer parameters than other set functions such as trapezoid, Gaussian function, etc. The output computation is straightforward and the interpolation process is transparent. Through numerous applications in robot control and data modelling, the B-spline controllers have shown good approximation capabilities and rapid convergence.

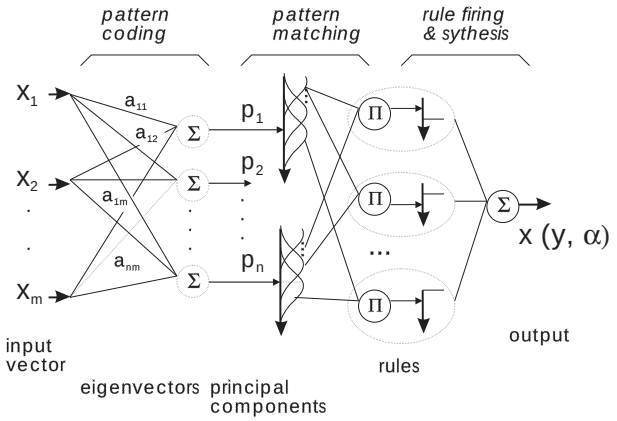



Fig. 4. The task based mapping can be interpreted as a neuro-fuzzy model. The input vector consists of pixels of a grey-scale image.

### *3.2 Dimension Reduction via PCA*

Let us assume $k$ sample input vectors $\vec{x}^1, \ldots, \vec{x}^k$ with $\vec{x}^i = (x_1^i, \ldots, x_m^i)$ originating from a pattern-generating process, e.g. the stacked input image vectors. The PCA can be applied to them as follows:

First the (approximate) mean value $\vec{\mu}$ and the covariance matrix $\mathbf{Q}$ of these vectors are computed according to

$$\mathbf{Q} = \frac{1}{k}\sum_{i=1}^{k}(\vec{x}^i - \vec{\mu})(\vec{x}^i - \vec{\mu})^T, with \quad \vec{\mu} = \frac{1}{k}\sum_{i=1}^{k}\vec{x}^i$$

The eigenvectors and eigenvalues can then be computed by solving

$$\lambda_i \vec{a}_i = \mathbf{Q}\vec{a}_i$$

where $\lambda_i$ are the $m$ eigenvalues and $\vec{a}_i$ are the $m$-dimensional eigenvectors of $\mathbf{Q}$. Since $\mathbf{Q}$ is positive definite all eigenvalues are also positive. Extracting the most significant structural information from the set of input vectors $\vec{x}^i$ is equal to isolating those first $n$ $(n < m)$ eigenvectors $\vec{a}_i$ with the largest corresponding eigenvalues $\lambda_i$. If we now define a transformation matrix

$$\mathbf{A} = (\vec{a}_1 \ldots \vec{a}_n)^T$$

we can reduce the dimension of the $\vec{x}^i$ by

$$\vec{p}^{\,i} = \mathbf{A} \cdot \vec{x}^i; \quad dim(\vec{p}^{\,i}) = n$$

The dimension $n$ should be determined depending on the discrimination ac-

curacy needed for further processing steps vs. the computational complexity that can be afforded.

## 4 Implementation

The working system implements two phases: off-line training and on-line evaluation (Fig. 5). In the off-line phase, a sequence between 10 and 100 training images showing the same object in different positions is taken automatically, i.e. without human intervention. For each image the position of the manipulator in the plane and its rotation about the $z$-axis, both with respect to the optimal grasp position for the current object, is recorded.

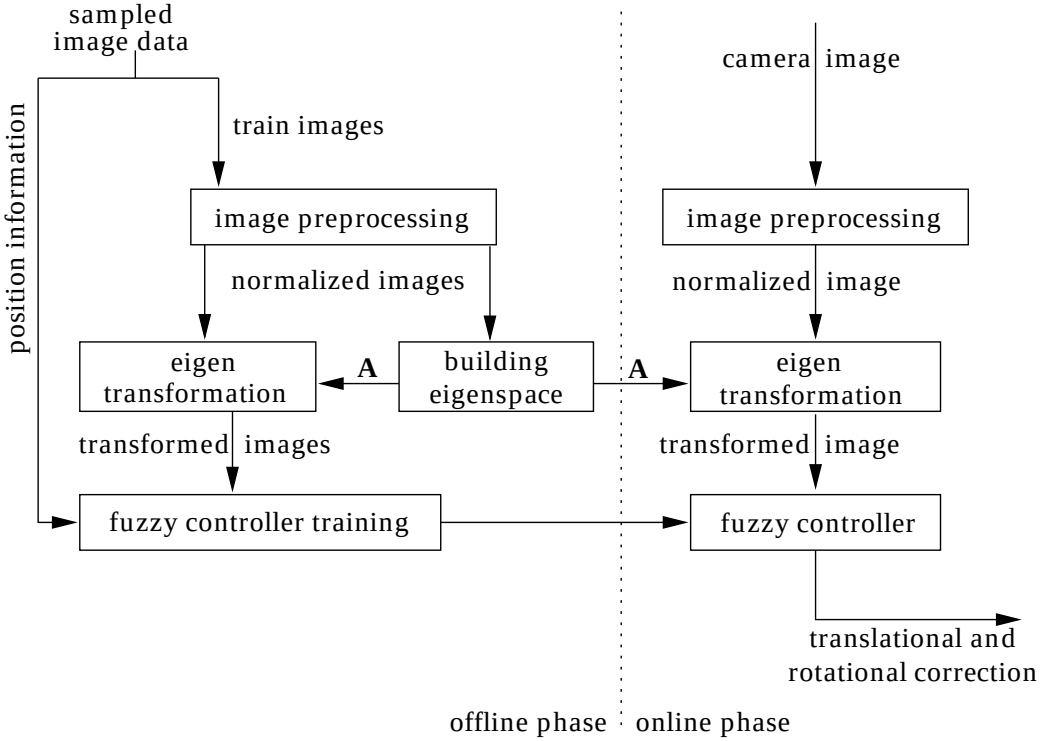



Fig. 5. The training and the application of the neuro-fuzzy controller.

Fig. 6 shows a typical pattern of positions for taking training images. The reduced eigenspace of the images is computed by PCA and the training data are transformed into this space. With these data B-spline fuzzy controllers are built that take the principal components as input variables and whose outputs correspond to the $x$-$y$-position and the angle $\alpha$ of the manipulator gripper. The training of the controller output is realised with the back-propagation. Since the local control feature of the B-splines, the training process can converge to the single minimum rapidly.

In the on-line phase, the camera output is projected onto the eigenspace and

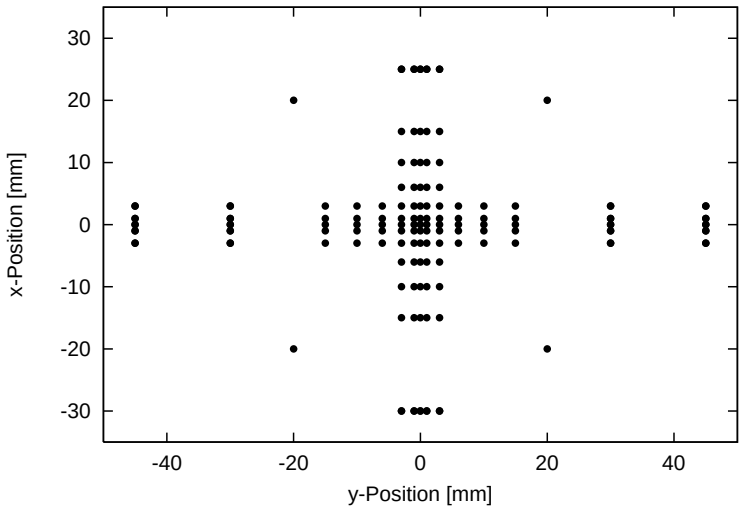



Fig. 6. The positions where the images for the $x$- and $y$-controllers are taken.

is then processed by the fuzzy controller. The controller output is the end-effector's position and angle correction.

### *4.1 Preprocessing*

The task of preprocessing is to select the focus object from an image with complex background and to transform the image to the standard using during the training. Fig. 7 (left) shows a typical image taken by the camera. Fig. 7 (right) shows the image after clipping by simple thresholding. This operation produces a single *Region of Interest*. After clipping all images are normalised with respect to their "energy" [7]:

$$x_j^i = \frac{\tilde{x}_j^i}{\sqrt{\sum_{l=1}^{dim} \left(\tilde{x}_l^i\right)^2}}$$

where $\tilde{x}_j^i$ is the intensity of the $j$-th pixel in the $i$-th image, $x_j^i$ is the intensity of the $j$-th pixel in the corresponding normalised image and $dim$ is the number of pixels in the image.

For detecting the rotation of an object, one more preprocessing step is necessary: since most of the variance in the images is caused by translations, the rotation cannot be learned properly from the eigen-transformed images (Fig. 8 and 9). To eliminate the variance caused by changes in the position, the region of interest should be moved to the centre of the image. As this removes the translational information from the images, two eigenspaces must be computed:

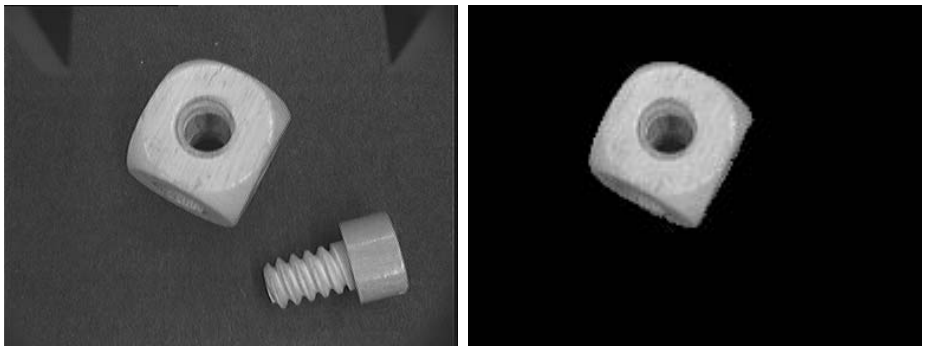


Fig. 7. A typical camera image, before and after clipping.

one based on the original images and one based on the shifted versions.

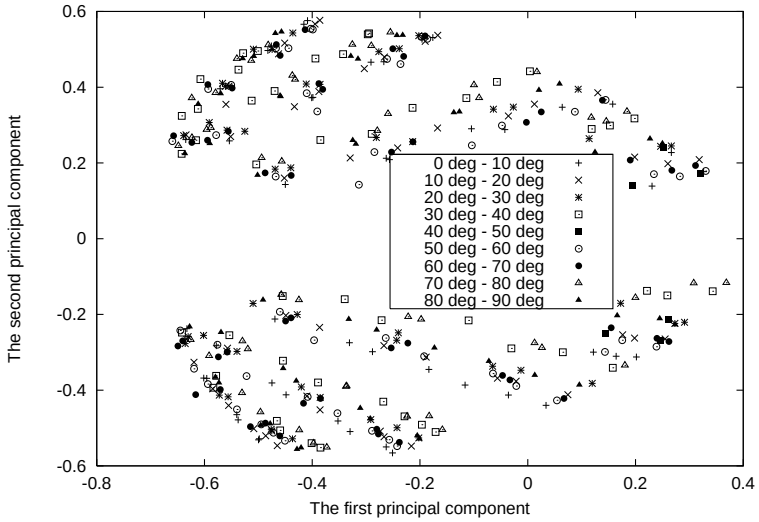



Fig. 8. Eigenspace vectors resulting from the training images with no position shifting. Only the first two components of these vectors are drawn in this projection. They are classified by the angle of the cube in each image to reveal their unordered placement, which makes it impossible for adaptive techniques to learn sensible structures (compare with Fig. 9).

### *4.2 Implementation of PCA*

The PCA is implemented by interpreting each of the $k$ training images as a vector $\vec{x}^i$ in which the pixel rows are stacked, i.e. stored consecutively. The covariance matrix $\mathbf{Q}$, however, is not computed explicitly because this would be completely intractable: let the image size be $192 \times 144$. Then, the number of pixels ($dim$) in this image is $192 \cdot 144 = 27648$ resulting in a size of the covariance matrix of $27648 \times 27648$, i.e. it consists of $(27648)^2 = 7.64 \cdot 10^9$ elements.

In [7] a procedure is described for computing the first $k$ most important eigen-

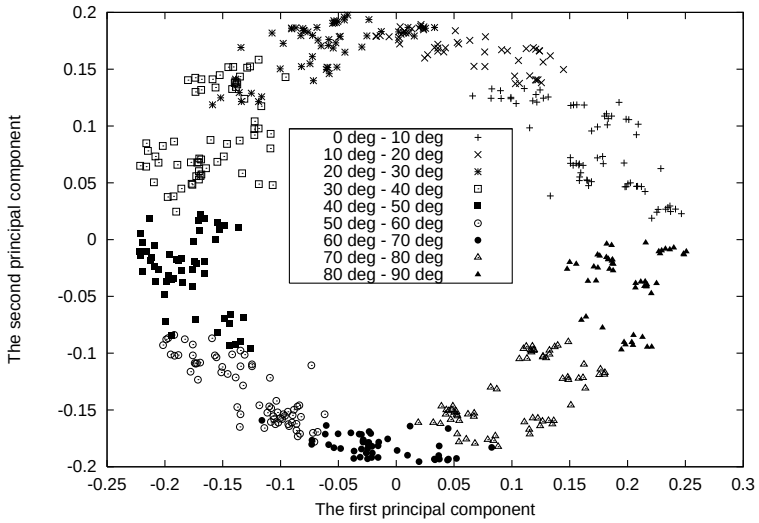



Fig. 9. Eigenspace like in Fig. 8. Here, the region of interest was shifted to the image centre. The obvious clustering is the basis for the adaptive learning scheme.

vectors and eigenvalues of this covariance matrix without computing the matrix itself. Of these $k$ eigenvectors we use only a subset of $n$ vectors, corresponding to the $n$ largest eigenvalues. When combining the PCA with standard pattern-matching techniques such as nearest-neighbour classification, $n$ is usually between 10 and 20. By contrast, for controlling a manipulator with our B-spline fuzzy controller, 3 or 4 input dimensions are sufficient.

Let $\{\vec{a}_i | i \in 1 \ldots n\}$ be the $n$ most important eigenvectors. Then, after the eigen-transformation, we have the following preliminary results:

- A matrix $\mathbf{A} = (\vec{a}_1 \cdots \vec{a}_n)^T$, which transforms images into the $n$-dimensional subspace of the eigenspace.
- $\vec{p}^{\,i} = \mathbf{A} \cdot \vec{x}^i$, the eigen-transformed and projected training image vectors.
- The position and angle of the object in each training image and hence the position and angle $\vec{d}^i = (x, y, \alpha)$ that corresponds to each vector $\vec{x}^i$.

### *4.3 Fuzzy Controller Training*

With the $\vec{x}^i$ and the corresponding $\vec{d}^i$ a B-Spline fuzzy controller is trained. We use third-order splines as membership functions and between 3 and 5 knot points for each linguistic variable. The distribution of these points is equidistant in the current implementation and constant throughout the whole learning process. The coefficients of the B-Splines (de Boor points) are initially zero. They are modified by the rapid gradient descent method during training [10].

In the on-line phase the same image preprocessing as in the off-line phase is applied. Then, the image vector is transformed into the eigenspace. The resulting $n$-dimensional vector is fed into the fuzzy controller, which, in turn, produces the position and angle of the object in the image. These values are then used to move the robot closer to the target object. This sequence is repeated several times; normally in our experiment no more than 3 steps are necessary until all parameters (i.e. deviation in $x$ and $y$ direction and residual angular deviation) are below a specific threshold (e. g. 0.5 mm and 1 degree).

To improve the raw algorithm outlined above several aspects were refined:

**Color Images:** Instead of the gray-scale images, the saturation parts of color images in the *Hue-Saturation-Intensity* color-space may be used. For objects with full colors ("rainbow" colors) the saturation part is high; for colors like teal, pink or light blue this component is low and for all grey-tints including black and white it is zero. This increases the contrast between objects and background when compared with the intensity image (Fig. 10). Thus, in the case of colored objects, the controller becomes highly independent of the hue of the objects.

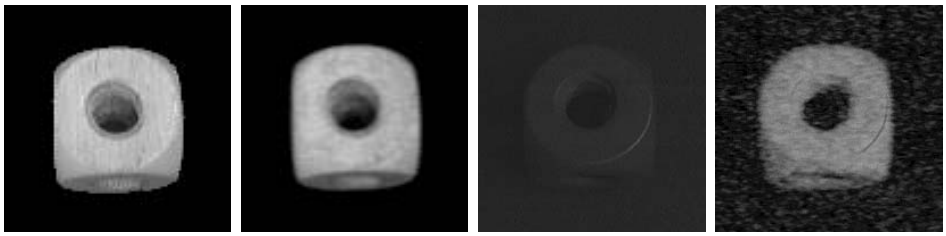


Fig. 10. The intensity image and the color-saturation image of the yellow cube (left two images) and the blue cube (right two images).

**Boosting image vectors:** The proposed PCA based dimension reduction method is not limited to one image per vector. For example, the vector $\vec{x}$ could consist of the intensity image, the saturation image, and a Sobel-filtered (Fig. 11) intensity image. This can help to suppress inaccuracies due to unusual lighting conditions. Obviously, further (possibly object-dependent) improvements can be achieved with specialised feature detectors (lines, angles, etc.).

**Hierarchy:** If, for an object of complex shape, the discrimination accuracy of the neuro-fuzzy controller is not sufficient, a hierarchical system may be built. The camera images are separated into regions, then an appropriate classifier detects in which region in the image the object to be grasped is located and, based on this information, the robot moves approximately to the optimal grasping position. After this movement, a neuro-fuzzy controller is

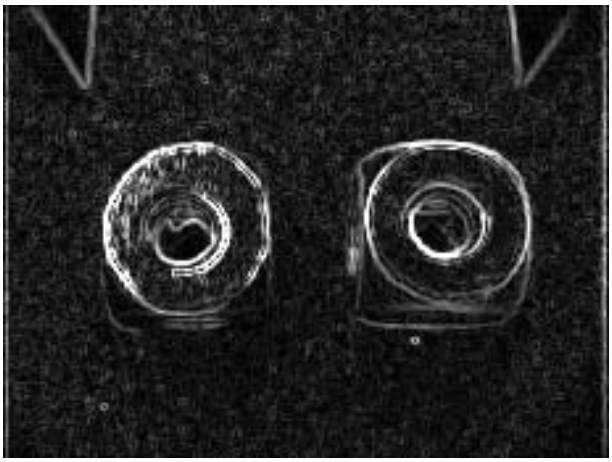


Fig. 11. A sobel-filtered image of two cubes.

trained. The training images for it need only show the object near the optimal position. Such a system is even more accurate than the neuro-fuzzy controller alone.

### *4.5 Optimal Choice of Training Images*

The appearance-based method is frequently criticised for the fact that the *training images* must be chosen manually, which often leads to simple trial-and-error. To cope with this problem we developed a method for automatically determining the positions where the camera images should be taken. Since the robot is allowed to do several steps, high accuracy is only needed near the optimal grasp position $\vec{d} = (x_0,\, y_0,\, \alpha_0) = (0,\, 0,\, 0)$.

**Rotation:** The angles at which the images are taken depend on the object symmetry $S$. For objects with an $S$ of less than 360 degrees there is more than one optimal grasp position. That is because it makes no difference whether a cube is grasped by the front and rear side or at the left and right side. So near the angles $0, S, 2S, \ldots$ more images are needed. The test objects in Fig. 12 possess the following symmetries: For the ledge $S$ is 180 degrees, for the cube $S$ is 90 degrees and for the screw head $S$ is 60 degrees.

To limit the number of images for objects with a small $S$, the following changes are made: If $S$ is smaller than 90 degrees, then it is multiplied by the smallest integer that produces a value of greater than or equal to 90 degrees. This leads, for example, to an $S$ of 120 degrees for the screw head. The following heuristic formula produced acceptable results:

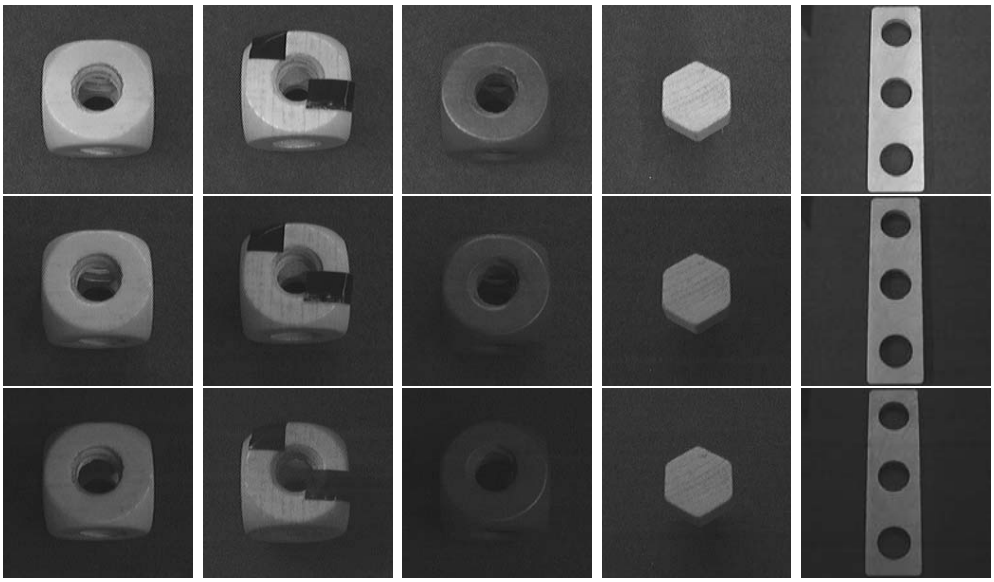


Fig. 12. Test objects for grasping, from left to right: yellow cube, partly covered yellow cube, blue cube, yellow screw head, ledge with 3 holes; from top to bottom: optimal, worse and poor illumination.

$$\mathbf{W} = \bigcup_{\substack{i \in \mathbf{N_0} \\ j = 0,\, 1, \ldots, 360/S - 1}} \left\{ \left\lfloor \frac{S}{2} \frac{1}{2^i} \right\rfloor + j \cdot S,\ S - \left\lfloor \frac{S}{2} \frac{1}{2^i} \right\rfloor + j \cdot S \right\}$$

For the cube, this formula gives the set of angles $\mathbf{W} = \{45,\ 23,\ 67,\ 11,\ 79,\ 6,\ 84,\ 3,\ 87,\ 1,\ 89,\ 0,\ 90,\ 45{+}90,\ 23{+}90,\ \ldots\ \}$(in degrees), with $\mathbf{W}$ containing 48 elements. Due to the clipping described in section 4.1 for rotation, only training images near the optimal grasping position are taken, at the points with coordinates $(0,1)$, $(1,0)$, $(0,0)$, $(0,1)$, and $(1,0)$. Long objects like the ledge can lie partly outside the image. In this case, images with 0, 90, 180 and 270 degrees are added at the 4 positions $(\pm 25\,\mathrm{mm}, \pm 25\,\mathrm{mm})$.

**Translation:** Images at the positions shown in Fig. 6 are taken with 0 degrees rotation. In most cases the resulting accuracy for the $x$- and the $y$-controller is satisfying with these images. If not, either the controller for $x$ or that for $y$ can be selected. If the $y$-placement is not correct, then we rebuild the $y$-controller with images at those positions in Fig. 6 where $x = 0$.

#### *4.5.1 Preparation of Continuous Output for Learning.*

Since the fuzzy controller learns to approximate a function, it works correctly only if the function to be learned is continuous, i.e. a differential change of the input will result in a differential change of the output. The correction angle $\alpha$ of the objects to be grasped has different rotation symmetry (lying screw:

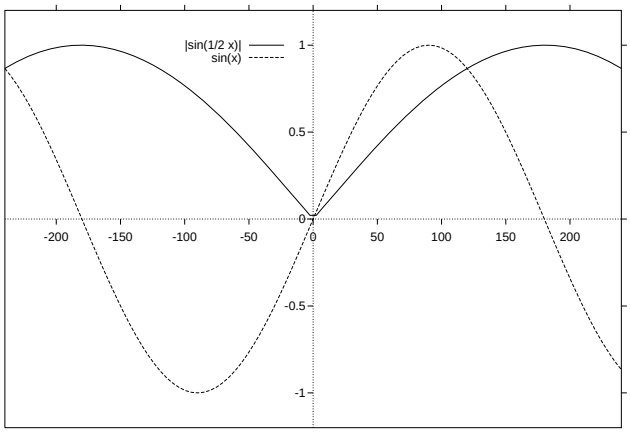



Fig. 13. Two functions used for fuzzy controller learning

360°, slat: 180°, block: 90°, standing screw with six-edge head: 60°). Therefore, we need to find a set of functions which meet the following conditions: a) continuous output values can be generated for fuzzy controller learning; b) the original correction angle can be uniquely reconstructed given the values of these functions. We propose the following two learning functions (Fig. 13):

$$\mathrm{L}_a = \sin\left(\frac{1}{2}\alpha\right), \qquad \mathrm{L}_s = \sin\left(\alpha\right)$$

These two fuzzy controllers are needed to learn $L_a$ and $L_s$ separately. The correction angle can be reconstructed as follows:

- The function arcsine supplies a value between $-90°$ and $+90°$. $|2\arcsin(\mathrm{L}_a)|$ supplies the absolute value of the correction angles.
- The sign of $\arcsin(\mathrm{L}_s)$ provides the information on whether the object is rotated clockwise or counter-clockwise with respect to the gripper.

In the application phase, the gripper motion should be corrected in the reverse direction of the object rotation. Therefore, the correction angle $\alpha$ can be calculated:

$$\alpha = -\mathrm{sign}(\mathrm{L}_s) \cdot |2\arcsin(\mathrm{L}_a)| \tag{1}$$

These two functions can be extended for objects with the symmetry $S$:

$$\mathrm{L}_a = \sin\left(\frac{360^\circ}{S}\frac{1}{2}\alpha\right); \qquad \mathrm{L}_s = \sin\left(\frac{360^\circ}{S}\alpha\right)$$

The reconstruction of the angle is then:

$$\alpha = -\frac{S}{360^\circ}\mathrm{sign}(\mathrm{L}_s) \cdot |2\arcsin(\mathrm{L}_a)|$$

## 5 Experimental Results

### *5.1 Test Objects*

The approach was applied to the grasping of different objects: a yellow cube, a partly covered yellow cube, a blue cube, a yellow screw head, and a ledge with 3 holes (Fig. 12). All training images were taken under optimal lighting conditions. For each object a specific controller was trained, except for the three cubes, where training (not grasping!) was restricted to the yellow cube. For the ledge, different training images for $x$ and $y$ were used (see section 4).

Only the eigenvectors corresponding to the three largest eigenvalues were used as input to the fuzzy controllers. The eigenvalues for rotation/translation and the corresponding eigenvectors, which have the same dimension as the training images and can hence be interpreted as images, are shown in Fig. 14. The eigenspace and the fuzzy controller that were derived from these data were applied to 15 different scenarios: the manipulator was to be positioned above the five objects, each with optimal, worse, and poor illumination (Fig. 12) and from the most remote starting position. The accuracy of the controllers was determined as the average error of 50 positioning sequences for each scenario.

### *5.2 Numerical Results*

Table 1 shows the *RMS error* for $x$, $y$, and the rotation angle $\alpha$ for positioning above the objects. Obviously, the positioning is correct even for the blue cube with the controller trained on the yellow one. It is easy to see that for the translation it makes hardly any difference whether the illumination is optimal or less optimal. The performance deteriorates under poor lighting conditions but it is still good enough to grasp the object. The rotation is more dependent

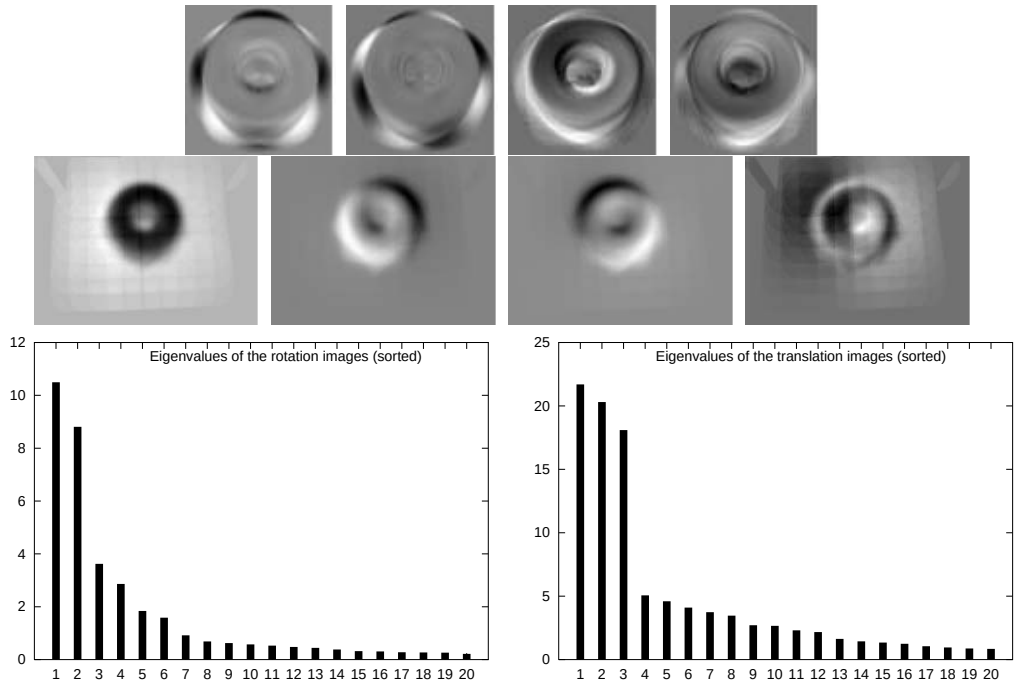



Fig. 14. Eigenvalues and eigenvectors for the cube. In the first line: the first 4 eigenvectors with the fixed placement but varying rotation angles; in the second line: the first 4 eigenvectors used for the translation; in the bottom line: the eigenvalues of the first 20 eigenvectors, sorted according to their values, the left one for rotation and the right one for translation.

on the illumination, in particular with the blue cube. That is because the vertical edges of the cube are practically invisible.

### *5.3 Linguistic Interpretation of the Controller*

One main advantage of the neuro-fuzzy system in comparison with other adaptive systems like the multi-layer perceptron is the interpretability of the controller's function. First we explain how the projections in the eigenspace can be transformed back into the original input image space as follows.

Eigenvectors are orthogonal to each other. If all the $m$ eigenvectors are used, then the resulting complete transformation matrix $\mathbf{A}_c$ is quadrate and orthogonal. The inverse of an orthogonal matrix is equal to its transposed:

$$\mathbf{A}_c^{-1} = \mathbf{A}_c^{\tau}, \tag{2}$$

We can transform the eigenspace projected image back in the original format

as follows:

$$\vec{x}^{\,i} = \mathbf{A}_c^{-1}\vec{p}^{\,i} = \mathbf{A}_c^{\tau}\vec{p}^{\,i}. \tag{3}$$

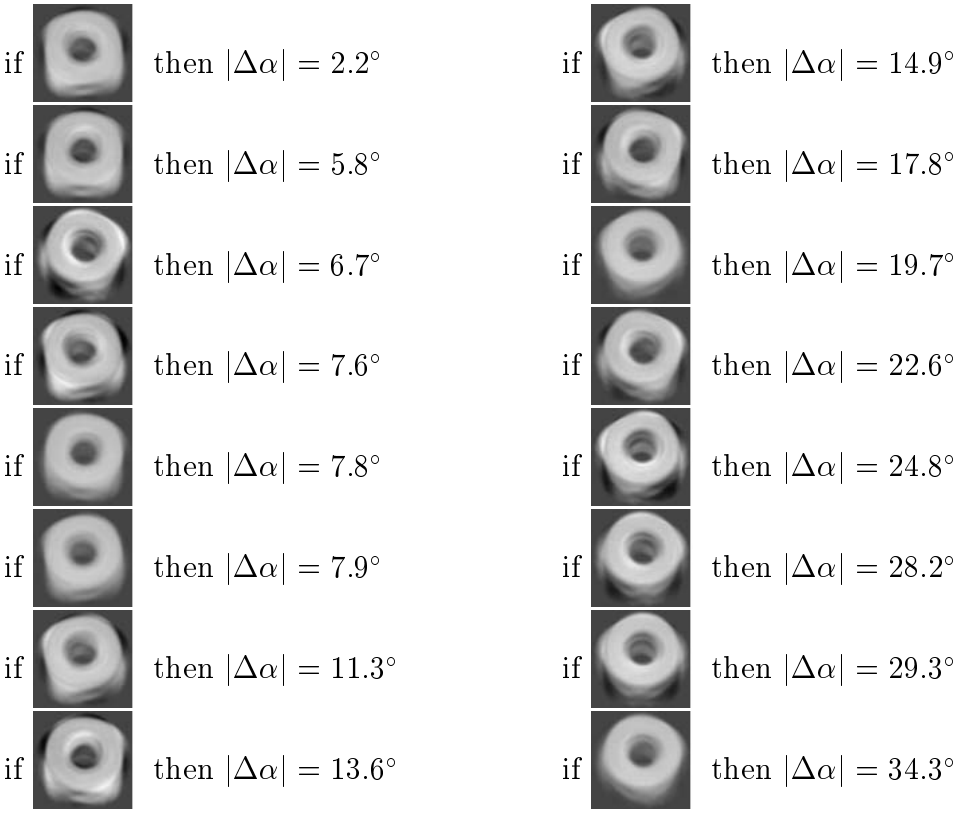
If $\mathbf{A}$ is incomplete, then the projection $\vec{p^i}$ is dimension reduced with respect to $\vec{x}^i$. The back transformation can be approximated by filling the unavailable components of $\mathbf{A}$ and $\vec{p}^{\,i}$ with zeros.

In this way, the control rules can be given an interpretation as follows:

IF Antecedent THEN Consequent

where *Antecedent* is a back-transformed image and the *Consequent* (the controller output) is the $x$-, $y$-value or the correction angle $\alpha$.

The following example illustrates the rules for a two-dimensional controller, each input variable with four linguistic terms. Therefore there are $4 \cdot 4 = 16$ rules altogether. The rotation control looks as follows:



## 6 Conclusions

We have shown that the PCA in conjuction with neuro-fuzzy control is a practical technique for performing multi-variant task-oriented image processing tasks. It is a general method which needs learning but no geometric features. By contrast, this approach has the following advantages over classical approaches:

**Calibration-free.** The camera need not be calibrated.

**Direct mapping.** No computationally expensive algorithms are needed for edge detection, region growing, etc. The projection into the eigenspace and the B-spline interpolation can be performed almost in real-time.

**Model-free.** No model for recognising an object is needed. Thus, it is no longer necessary to implement special algorithms for each object.

**Robust.** The appearance-based approach is robust even when the camera focus is not correctly adjusted or objects are soiled.

In complex scenarios or by considering more degrees of freedom of the robot, a large amount of observation data are not necessarily correlated closely enough to make the PCA efficient. We are currently investigating the dimension reduction capability for vision based control by using self-organising neural networks and output-related features. For complex tasks, we are working with hierarchy to automatically divide a complex image sequence into local "situations". A local controller for one situation should contain a limited number of output-related features and at the same time minimise the interpolation error. To enhance each local controller in diverse complex environments, we are also working with adding further components into the input vector, like redundant camera data, as well as some robust, easily extractable features because the proposed neuro-fuzzy model intrinsically possesses the capability of integrating multiple sensors and multiple representations.

## 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] M.J. Black and Allan D. Jepson. "EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Proceedings of the ECCV'96, Cambridge*, pp. 329–342, 1996.

[3] I. Kamon, T. Flash, and S. Edelman. Learning visually guided grasping: A test case in sensorimotor learning. *IEEE Transactions on System, Man and Cybernetics*, 28(3):266–276, May 1998.

[4] Knoll, A. and B. Hildebrandt and J. Zhang, "Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language", in *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, NM, pp. 888–894, 1997.

[5] 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.

[6] M. A. Moussa and M. S. Kamel. An experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on System, Man and Cybernetics*, 28(2):239–253, May 1998.

[7] S. K. Nayar, H. Murase, and S. A. Nene. "Learning, positioning, and tracking visual appearance," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3237–3244, 1994.

[8] 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.

[9] 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.

[10] J. Zhang, 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.

| yellow cube, completely visible | | | |
|---|---|---|---|
| illumination | x[mm] | y[mm] | $\alpha$[degree] |
| optimal | 0.399 | 0.665 | 0.608 |
| worse | 0.595 | 1.525 | 2.606 |
| poor | 3.126 | 1.038 | 6.059 |

| yellow cube, 20% covered | | | |
|---|---|---|---|
| illumination | x[mm] | y[mm] | $\alpha$[degree] |
| optimal | 0.832 | 1.093 | 0.997 |
| worse | 0.524 | 2.373 | 1.141 |
| poor | 6.395 | 4.728 | 19.786 |

| blue cube | | | |
|---|---|---|---|
| illumination | x[mm] | y[mm] | $\alpha$[degree] |
| optimal | 1.658 | 0.946 | 1.481 |
| worse | 0.494 | 2.020 | 1.979 |
| poor | 1.006 | 0.928 | 10.803 |

| screw head | | | |
|---|---|---|---|
| illumination | x[mm] | y[mm] | $\alpha$[degree] |
| optimal | 0.630 | 0.535 | 1.850 |
| worse | 0.323 | 0.851 | 1.897 |
| poor | 0.610 | 0.751 | 1.281 |

| ledge with 3 holes | | | |
|---|---|---|---|
| illumination | x[mm] | y[mm] | $\alpha$[degree] |
| optimal | 0.272 | 0.728 | 0.452 |
| worse | 0.940 | 0.704 | 0.386 |
| poor | 1.198 | 0.612 | 0.404 |

Table 1
RMS-errors for the three objects under different lighting conditions. Controllers with three input dimensions and four linguistic terms for each dimension were used.