A General Learning Approach to Visually Guided 3D-Positioning and Pose Control of Robot Arms

In "Biologically Inspired Robot Behavior Engineering, edited by R. J. Duro, J. Santos and M. Graña, Physica-Verlag, 2003, Chapter 15, pp. 417-438.

Jianwei Zhang and Alois Knoll

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

Abstract. We describe a general learning approach to fine-positioning of a robot gripper in three-dimensional workspace using visual sensor data. This is a two-step approach: a) a hybrid representation for encoding the robot state perceived by visual sensors; b) partitioning the action space of the robot to let multiple specialized controllers evolve.
The input encoding consists of representing position by geometric features and uniquely describing orientation by combination of principal components. Such a dimension-reduction procedure is essential to apply supervised as well as reinforcement learning. A fuzzy controller based on B-spline models serves as a function approximator using this encoded input and producing the motion parameters as outputs.
A complex positioning and pose control task is divided into consecutive subtasks. Each subtask is solved by a specialized self-learning controller. The approach has been successfully applied to control 6-axes-robots translating the gripper in the three-dimensional workspace and rotating it about the z-axis. Instead of undergoing cumbersome hand-eye calibration processes, our system lets the controllers evolve using systematic perturbation motion around the desired position and orientation.

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 calibrated 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 for working 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. Normally, such a calibration procedure is time-consuming and not fault-tolerant. Every time when the hand-eye configuration is changed, the calibration should be performed. Such methods can only work well if:

- the hand-eye configuration is strictly fixed and completely known (including camera parameters) and
- the position/orientation based on these geometric features can be reconstructed.

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).

An alternative methodology is learning the transformation. Recently, neural network based learning has also found applications in grasping. The work in [7,11,3] used geometric features as input to the position 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 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 modeled 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 into effective factors and then into action values.

In [8] 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.

Reinforcement learning enables the trained system not only to be adaptive in the training phase like supervised learning methods. In fact, no training and on-line application are differentiated and the system learns life-long [4]. Although reinforcement learning has been applied to mobile robot systems [10], there are few applications to sensor-based manipulation tasks.

2 Experimental Setup

Learning of sensor-based elementary operations is motivated by building a demonstrative robot system for our project SFB-360 “Situated Artificial Communicators” [13]. Our robot system 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 to be grasped (simply called objects in the following parts), 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.

Each of the manipulators of type Puma 260 is equipped with a parallel jaw gripper and a hand-camera. The object is visible in the image of a “self-viewing” eye-in-hand camera (Fig. 2). The aim is to control the robot hand from its current

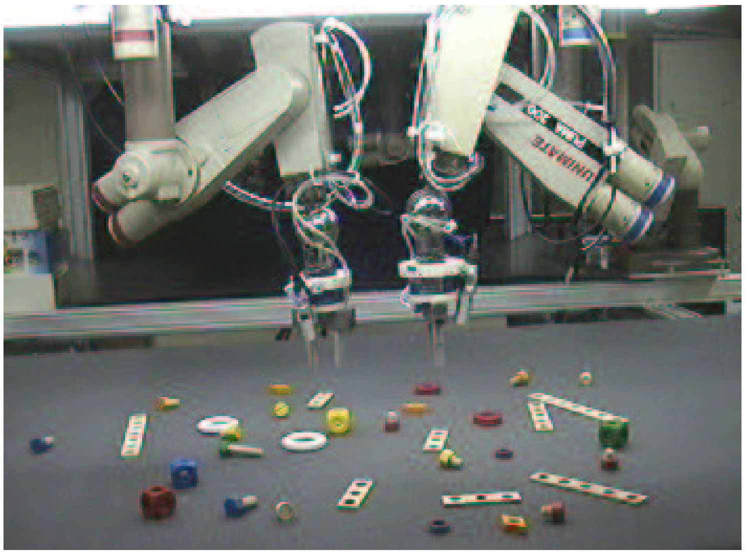


Fig. 1. The working cell of our robot system. The "Baufix"-parts on the assembly table are to be grasped by the robot gripper.

position/orientation (called position for short in the following parts) (Fig. 3 left) to a new position so that the hand-camera image matches the optimal grasping one (Fig. 3 right). In our setting there are some 20 different objects to be handled. Some of the objects 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 specialized algorithm for each object* and shows stable behavior under varying object brightness and color.

3 Problem Description

The vision based 3D positioning and pose control of robot arms is a perception-action mapping: $(x_1, \ldots, x_m) \to (t_x, t_y, t_z, r_x, r_y, r_z)$, where $x_1, \ldots, x_m$ are the input variables as read from the sensing device, t_x, t_y, t_z are the translational degrees of freedom along $x-$, $y-$ and $z-$directions in the tool coordinate system and r_x, r_y, r_z are the rotational degrees of freedom around $x-$, $y-$ and $z-$axes.

Two basic problems arise:

Input space: No universal function approximator models, e.g. RBFN, B-spline network, can be directly used due to the "curse of dimensionality". If each input variable x_j is covered with b_j basis functions, then the complexity of the model (number of the hidden nodes, number of rules etc.) is exponential in m: $b_1 \times b_2 \times \cdots \times b_m$.

Output space: A six dimensional output space results at least in the following difficulties for learning:

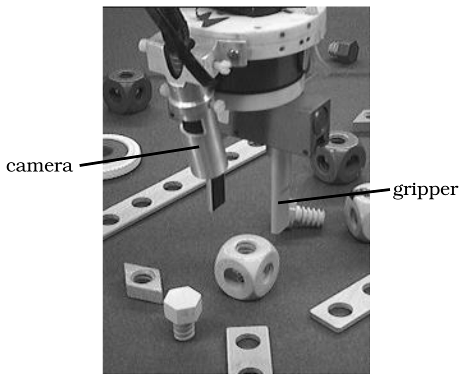


Fig. 2. The end-effector of the manipulator with a hand-camera (positioned optimally over the yellow cube).

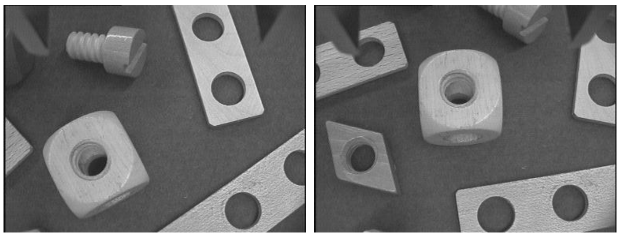


Fig. 3. A cube viewed from the hand-camera – before and after fine-positioning.

- To construct a monolithic controller using supervised learning, there exist no methods to find a reasonable number of commonly used input variables for all six DOFs.
- For reinforcement learning, the number of actions for selection is $3^6 = 729$ (each DOF with +, - or zero changes). The learning time increases dramatically.

Limited by the technically usable computer memory and available time for learning, directly applying a monolithic function approximator to this MIMO for positioning and pose control is not practical. We utilize the following two strategies to cope with these problems:

In input space: compact coding of robot state (described in section 4, 5) and
In output space: decomposition of the degrees of freedom (described in section 6).

4 Robot State Coding Using Geometric Features

The key step to apply reinforcement learning in manipulation tasks is finding a compact and unique coding of measurable states of a robot arm. Compactness of the robot state means that the number of the description parameters should be as low as possible so that the "curse of dimensionality" problem can be avoided. Only a unique coding can guarantee that the controller, after learning, works robustly. In the following sections, the necessary procedures for a general coding method with hand-camera vision data will be described in detail.

4.1 Extracting the Object in Focus from an RGB Image

The RGB images grabbed by the hand-camera cannot be used directly as the state vector because a) they contain much irrelevant information, and b) they consist of too many pixels. Therefore, the camera images are further processed. Firstly, the irrelevant information is filtered from the original image set. The result is represented as a binary image. The dimension reduction methods are used on the relevant information after the preprocessing (see sections 4.2 and 5).

For most of the grasping task it is sufficient if the binary image merely contains a *silhouette* of the object. Based on a sequence of preprocessing steps, the RGB-images from the hand-camera are converted into cleaned binary images.

To eliminate the disturbances in the background, e.g. the granulative structure in our setup, smoothing filters should be applied. After a comparative study of the median filter, mean filter and the binomial filter, we chose the mean filtering with the calculation of the new pixel value.

The next important step is to separate the object in focus from the background. In this work we consider colorful objects and need a method which works for different colors. It can be observed that the maximal difference between the RGB values is an indicator for "how colorful" a pixel is. Such a maximal difference can be used:

$$f_{\mathrm{MaxDiff}}(r,g,b) = \max_{x,y\in\{r,g,b\}}(|x-y|) \tag{1}$$

After the binarization, some remaining disturbances can still be found in the background of the binary image, especially if the small objects are separated or the color of the object is dark. Based on the assumption that the objects are solid and have thus closed segments, a filtering approach is applied again to eliminate the pixels labeled with 1 which do not have the eight neighborhoods with value 1.

All these procedures can be summarized in Fig. 4.

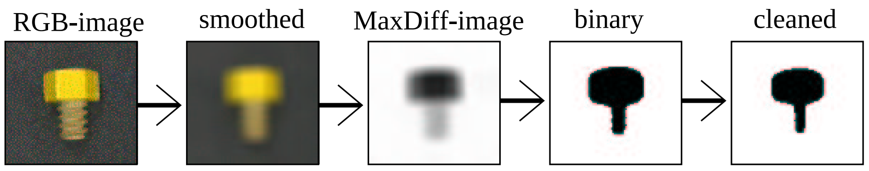


Fig. 4. Overview of all image processing steps.

4.2 Geometric Features

After a binary image is generated which contains only the silhouette of the object, geometric features such as area, center of gravity (COG) and, under certain circumstances, orientation can be computed.

The general definition of moments of order $p + q$ $(p, q \geq 0)$ is as follows:

$$M_{pq}(x, y) = \iint_B b(x, y)\ x^p\ y^q\ dx\ dy \tag{2}$$

Height Perception The area of an object in a binary image can be calculated based on the zero-order moment $M_{00}(x, y)$. If the height of the hand/camera relative to the object changes, the area of the object is the best measure of the height information. Therefore, the M_{00} will be used for coding the height.

Planar Position Assume that $(\bar{x}, \bar{y})$ describes the COG. Then, they can be calculated using the zero-order and first-order moments:

$$\bar{x} = \frac{M_{10}(x, y)}{M_{00}(x, y)}, \qquad \bar{y} = \frac{M_{01}(x, y)}{M_{00}(x, y)} \tag{3}$$

The computation of COG works for coding the planar position of arbitrary objects.

Orientation The calculation of the orientation is based on the determination of an axis along which the spacial expansion is the largest. Since the axis representing the orientation we are looking for should intersect the COG, we first transform the coordinate system to the COG system:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \tag{4}$$

The orientation θ can be defined as the angle between the $x-$axis and the axis to be searched. There exists the following relation:

$$\tan 2\theta = \frac{b}{a - c} \tag{5}$$

where

$$a = M_{20}(x', y') \tag{6}$$
$$b = 2M_{11}(x', y') \tag{7}$$
$$c = M_{02}(x', y') \tag{8}$$
$$\tag{9}$$

Thus θ is derived as:

$$\theta = \frac{1}{2} \arctan \left(\frac{b}{a - c} \right) \tag{10}$$

if it is not the case that $b \simeq 0$ or $a \simeq c$. Otherwise, the object possesses no distinctive axis, meaning that it is "too round".

While the COG can be used for describing position of any objects, the orientation computation can be only solved for long objects. In the next section, we propose an appearance-based, automatic method to encode the orientation of arbitrary objects.

5 General Automatic Orientation Coding

5.1 PCA Approach

To reconstruct the robot position, methods for computing geometric parameters can be used directly. For describing the orientation of the object, it is often the case that no obvious main axes can be found stably if the shape of the object does not possess a long axis. The following method based on PCA can enable a general orientation coding using minimal parameters.

This method is based on offline learning. First, a sequence of images of the object is taken. The object is placed approximately at the same position in each image but at different orientations. The image lines are concatenated, and the images are represented as vectors. We consider only the first l principal components $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_l$ of these vectors. Because the main difference between the images lies in the orientation of the object, the first l principal components contain the orientation information. All images are projected into the *Eigenspace* which these l principal components construct. These projections serve as the candidates for coding of the object's orientation.

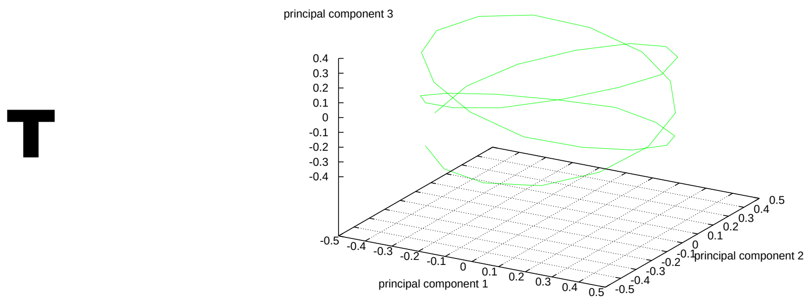


Fig. 5. Example of a manifold in the subspace constructed by the first three principal components which is generated by rotating the object on the left.

Assume that an image series of an object is taken, where the object in each image is rotated several degrees further and is located in the same starting orientation at the end. This way, the projections of these images in the eigenspace form a continuous, closed manifold (Fig. 5). Based on this observation, a correct coding is achieved.

However, the "wired" spiral structure raises the question here, if these principal components are sufficient for a unique coding. Since the approach should be applied in arbitrary objects, the following question needs to be answered in general: How many and which principal components are necessary for a unique coding of the object's orientation? As stated above, as few principal components as possible should be used to achieve the most compact coding.

5.2 Implementation of PCA

The PCA is implemented by interpreting each of the k training images as a vector $\boldsymbol{x^i}$, 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. In [8] a procedure is described for computing the first l most important eigenvectors and eigenvalues of this covariance matrix without computing the matrix itself. Of these l eigenvectors corresponding to the l largest eigenvalues we use only a subset.

Let us assume k sample input vectors $\boldsymbol{x^1}, \ldots, \boldsymbol{x^k}$ with $\boldsymbol{x^i} = (x_1^i, \ldots, x_m^i)$ 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 $\boldsymbol{\mu}$ and the covariance matrix $\mathbf{Q}$ of these vectors are computed according to

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

The eigenvectors and eigenvalues can then be computed by solving

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

where λ_i are the m eigenvalues and $\boldsymbol{a}_i$ 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 $\boldsymbol{x^i}$ is equal to isolating those first l ($l < m$) eigenvectors $\boldsymbol{a}_i$ with the largest corresponding eigenvalues λ_i. If we now define a transformation matrix

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

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

$$\boldsymbol{p^i} = \mathbf{A} \cdot \boldsymbol{x^i}; \qquad dim(\boldsymbol{p^i}) = l$$

The dimension l should be determined depending on the discrimination accuracy needed for further processing steps vs. the computational complexity that can be afforded.

5.3 Combining Two Principal Components

Firstly, a method can be found to convert the two components of a data point into a single parameter. Based on the observation that all the points are located on a circle around a coordinate system, we use the following function to transform two principal components into a one-dimensional value:

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \frac{\pi}{2} & \text{if } x = 0 \ \wedge \ y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \ \wedge \ y < 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \ \wedge \ y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \ \wedge \ y < 0 \end{cases} \tag{11}$$

This function maps each point (x, y) on the plane to the angle between the line through the point and origin, and the positive $x-$axis. It is obvious that the desired coding function should fulfill the following conditions:

- It is sufficiently continuous and smooth. Similar data points are mapped to similar function values.
- It is definite. There are no two different data points which are mapped to the same function value.

5.4 Search for Principal Component Pairs

To verify the two-to-one mapping from a principal component pair to a unique atan2 value, we need an algorithm to check how many periods the atan2 graph possesses.

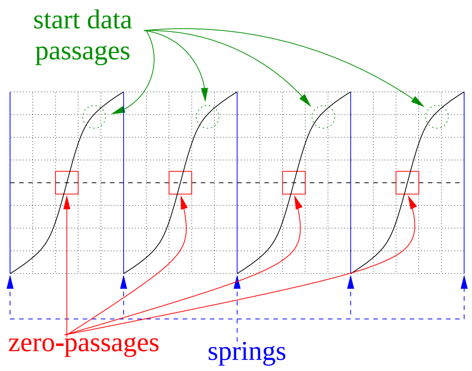


Fig. 6. Prominent positions in an atan2 graph. *Zero-passages* are parts where the function value of the interpolating curve intersects the $x-$axis in the monotonous direction. *Springs* divide the monotonous phases. *Start data passages* are positions where the graph passes by the atan2 value of the first data point in the monotonous direction.

Firstly, this graph possesses a preferred *monotonous direction*: if the most trends from a calculated value to the next step shows upwards, then the monotonous direction is increasing, otherwise it is decreasing. We can observe three types of prominent parts in the graph (Fig. 6). Based on this, the following key figures can be identified:

- z: the number of zero-passages,
- s: the number of the springs, and
- t: the number of the passages of the start data point in the monotonous direction.

Using these three key figures, it can be determined if a circular structure is observed as well as how many circles there are. These key figures in an arbitrary graph are uncorrelated. In a graph like in Fig. 6, their differences can be at most one. If the maximal difference between z, s and t is larger than one, then no circular structure is present.

We propose an algorithm to determine the period number using z and s. If they have the same value, the value can be used. Additionally t should be used for the final decision. It can be shown that the variable t is always the number of periods minus one, if s and t are the same. Fig. 7 illustrates the complete algorithm.

5.5 Selection of Inputs for State Encoding

Here we investigate if sufficient information is available to uniquely encode the orientation of an object. By the arguments in the above discussions, it can be shown that this will be the case when we have found a principal component pair whose atan2 function only possesses a single period, as in Fig. 8. In this case the orientation can be represented with one parameter.

However, atan2 functions with multiple periods would make the mapping ambiguous. Different orientations of the object would be encoded by the same state. Fortunately, we can combine several atan2 functions. By experiments it can be observed that it is sufficient in many cases to combine two such functions (if they exist for the object) to find a unique coding.

Consider the following analogy: the function $\sin(x)$ has the period 2π and the function $\sin(kx)$ has the period $\frac{2\pi}{k}$. Two functions $\sin(k_1x)$ and $\sin(k_2x)$ have the periods $\frac{2\pi}{k_1}$ and $\frac{2\pi}{k_2}$. If we combine these two functions, e.g. by adding them, then the periodicity of the combined function is the smallest common multiple (SCM) of each periodicity:

$$\mathrm{SCM}\left(\frac{2\pi}{k_1}, \frac{2\pi}{k_2}\right) = 2\pi \cdot \mathrm{SCM}\left(\frac{1}{k_1}, \frac{1}{k_2}\right) \tag{12}$$

(12) is 2π, if k_1 and k_2 have no common divisor. This means that although the periodicities of both single functions can be divided by the factor k_1 or k_2, the combination of these two functions has the complete periodicity.

To achieve the periodicity 2π by combining three functions, at least two of the three k_i should have no common divisor. Since the combination of these two

```
$z := 0$
$s := 0$
$t := 0$
for $i = 1$ to (length(array) $- 1$)
    $l :=$ array[$i$]
    $r :=$ array[$i + 1$]
    if the monotonous direction is "increasing" then
        if $l > r$ then $s := s + 1$
        if $(l < 0) \wedge (r > 0)$ then $z := z + 1$
        if $(l <$ array[1]$) \wedge (r >$ array[1]$)$ then $t := t + 1$
    else
        if $l < r$ then $s := s + 1$
        if $(l > 0) \wedge (r < 0)$ then $z := z + 1$
        if $(l >$ array[1]$) \wedge (r <$ array[1]$)$ then $t := t + 1$
    end if
end for
if $\max_{x,y \in \{z,s,t\}} |x - y| > 1$ then
    return
else if $z \neq s$ then
    $p := \max\{z, s\}$
else
    $p := t + 1$
end if
```

Fig. 7. Algorithm for computing the periodicity of an atan2 mapping of a set of data points, the array contains the calculated atan2 value of the data points, "return" means that the data points possess no circular structure.

functions already has the periodicity 2π, the third function is redundant. The same applies if the atan2 functions are used.

To summarize, we propose the following strategy to find the minimal coding for the orientation of an object:

1. Search for a principal component pair whose atan2 function shows a single period.
2. If step 1 fails: search for two principal component pairs whose numbers of period have no common divisor.
3. If step 2 fails again, then use the first l principal component for the state encoding.

6 Partition of DOFs

Generally, a robot arm needs six DOFs to grasp a object from any orientations and positions. For clarity, we first show the solution for three DOFs.

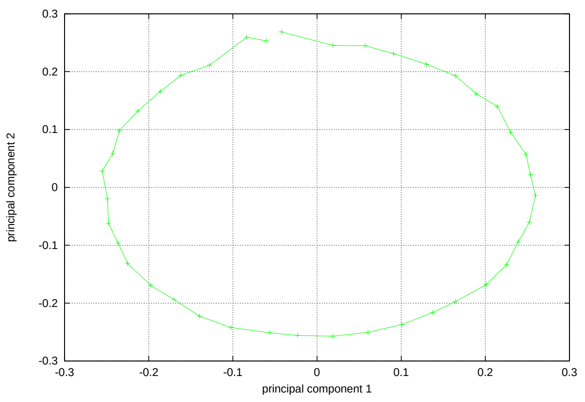


Fig. 8. The subspace constructed by the first and second principal components of the image sequence of a cubic object.

6.1 Three Dimensional Cases

To grasp quasi-planar objects, we assume that the gripper is perpendicular to the table and its height is known. There are still three DOFs to control the robot arm: motions parallel to the table plane (x, y) and the rotation about the axis which is perpendicular to the table (θ).

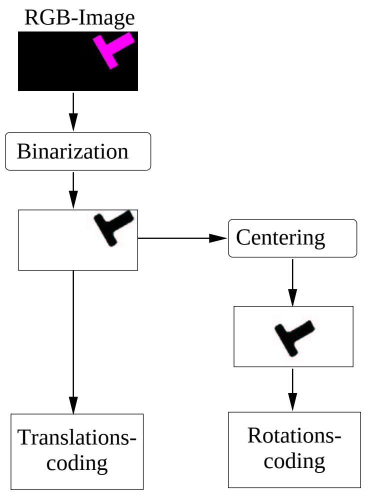


Fig. 9. Example of controlling x, y, θ. The pre-processed images are additionally centered for the rotation coding.

In order to generate a small action space, learning is divided between two learners: one for the translational motion on the plane, the other for the rotational motion (Fig. 9). The *translation-learner* then has four actions to select (two in x- and two in y-direction). The *rotation-learner* has two actions to select (rotation clockwise and counter-clockwise). Such a partition has the following advantages compared with a monolithic learner:

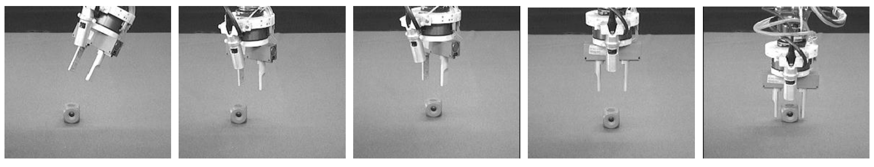
- The state space is significantly smaller.
- The state coding is so designed that the state vectors merely contain the relevant information for the corresponding learner.

In practice, the two learners are applied alternately. First, the translation-learner is applied with long learning steps until it achieves its goal defined with its state coding. The translation-learner is then replaced by the rotation-learner which is applied similarly with long learning steps until its goal is achieved. At this moment it may occur that the translation goal state is disturbed by the rotation-learner. Therefore, the translation-learner is activated once again. The procedures are repeated until both learners report that they all arrived the goal state. This state is thus the common goal state.

6.2 Extension to Six DOFs

To utilize all six DOFs, additional learners should be introduced. Fig. 10 shows how such a positioning can be realized in four steps.

1. The first learner possesses two DOFs and its task is that the object will be observed from a certain perspective. Typically for a planar table, this means to position the gripper perpendicularly to the table surface (a $\rightarrow$ b).
2. Apply the x/y learner (b $\rightarrow$ c).
3. Apply the θ rotation-learner (c $\rightarrow$ d).
4. The last height-learner will correct the z-coordinate, the height over the table (d $\rightarrow$ e).



(a) (b) (c) (d) (e)

Fig. 10. Positioning and pose control with 6 DOFs in four steps.

7 The Perception-Action Transformation

7.1 The Neuro-Fuzzy Model

Our experimental results show under the most diverse conditions that we can extract geometric features based on the calculations of moments to encode the positioning information and to find non-geometric parameters based on combining principal components. Therefore, if the input is high-dimensional, an effective dimension reduction can be achieved by projecting the original input space into a minimal subspace.

Variables in the subspace can be partitioned by covering them with linguistic terms (the right part of Fig. 11). In the following implementations fuzzy controllers constructed according to the B-spline model are used [12]. This model provides an ideal implementation of the CMAC model (cerebellar model articulation controller) (proposed by Albus [1]). We define linguistic terms for input variables with B-spline basis functions and for output variables with singletons. This method requires fewer parameters than other set functions such as trapezoid, Gaussian function, etc. The output computation is very simple and the interpolation process is transparent. We also achieved good approximation capabilities and rapid convergence of the B-spline controllers.

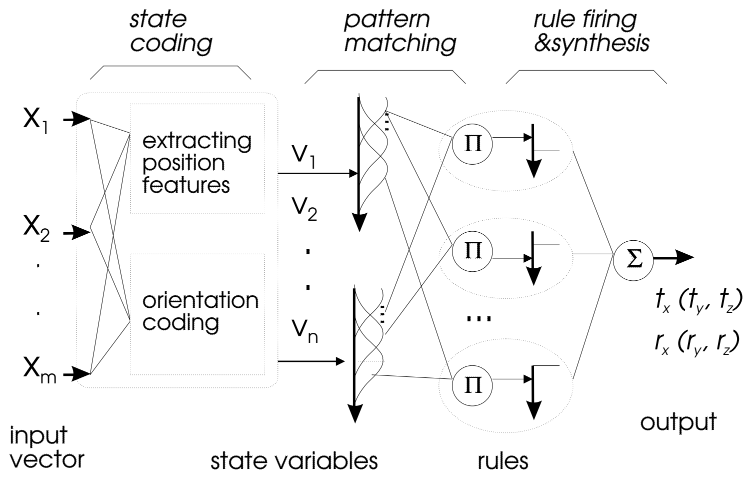


Fig. 11. The task oriented mapping can be interpreted as a neuro-fuzzy model. The input vector consists of pixels of a binary or grey-scale image.

7.2 Supervised Learning

Since robot manipulators possess a high repeatability, they can be easily programmed to systematically generate training images as well as training data. If the hand-eye configuration remains the same, supervised learning is a method which needs only little training time. The working systems implements two phases: off-line training and on-line evaluation. In the off-line phase a sequence between ten and one hundred training images showing the same object in different positions is taken automatically, i.e. without human intervention. For each image the position vector of the manipulator with respect to the optimal grasp point for the current object is recorded.

In the on-line phase the camera output is transformed into the hybrid state space based on geometric features and combination/transformation of principal component pairs. The state variables are then processed by the fuzzy controller. The controller output is the end-effector's position and angle correction (Fig. 12).

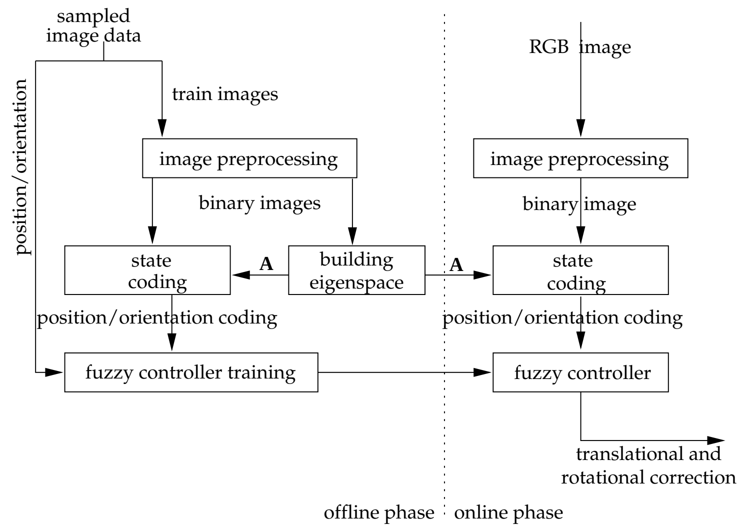


Fig. 12. The training and the application of the PCA neuro-fuzzy controller using supervised learning.

With the $\boldsymbol{x}^i$ and the corresponding $t_x, t_y, t_z, r_x, r_y, r_z$, 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 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 [12].

7.3 Reinforcement Learning

As described above, supervised learning is based on training examples with sensor input and actuator output. This approach is adaptive only during the training phase. In the application phase, the system will not react to the changes by itself. The changes can be the modification of the hand-eye configuration, or the object should be grasped in another way. In these cases, the controller must be trained with new examples again. Using *reinforcement learning*, no differentiation will be made between the training and application. The system learns continuously. No training examples are necessary. What is needed is only to give a *reward* for a successful grasping, e.g. correct positioning in this work.

During all the motion time, the learner improves its behavior permanently from the state determination and action execution. The current state is determined by grabbing an image, preprocessing it and transforming it to a state vector. This cycle is relatively computationally expensive. Therefore, it is desirable that the learning goal can be achieved with as few camera images as possible. Thus we propose the *variable learning steps*: if the robot is far from the goal, the stepsize is set larger; if the robot is near the goal, the stepsize is correspondingly reduced.

The maximal and minimal stepsize are defined and the Euclidean distance of an arbitrary state to the goal is estimated. The current action step is selected as proportional to the distance between the current state and the goal. Similar as the Reward Distance Reduction which assumes that states are so encoded that similar states have the similar codings. This condition is fulfilled by using the coding approach described in section 4 and 5.

7.4 Implementation Issues of Reinforcement Learning

The reinforcement learning approach has been implemented as an object hierarchy (Fig. 13). Each of the components is defined as follows:

qlearner: The class `qlearner` defines the basic interface for all the learner classes which employ Q-Learning. In these modules the stochastic, undirected action selection (13) can be used:

$$P(a|s) = \frac{e^{\frac{\hat{Q}(s,a)}{T}}}{\sum\limits_{a' \in A} e^{\frac{\hat{Q}(s,a')}{T}}} \tag{13}$$

where s is the robot state, a is the action to be executed from the action set A, $\hat{Q}(s,a)$ is the estimated Q value using an update rule, and the parameter T is called **exploration temperature**.

qlearnerER: The class `qlearnerER` extends the class `qlearner` to include all the necessary components for Experience Replay, Reward Distance Reduction and Backstep Punishment. They contain a memory for experiences and methods to record and recall of experiences.

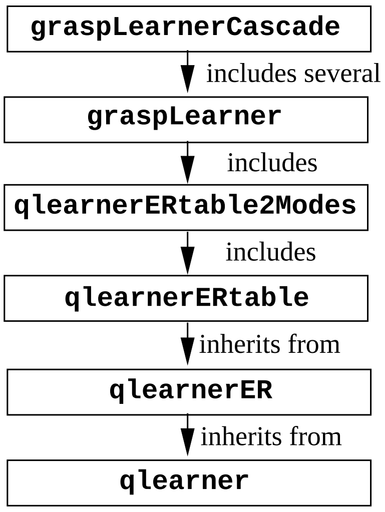


Fig. 13. Hierarchy of the learner objects.

qlearnerERtable: The class `qlearnerERtable` implements the table of the $\hat{Q}$-value.

qlearnerERtable2Modes: The class `qlearnerERtable2Modes` is a container-class for more `qlearnerERtable` objects. For example, one object learns the translational motion, another learns the rotation.

graspLearner: The class `graspLearner` includes a `qlearnerERtable2Modes` object.

graspLearnerCascade: The class `graspLearnerCascade` realizes a learner cascade. It is a container-class for multiple `graspLearner` objects.

Based on this object hierarchy, grasping experiments with real objects (Fig. 14) were successfully carried out.

8 A Complete Recognition-Grasping Example

We now show a complete sample run of the hybrid system composed of a recognition subsystem and the learned grasping controllers. For the scene shown in Fig. 15 all steps are performed in an integrated way: a top view image is taken (Fig. 15a); the recognition system based on fuzzy invariant indexing (FII) [6] detects the objects of interest (Fig. 15b); the manipulator moves approximately above the object; the evaluation of hand-camera images guides the gripper directly to the grasp position; the object is grasped.

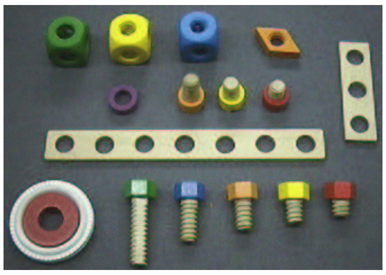
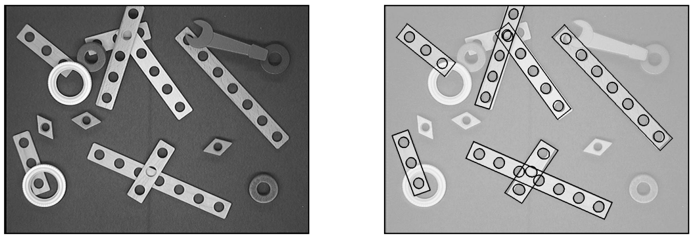


Fig. 14. These Baufix contruction parts are used as test objects.



(a) Original Image

(b) Result

Fig. 15. Test scene (left) and recognized slats (right).

The task to be solved is the grasping of a slat (Fig. 15a). The FII-recognition system provides all the slats in the image (occluding each other or not), see Fig. 15b.

As we have not yet implemented heuristics that automatically pick a slat according to some given criteria, we manually choose the one that is not difficult to grasp: the 3-hole-slat lying on top of the 7-hole-slat. Figure 16 shows a part of the image sequence taken by the hand camera; the right row shows the results with the 7-hole-slat clipped based on the position information obtained by the FII-recognizer.

Finally, as shown in Fig. 17, the slat is grasped successfully.

9 Conclusions

The innovation of the approach lies in the automatic coding of robot states and the application in the visually guided grasping. While the representation based on purely geometric features is suitable for long objects, this approach can be applied to objects with arbitrary shapes, also when no robust geometric features can be extracted. Based on this representation, the robot states during the visually guided motions are represented with the minimal number of parameters. Supervised as well as reinforcement learning can be therefore put into practice for a wide range of grasping operations. This approach has the following advantages over classical approaches:

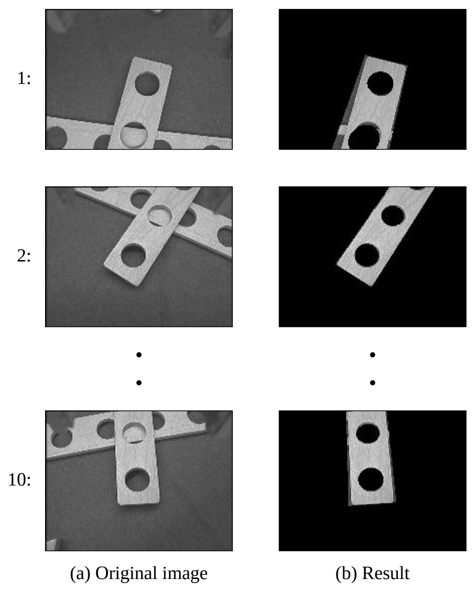


Fig. 16. Images of the hand camera taken at different time stamps.

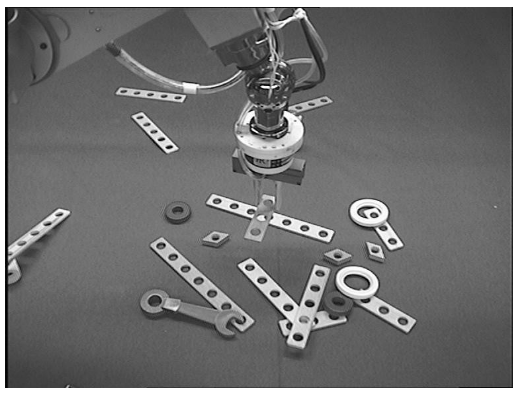


Fig. 17. Grabbed 3-hole-slat

Calibration-free. The camera need not be calibrated with respect to the hand/arm.

Minimal state coding. Three geometric features from the camera images are suggested for representing the 3D position of the robot-object relation. The method using combination of principal components and transformation to an atan2 function value is universal for arbitrary object shapes. The minimal number of orientation state coding can be found by our algorithm.

Model-free. No model for recognizing 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.

Our current work aims at introducing a 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 minimize the learning error. To enhance each local controller in diverse complex environments, we are also working on adding further components to 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.

In the future, the complete quality of a grasping trial will be automatically evaluated by active cameras and active test motions. After the grasping, other cameras which have good viewing points can take images which are further processed to estimate the grasping quality. As grasping with a human hand, an optically correct positioning over an object does not guarantee a stable grasping. Test motions can be programmed for the grasping hand, e.g. along the normal, orientation and approach vectors. The cameras should monitor the grasp fingers and the object continuously. By using visual monitoring and force-guarded motion, the grasping hand can actively place the object on the table. The slip motion between the object and the hand can be detected by evaluating the force/torque sensors. The possible large position changes can be found by the hand-camera.

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 to grasp using visual information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2470–2476, 1996.
4. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, Mai 1996.

5. 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.
6. A. Knoll, J. Zhang, T. Graf and A. Wolfram. *Recognition of Occluded Objects and Visual Servoing: Two Case Studies of Emplying Fuzzy Techniques in Robot Vision.* In "Fuzzy Techniques in Image Processing", edited by E.E. Kerre and M. Nachtegael, Springer Verlag, 2000.
7. 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.
8. 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.
9. 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.
10. Sebastian Thrun and Anton Schwartz. Finding Structure in Reinforcement Learning. In *Advances in Neural Information Processing Systems 7*, pages 385–392, 1995.
11. 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.
12. 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.
13. J. Zhang, Y. von Collani, and A. Knoll. Interactive assembly by a two-arm robot agent. *Journal of Robotics and Autonomous Systems*, 29:91–100, 1999.