Biological Cybernetics manuscript No.
(will be inserted by the editor)

Learning visuomotor transformations for gaze-control and grasping

Heiko Hoffmann[1], **Wolfram Schenck**[2], **Ralf Möller**[2]

[1] Cognitive Robotics, Department of Psychology, Max Planck Institute for Human Cognitive and Brain Sciences, 80799 Munich, Germany
[2] Computer Engineering Group, Faculty of Technology, Bielefeld University, 33594 Bielefeld, Germany

Received: 1 Feb 2005 / Accepted: 14 Apr 2005

Abstract For reaching to and grasping of an object, visual information about the object must be transformed into motor or postural commands for the arm and hand. In this paper, we present a robot model for visually guided reaching and grasping. The model mimics two alternative processing pathways for grasping, which are also likely to co-exist in the human brain. The first pathway uses directly the retinal activation to encode the target position. In the second pathway, a saccade controller makes the eyes (cameras) focus on the target, and the gaze direction is used instead as positional input. For both pathways, an arm controller transforms information on the target's position and orientation into an arm posture suitable for grasping. For the training of the saccade controller, we suggest a novel staged learning method which does not require a teacher that provides the necessary motor commands. The arm controller uses unsupervised learning: it is based on a density model of the sensor and the motor data. Using this density, a mapping is achieved by completing a partially given sensorimotor pattern. The controller can cope with the ambiguity in having a set of redundant arm postures for a given target. The combined model of saccade and arm controller was able to fixate and grasp an elongated object with arbitrary orientation and at arbitrary position on a table in 94% of trials.

1 Introduction

Eye-hand coordination in visually guided reaching has attracted increasing attention in neuroscience in recent years (Lacquaniti and Caminiti, 1998; Batista et al., 1999; Snyder, 2000; Snyder et al., 2000; Buneo et al., 2002; Carey et al., 2002; Crawford et al., 2004). Reaching and grasping require a target object to be represented in arm coordinates (these could be the joint angles[1]); in contrast, the object is given in retinal (eye-centered) coordinates. Thus, the brain needs to transform eye-centered coordinates into arm coordinates. To understand how this transformation takes place, neuroscience addressed the question of which coordinate system is used at a specific stage in the process. In a monkey study, Batista et al. (1999) reported that 81% of the recorded neurons in the parietal reach region (the reach-planning area in the posterior parietal cortex) encode location in eye-centered coordinates. However, reaching must also include the position and orientation of the eyes relative to the body frame. In the brain, the position on the retina and the orientation of the eyes are represented by population codes (Zipser and Andersen, 1988). To obtain the spatial location of targets, the retinal position is modulated by the population code of the gaze direction. This modulation has been termed 'gain fields' (Snyder, 2000). It is possible to combine the two population codes with a neural network to obtain a head-centered position code (Zipser and Andersen, 1988).

On the other hand, a processing pathway without retinal position coding is suggested by the following experiment. Carey et al. (1997) studied the case of patient D., which had widespread cortical atrophy (see also the review by Carey et al. (2002)). D. was visually aware of targets outside her fovea. However, if asked to reach to these targets, she would reach to the point she was fixating ('magnetic misreaching'). Reaching to targets in the fovea does not require coding in eye-centered coordinates; instead, the gaze direction is sufficient for location coding (for a fixed head position). The same seems to hold for healthy humans, which usually fixate an object before grasping it. Apparently, there are multiple redundant pathways for reaching in the brain.

Here, we present a model that can realize both pathways: One path uses only the retinal information (for a

[1] Behavioral data suggest that arm trajectories are planned in these kinematic coordinates rather than in dynamic coordinates, like muscular activation (Wolpert et al., 1995).

fixed gaze-direction); the other path uses the gaze direction instead. In our model, a robot learns to fixate, reach for and grasp an object presented visually (Schenck et al., 2003). One of the pathways needs a *saccade controller* which brings the object into the fovea, and both pathways include an *arm controller* which associates an arm posture with the image of the object.

The learning of these controllers has to cope with three major problems. First, for saccades, there is no teacher that can provide the correct motor command for a given target location. Second, the arm controller needs to map a high-dimensional camera image (where every pixel represents a dimension) onto an arm posture. Third, the kinematic arm problem is redundant, that is, several arm postures exist for a given object location and orientation. Thus, the controller has to learn a one-to-many mapping. Function approximators, specifically feed-forward neural networks like the multi-layer perceptron, fail to learn such a relationship. A function approximator can only average over multiple output postures, and the average of redundant joint-angle sets is generally not appropriate (Movellan and McClelland, 1993; Hoffmann and Möller, 2003). These problems are not restricted to the robot setup we present; they are of wider interest in robot control, and also many biological systems have to cope with them. In this article, we present general learning strategies suitable to tackle these problems.

As a solution for the missing teacher signal, we present a staged learning procedure. This learning mechanism is illustrated in the example of a saccade controller, which is modeled by a multi-layer perceptron. The network learns the mapping from target coordinates to motor commands. At each learning stage, the saccadic movements are obtained by adding random movements to the output of a previously learned controller. To generate new training patterns for the network, only those movements are selected that bring the target toward the fovea within a quality threshold. From stage to stage, the quality threshold gets more restrictive, and the pattern set and the controller improve. In the training of each stage, we exploit the property of feed-forward neural networks to average over multiple outputs. Thus, over- and undershoots that occur for a given target position partly compensate, and the resulting average is closer to the optimal motor commands.

To cope with the mapping from a high-dimensional camera image to an arm posture, we pre-process the image. The image processing extracts information on the position and orientation of the target object. Both position and orientation are encoded with a population code, which can be gained by locally processing the image.

For the mapping itself, we propose a new learning and recall scheme that can cope with the redundancy in the output. The new mechanism operates like a recurrent neural network: stored patterns act like attractors for a pattern completion process. In each training pattern, corresponding sensor and motor components are integrated, forming a sensorimotor pattern. In the training phase, the distribution of these patterns is approximated by a density model. Given such a density model, a partially given sensorimotor pattern (the input) can be completed uniquely with a new recall mechanism (Hoffmann and Möller, 2003). The completion yields the output in its components. In the example of the arm controller, the input pattern is derived from the image of the target object and from the gaze direction for one of the pathways; the output pattern is the grasping posture.

The remainder of this article is organized as follows: section 2 describes the architecture of the overall system. Sections 3 and 4 present the details of saccade and arm controller, respectively. Section 5 presents the result of the robot experiments. These results are discussed in section 6, which also relates the model to the literature.

2 Overall system architecture

Our model contains two alternative pathways: the *retinal pathway* and the *gaze pathway*. The retinal pathway assumes that the gaze direction is fixed and uses only preprocessed image information to obtain an appropiate grasping posture (Fig. 1, left). In the gaze pathway, a saccade controller fixates the target, and the gaze direction and shape information from the image are used to associate a grasping posture (Fig. 1, right).

2.1 Experimental setup

The setup consists of two cameras mounted on a pan-tilt unit (*Directed Perception*) and a six degrees-of-freedom (6 rotatory joints) robot arm (*Amtec Robotics*) with a linear two-finger gripper (Fig. 5). The cameras were mechanically fixed relative to each other. Therefore, vergence movements had to be realized by software (section 3.1). The camera pictures are in color and have a resolution of 320×240 pixels (Fig. 12 shows the view from the left camera). Throughout the experiment, the illumination in the room was kept constant.

To train and test the saccade controller, 42 colored wooden bricks were placed on a table in front of the cameras and the arm. For grasping, however, only a single elongated red brick ($74 \times 24 \times 24$ mm) was used.

2.2 Retinal pathway

To grasp an object from the table via the retinal pathway, an image is taken with the left camera (Fig. 1, left). From this image, information about the object's position and orientation are obtained. Given this information, the arm controller associates (open loop) a sequence of two grasping postures: pre-grasping and grasping. To grasp the object, the arm is first moved into the pre-grasping and then into the grasping posture.

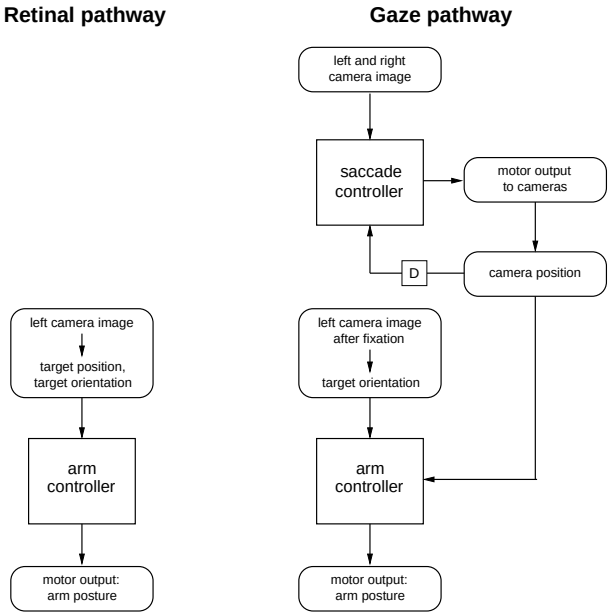

Fig. 1 Information flow in retinal and gaze pathway. The box labeled *D* delays the feedback by one time step.

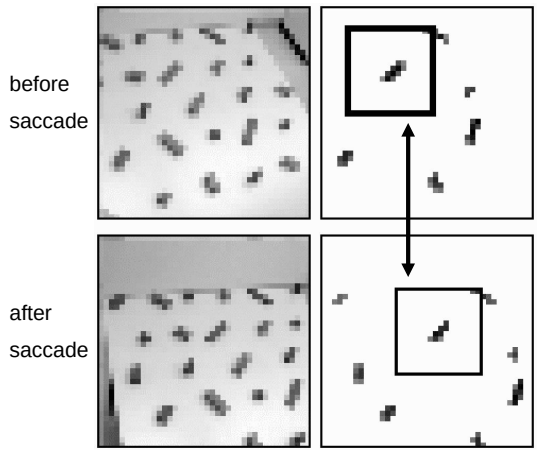

Fig. 2 Target re-identification. After a saccade, cross-correlation—calculated in the salience image for each color, over the areas marked by black boxes—was used to find the target again in the new image. The images on the right are the salience images for red.

2.3 Gaze pathway

For the gaze pathway, the saccade controller has to select and fixate the target object (Fig. 1, right). This controller operates in a feed-back loop until the fixation is optimal. The resulting gaze direction is described by pan, tilt, and vergence values. Afterwards, an image is taken with the left camera. From this image, information about the object's orientation is extracted (as in the retinal pathway). This information—together with the pan, tilt, and vergence values—is fed to the arm controller, which associates a grasping posture. To grasp the object, the arm is moved as described for the retinal pathway.

3 Saccade controller

The task of the saccade controller is to move the cameras such that a selected target object is fixated. In a typical fixation trial, first, the target object is located. Then, the saccade controller processes the sensory input and generates a motor command that attempts to center the object in both cameras. After re-locating the target object, either the fixation success is evaluated (in controller learning), or another saccade is started until the fixation is successful; that is, the controller carries out a series of saccades until the target object is centered.

3.1 Image processing

The image processing is adjusted to a special scenery made of colored wooden bricks on a white background. The definition of a goal color (red, green, or blue) served as a simple attention mechanism. This color determined from which class of visible objects a fixation target was selected.

In the first processing step, software vergence was achieved by extracting a quadratic region from each camera image; this region covered nearly the full height of the image, while its horizontal position depended on the vergence value. The horizontal vergence offset in one camera image was mirror symmetric to the offset in the other. Next, the obtained vergence regions were sub-sampled to 55×55 pixels. Afterwards, a contrast mechanism enhanced the goal color and produced gray-scale images (for example, $R - (G + B)/2$ was used to enhance red; R, G, and B stand for the red, green, and blue channels). The resulting images for one goal color are called 'salience images'.

Before a saccade, a fixation target has to be identified. In one of the two salience images from both cameras, the object was chosen that matched the goal color. If several objects had the same color, one of them was selected at random. The corresponding position of the target object in the salience image of the other camera was determined by searching for the neighborhood region (size: 24×24 pixels) with the maximum cross-correlation to the neighborhood of the selected target. The cross-correlation was computed in the salience images for all three goal colors separately, and then the three values were multiplied. After a saccade, when the visual input had changed, the target object was re-identified in both salience images using the same cross-correlation method (Fig. 2).

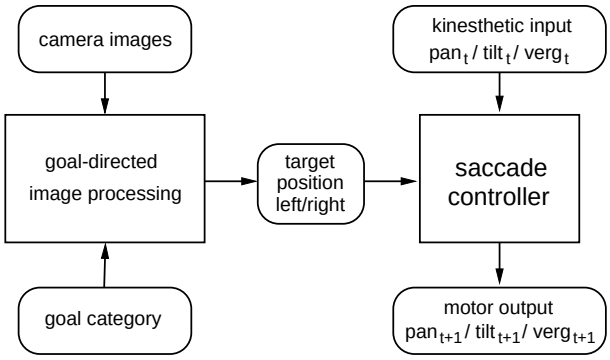

Fig. 3 Saccade controller input-output scheme. (Figure adapted from Schenck et al. (2003).)

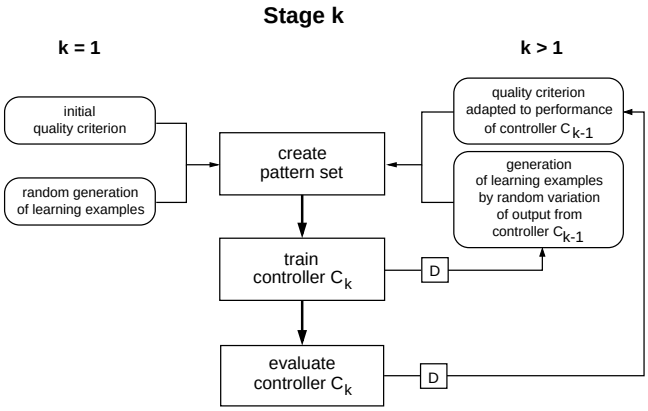

Fig. 4 Basic steps of staged learning. The boxes labeled D delay the information by one learning stage.

3.2 Controller input and output

The camera system had three degrees of freedom: the pan and the tilt angle of the camera head, and the vergence offset in both images. The sensory input to the saccade controller was both kinesthetic[2] and visual (Fig. 3). The kinesthetic part contained the current pan, tilt, and vergence values of the camera system (pan_t, $tilt_t$, $verg_t$ at the current time step t); these values are sufficient to describe the position of the cameras. The visual part also contained three parameters, the image coordinates of the center of mass of the selected fixation target in both the left and the right salience image ($x_{l,t}$, $x_{r,t}$, $y_{lr,t}$). Due to the design of the camera system, both vertical coordinates were given by just one parameter $y_{lr,t}$. The motor output of the saccade controller contained the parameters pan_{t+1}, $tilt_{t+1}$, and $verg_{t+1}$ for the next time step $t+1$.

3.3 Controller training

In our model of saccade control, the learning needs to cope with two problems. First, there is no teacher that could provide the correct motor commands and thus perfect learning examples. Like the biological system, our model has no feedback that could map the fixation error after a saccade into motor space. Second, a brute force search in the motor space to find perfect learning examples is far too time-consuming. We solved these problems by using less-than-perfect learning examples (under- and overshoot saccades) for controller training, and by exploiting the averaging properties of a multilayer perceptron (MLP). The MLP averaged over under- and overshoots for a given sensory input, and produced an average close to the optimal saccade. The trained controller was still erroneous, however, but it could be improved over several stages of learning. At each stage, the already trained controller from the preceding stage was used as a basis for the training of an improved controller (Fig. 4). In this way, a mapping of the fixation error into motor space could be omitted.

3.3.1 Generation of a single learning example A learning example for the saccade controller contains the sensory input and the motor output as described in section 3.2. To generate one learning example, the following steps were carried out:

1. Generate a random starting position (pan_t, $tilt_t$, $verg_t$) for the cameras.
2. Select a target object (goal colors alternate) and determine its coordinates in both salience images ($x_{l,t}$, $x_{r,t}$, $y_{lr,t}$).
3. Generate a new motor command pan_{t+1}, $tilt_{t+1}$, $verg_{t+1}$.
4. Move the cameras to this new position.
5. Re-identify the target object and determine its new coordinates $x_{l,t+1}$, $x_{r,t+1}$, $y_{lr,t+1}$.
6. Include the learning example (pan_t, $tilt_t$, $verg_t$, $x_{l,t}$, $x_{r,t}$, $y_{lr,t} \rightarrow pan_{t+1}$, $tilt_{t+1}$, $verg_{t+1}$) in the training set if it fulfills a certain quality criterion.

In the first stage (without any existing controller), steps 3 and 6 differ from the corresponding steps of the following stages. In step 3, the motor commands pan_{t+1}, $tilt_{t+1}$, $verg_{t+1}$ were generated by adding random noise to pan_t, $tilt_t$, $verg_t$. In step 6, a learning example was included whenever the target object got closer to the center of the salience image on both sides after the saccadic movement.

From the second stage on, the controller C_{k-1} from the preceding stage $k-1$ is available. To generate a new motor command (pan^k_{t+1}, $tilt^k_{t+1}$, $verg^k_{t+1}$), random noise was added to the output (pan^{k-1}_{t+1}, $tilt^{k-1}_{t+1}$, $verg^{k-1}_{t+1}$) of controller C_{k-1}. The range of noise decreased from stage to stage (Schenck and Möller, 2004).

[2] For the control of human eye movements, the efference copy might be more important than extraocular proprioception (Ruskell, 1999; Bridgeman, 1995). Here, we use the term 'kinesthetic' for the eye position, independent of the origin of the information.

At each stage $k > 1$, the quality criterion is based on the Euclidean distances r_L and r_R between the position of the fixation target in the left and right salience image, respectively, and the corresponding image center. The values for r_L and r_R were scaled to the range $[0; 1]$. An overall target distance r was computed as mean value of r_L and r_R. Postsaccadic target distances $r^{\,\mathrm{post}}$ were obtained for 200 trials using controller C_{k-1}. The resulting distance $r^{\,\mathrm{post}}$ was plotted against the presaccadic target distance $r^{\,\mathrm{pre}}$ (Fig. 10). A functional relationship between pre- and postsaccadic target distance was obtained by fitting a quadratic function $Q(r^{\,\mathrm{pre}})$ to the data. A learning example was included in the training set only if the postsaccadic target distance of the example was smaller than $Q(r^{\,\mathrm{pre}})$ for both the left and the right side (Fig. 10).

3.3.2 Network training At each stage, 10 000 learning examples were collected. All input and output parameters were scaled to the range [-1; 1]. The MLP had six input units, 40 hidden units in one layer, and three output units. The activation functions for the respective layers were the identity function, the sigmoidal function, and again the identity function. Training was carried out using 2 000 epochs of resilient propagation (Riedmiller and Braun, 1993). For each stage, the configuration of wooden blocks on the table alternated between two sets. The final controller was obtained after seven stages.

4 Arm controller

The purpose of the arm controller is to find an arm posture that allows grasping of an object. For each processing pathway, a separate arm controller is trained. First, we describe the collection of training data, then the image processing, and finally the learning and recall of a new associative memory.

4.1 Collection of training data

Random exploration was used to collect training data. Initially, the robot arm was in a resting posture, such that it did not occlude the table from the perspective of the cameras. A brick was put in-between the gripper. A sequence to collect one pattern consists of several steps. First, a random position on the table within a 40×30 cm rectangle (the area covered can be seen in Fig. 12) and a random orientation (0° to 360°) were chosen. For a given object position and orientation, up to eight redundant arm postures are possible for grasping. At least two equivalent solutions exist since a 180° turn of the joint near the gripper does not change the appearance of the posture. Thus, the inverse kinematics[3] of the arm provides a set of solutions. From this set, one solution was chosen at random (ambiguous solutions will appear in the final training set due to the density of training patterns). We call the resulting arm posture the 'grasping posture' (Fig. 5 right). In addition to the position on the table, a second one 60 mm directly above was determined together with a corresponding joint-angle set, as before. This is the 'pre-grasping posture' (Fig. 5 left). The grasping and the pre-grasping posture were stored as part of the same pattern.

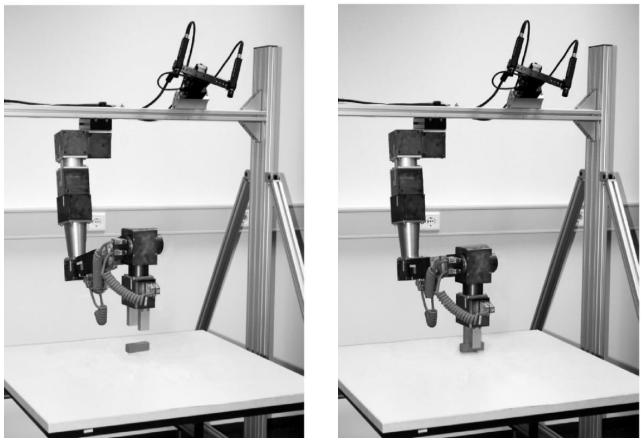

Fig. 5 Pre-grasping and grasping posture

In the next step, the robot put the brick on the table. Between two postures, all six joint angles were transformed simultaneously and linearly. The arm moved via the pre-grasping to the grasping posture. The insertion of the pre-grasping posture avoids collisions with the table and the brick, which could occur if the arm would turn directly from the resting to the grasping position. After the brick was put on the table, the arm moved again to the resting position.

In this posture, visual information of the brick was obtained. Initially, the cameras pointed into a predefined direction. In this state, an image from the left camera was recorded (this was used later for the retinal pathway—see section 4.2). Then, using the trained saccade controller, the cameras were turned such that they fixated the brick. In this position, pan, tilt, and vergence values were stored, and again a picture from the left camera was taken for later processing (for the gaze pathway—see section 4.2). Afterwards, the arm moved through the pre-grasping to the grasping posture to pick up the brick, and then back to its resting posture. This concludes a sequence to collect one pattern. The sequence can be repeated without human intervention. In total, we collected 3371 training and 495 test patterns.

[3] We solved the inverse kinematics analytically, which is a deviation from a pure random exploration. The analytic solution is a technical short-cut avoiding the use of a feedback controller that brings the robot tip onto the table.

4.2 Image processing

Retinal and gaze pathway have different image processing procedures. For the retinal pathway, position and orientation information is extracted. For the gaze pathway only the orientation information is extracted and then combined with the gaze direction.

4.2.1 Position To extract the position of the brick, the original image (here, the image taken while the cameras pointed to a fixed direction—see section 4.1) was first transformed into a gray-scale image, enhancing red $(R-(G+B)/2)$. From this image, a rectangular region was extracted that enclosed all possible locations of the target (Fig. 6). In this region, a population code of the brick's position was obtained by superimposing a grid of 4×4 'neurons' with Gaussian receptive fields (Fig. 6). Their centers covered the image uniformly. The activation of a neuron is the weighted sum over all pixels, with weight factors taken from the corresponding Gaussian function. The resulting 16 activation values were part of one training pattern.

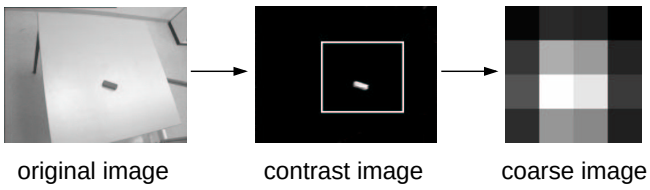

Fig. 6 Pre-processing to obtain the position information. The coarse-grained image was gained from the region inside the white rectangle within the contrast image.

4.2.2 Orientation To obtain the brick's orientation, first, a contrast mechanism enhanced red, resulting in a gray-scale image, as above. Then, the gray-scale image was low-pass filtered. In the next step, four compass filters (Umbaugh, 1998, pp. 65) enhanced the edges in four different directions (0°, 45°, 90°, and 135°). A threshold function was applied to the four resulting images (Fig. 7). The remaining pixels in each image were counted to give a value for the distribution of edges in a given direction. The result is a histogram showing the edge-direction distribution in the gray-scale image. Such a histogram can uniquely encode the orientation of the brick at a given location. The four values of this histogram were part of one training pattern.

4.3 Training patterns

Each training pattern combines visual and postural information. For the retinal pathway, the visual part contained the edge histogram and the activation of the 16

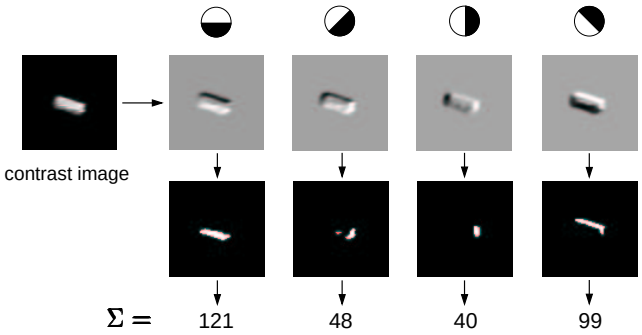

Fig. 7 Image processing to extract information about the brick's orientation. On the very left is the contrast image. In each column, the preprocessing steps for one compass filter (top) are shown, the edge-image, the threshold image, and the sum of white pixels in the threshold image. (Figure adapted from Schenck et al. (2003).)

Gaussian neurons, and the posture was only composed of the 12 joint angles (six for the pre-grasping and six for the grasping posture). For the gaze pathway, the four values from the edge histogram form the visual part, and the postural part consists of the 12 joint angles and the pan, tilt, and vergence values.

All postural variables were encoded by tuning curves; that is, a variable x was represented by the activation of four neurons with Gaussian receptive fields, $\exp(-(x-c_i)^2/(2\sigma^2))$, whose centers c_i were uniformly distributed within the maximal range of the variable. The width σ was chosen to be equal to the distance between two neighboring centers (Fig. 8).

For the retinal pathway, a training pattern was thus 68-dimensional, but 64-dimensional for the gaze pathway. Before training, the patterns were normalized to have unit variance in each dimension.

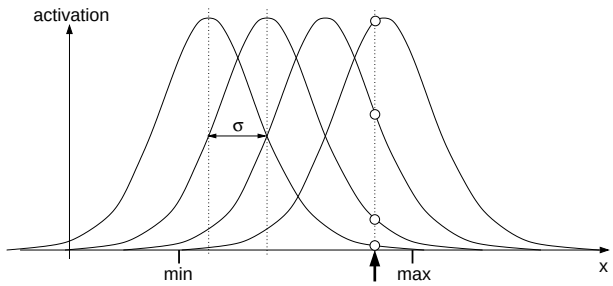

Fig. 8 A population of four broadly tuned neurons encodes the value (thick arrow) of a postural variable x with four values (circles).

4.4 Associative memory

Associative memories complete patterns and can therefore learn one-to-many mappings in contrast to feed-

forward neural networks, which average over all possible solutions and thus fail. Here, we used a new model (Hoffmann and Möller, 2003) that shares the characteristics of a recurrent neural network: patterns are completed by recalling a solution close to the domain of training patterns (Steinkühler and Cruse, 1998).

In the training phase, the distribution of training patterns is described by a collection of ellipsoids. Their axes are aligned with the principal components of the local pattern distribution. We used a neural PCA algorithm (see, for example, Diamantaras and Kung (1996)). To fit the mixture of ellipsoids, Hoffmann and Möller (2003) used an extension of the vector quantizer 'Neural Gas' (Martinetz et al., 1993) to local PCA (Möller and Hoffmann, 2004). In the present study, a different method was chosen that led to slightly better results and was less sensitive to the training parameters.

A mixture of Gaussian functions approximates the density of the data distribution. The Gaussian centers were initialized with the Neural Gas algorithm. Then, the axes were adjusted (and the centers' position fine-tuned) using the 'mixture of probabilistic principal component analysis' algorithm (Tipping and Bishop, 1999). Herein, different from Tipping and Bishop (1999), we extracted the principal components using the 'robust recursive least squares learning algorithm' (Ouyang et al., 2000).

The 'network' parameters are the number of ellipsoids and the number of principal components. For all tests, 120 ellipsoids were used and four principal components extracted. Four principal components were used because, the brick on the table has three degrees of freedom (two for position and one for orientation). Thus, the distribution of patterns has an intrinsic dimensionality of three, and can therefore be described locally by three principal components. An additional principal component was extracted to account for the curvature of the data distribution (this turned out to improve the performance).

In the recall phase, the pattern completion is used as a mapping from an input to an output. The input constrains the recall. Thus, the output space is a subspace (spanned by the basis vectors of the components assigned to the output) whose offset is given by the input. To compute an output (appendix A), the ellipsoid is selected that is closest (smallest normalized Mahalanobis distance) to the constrained subspace. Then, on this constraint, the point that is closest to the chosen ellipsoid gives the desired output values. Figure 9 illustrates this process. In contrast to recall in a true recurrent neural network like the Hopfield network (Hopfield, 1982), in our method no spurious memories exist (appendix A).

This recall algorithm has two advantages over feed-forward networks (Hoffmann and Möller, 2003). First, it does not fail on one-to-many mappings because one valid output is chosen out of many possible ones. Second, the input and output dimensions can be selected after training. Thus, a trained network can (without relearning) operate in different directions. Therefore, in principle, we could also recall image information from an arm posture; in the present study, however, we exploit only the advantage on one-to-many mappings.

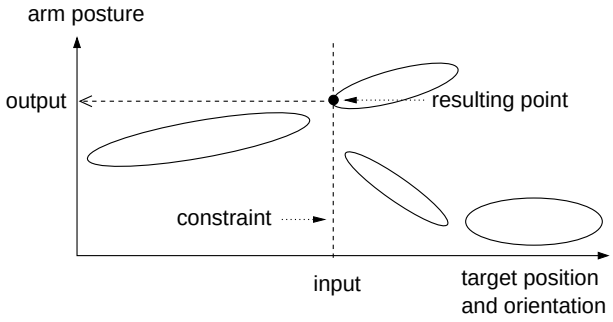

Fig. 9 Pattern completion. The recall is based on a density model of the data (ellipses). The ellipses describe points having the same local density. (Figure adapted from Hoffmann and Möller (2003).)

After recall, the population-coded postural variables were re-transformed into their original form. The result for one variable was obtained by finding the center of a Gaussian that fitted best (minimum sum of squared error) the bell shaped activation over the four neuron centers.

5 Results

First, we show separately the results of saccade and arm controller. These results were gained from a database of real images. Then, we show how these two modules work together.

5.1 Saccade controller

The staged learning of the saccade controller was evaluated by inspecting the collected learning examples, and by measuring the performance of the resulting controller at each stage. The average postsaccadic target distance of all learning examples decreased from 0.38 in the first stage to 0.0 in the seventh stage. In the final stage, all training saccades end at a target (Fig. 10, left column, bottom row). These saccades were 'perfect' in the sense that the precision limit of the image processing did not allow a better performance (the center of mass of the target object was at the center pixel of both the left and right salience image). The number of random movements needed to find one learning example fulfilling the quality criterion increased from 4.2 at the first stage to 19.1 at the seventh stage. Altogether, to obtain one perfect saccade, about 60 random movements were needed during the history of staged learning. In comparison, if searching from scratch, around 60,000 random moves are required to find a perfect learning example.

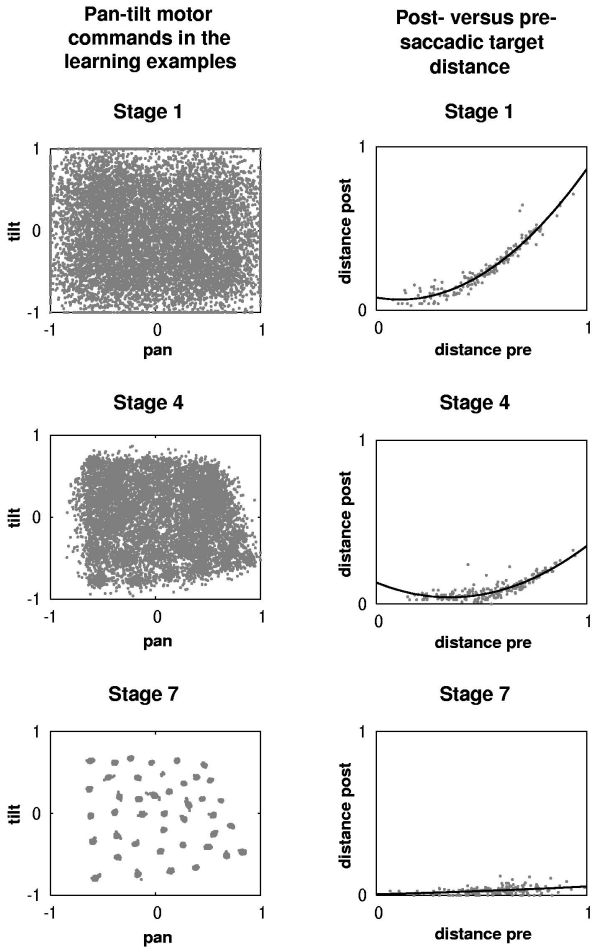

Fig. 10 In the left column, the pan-tilt coordinates of the motor output of all collected learning examples are shown (pan horizontally, tilt vertically). From stage one to seven, the quality of the learning examples improved considerably. In the right column, the overall postsaccadic target distance (y-axis) is plotted against the overall presaccadic target distance (x-axis) for 200 controller movements. A quadratic function (black curve) was fitted to this relationship and was used as the quality criterion in the next stage of training. (Remark: Overall target distance is computed as average of left and right target distance.)

The controller performance was tested on 200 fixation trials. Each fixation trial started from a random camera position with a randomly selected target object. After a series of saccades, a fixation trial was only counted as successful if fewer than 20 saccades were needed. The controller of the first stage was successful in 36% of the trials. Moreover, on average, 11.2 saccades were needed for fixation in a successful trial. In contrast, the controller of the seventh stage was successful in 99% of the trials, with an average number of 3.49 saccades. The reduction of the postsaccadic distance to the target over the stages of training is shown in the right column of Fig. 10. On average, this distance was 0.48 at the first stage, and it decreased to 0.06 after a single saccade at the seventh stage (for more detailed results see Schenck and Möller (2004)).

5.2 Arm controller

The arm controller was evaluated on 495 test patterns (section 4.1). A decision whether a grasp was successful was made using a one-to-one computer model of the gripper and the brick (Fig. 11), whose coordinates and orientation were given in the test set. The position and orientation of the gripper were computed from the joint angles using the arm kinematics. Table 1 displays the result for the mean position error, the mean orientation error, and the success rate for grasping. The position error is the distance between the center of the brick and the center of the two gripper tips.

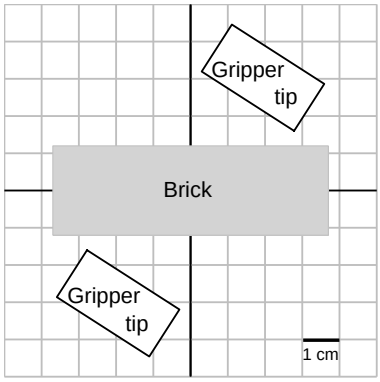

Fig. 11 Model to determine if a grasp was successful. With zero orientation error the position tolerance is ±16 mm, and with zero position error the orientation tolerance is ±41°.

Table 1 Grasping performance

variant	position err. (mm)	orient. err. (degrees)	success (%)
retinal pathway	7.4	3.9	95
gaze pathway	9.3	3.8	94
no tuning curve	20.8	3.6	51
using look-up	15.6	4.9	79
using MLP	241	54	0

The performance is given for the arm controller using the retinal and gaze pathway (first and second row). The following rows show the result of the modifications for the gaze pathway: one model omits tuning curves (third row); one model uses a look-up table (fourth row), and one model uses a multi-layer perceptron (MLP) as network (last row). All error values are averages over 495 trials.

The arm controller could learn the transformation for both retinal and gaze pathway (success rates: 95% and 94%, respectively). The coding of proprioception with

tuning curves (section 4.3) was necessary to achieve a good performance (success rate: 94% compared to 51%). In addition, our associative memory did better than a table look up model (success rate: 94% compared to 79%) which chose out of all training patterns the one that was closest to a given input (Euclidean distance). Thus, the network generalized well. The result was also compared to a variant replacing the associative memory by a feed-forward network. Here, a multi-layer perceptron was used, with one input layer (16 neurons), one hidden layer (20 neurons), and one output layer (48 neurons), comprising the activation functions identity, sigmoid, and identity, respectively. As expected, the feed-forward network could not learn the one-to-many mapping (success rate: 0%).

5.3 Combined model

To test the combined model in the real world, 100 trials were carried out. In each trial, a red brick (as in the training for the arm controller) was placed (by hand) on the table within the rectangle enclosing the training positions. The brick was then fixated, and the arm attempted to grasp it (via the gaze pathway). The same trained arm controller was used, as before.

In all of these trials, fixation with the saccade controller was successful. The mean number of saccades was 2.15. In 54 trials, the brick was fixated with only one saccade. In 94 trials, the brick was grasped without colliding with any of the two gripper fingers. In four trials, the brick slightly collided with one of the fingers (this would be counted as an error in the simulation), but it still could be picked up; in the remaining two trials this was not possible. Figure 12 shows the distribution of successful trials (those with no collision) and errors on the table.

Fig. 12 Grasping performance of the combined model. Successful trials are drawn in gray, error trials in black

6 Discussion

Our robot model demonstrated the ability to fixate a target object with the cameras, to reach for the object and to grasp it with the robot gripper. The model could process the target's position via two pathways: the retinal pathway and the gaze pathway. In the retinal pathway, the object's position is given as a coarse-grained image of the object. This image together with information on the object's orientation is mapped onto an arm posture that is suitable for grasping. In the gaze pathway, first, a saccade controller fixates the target. After fixation, the direction of the cameras together with the target's orientation (as above) are mapped onto an arm posture for grasping.

6.1 Solution to learning problems

We solved two learning problems of general interest: learning without a teacher signal and learning with ambiguous goal values. For saccade learning, to cope with the missing teacher signal in motor space, a learning by averaging procedure was suggested, which was combined with a staged learning mechanism. The missing teacher signal is replaced by selecting training samples based on the quality of their sensory effect. Such a selection in itself would require a prohibitive search in motor space. In combination with staged learning, however, the search is more effective. Stage learning reduced the number of trials needed to acquire a perfect learning example from 60 000 to 60.

In both pathways, the mapping onto the arm posture is one-to-many. Since such a mapping cannot be learned by feed-forward networks, we suggested, as a solution, a new associative memory. This memory is based on a density model of the distribution of sensorimotor patterns. Herein, we did not distinguish between sensory and motor variables. Both are part of the components of a training sample. The density model is a mixture of Gaussians, which can be visualized as a mixture of ellipsoids. A new recall mechanism completes patterns based on this mixture model. From the mixture of ellipsoids, the ellipsoid is chosen that best completes a pattern using a linear mapping. Different ellipsoids can represent different possible solutions. Thus, by choosing *one* ellipsoid, the model copes with ambiguous goal values.

The grasping performance via the retinal pathway was slightly better than for the gaze pathway. However, the saccade controller performed accurately, and we also do not expect any difference in the performance of the associative mapping for the different pathways. Instead, the reason for the higher position error in the retinal case is probably explained by the reduced image resolution (55×55 pixels) used for the saccade controller. Thus, the target's position is less accurately processed.

6.2 Saccade controller compared to literature

In the literature, alternative solutions exist for controller learning without a teacher signal. These solutions are called 'direct inverse modeling' (Kuperstein, 1988), 'feedback-error learning' (Kawato et al., 1987; Kawato, 1990), and 'distal supervised learning' (Jordan and Rumelhart, 1992).

In direct inverse modeling, the sensory consequences of motor commands are first observed. The resulting pairs of sensory and motor commands are then used to train the controller by choosing the sensory values as input (desired goal) and the motor values as output. The disadvantage of this type of learning is that it is not goal-directed, which means that the random search in motor space ignores the goal. Thus, task relevant sensory values may be underrepresented (Jordan and Rumelhart, 1992).

In feedback-error learning, a linear mapping (feedback gain) of the sensory error onto the motor commands can be used to train a non-linear controller. This scheme has been shown to converge to the desired trajectory of motion, but the limitations are that the sign of the feedback gain needs to be known beforehand, and that the initial controller needs to be sufficiently close to the desired solution.

In distal supervised learning, a forward model is learned beforehand. Afterwards, the controller is trained by propagating the error signal back through the forward model. Since the forward model is learned by sampling in the space of motor commands, as in the case of direct inverse modeling, the learning is again not goal-directed. In contrast, staged learning by averaging leads to an accumulation of goal-relevant training samples, because this learning scheme, first, tightens a threshold function around the desired goal, and second, produces new training samples next to goal-relevant samples.

6.3 Arm controller compared to literature

The arm controller could cope with ambiguous goal values, that is, one-to-many mappings. We are aware of two alternative models with this property which are usable for robotics: the 'mean of multiple computations (MMC)' network (Cruse and Steinkühler, 1993; Steinkühler and Cruse, 1998) and the 'parametrized self-organizing map (PSOM)' (Ritter, 1993; Walter et al., 2000).

The MMC network is a recurrent neural network. The weights are given a priori and incorporate the dynamics of the plant (a robot arm, for example). This network can complete patterns by constraining some state values and by letting the network relax to a stable state. However, learning is not considered; the weights have to be specified beforehand.

The PSOM fits a manifold, given as a sum of basis functions, to the distribution of training patterns. Patterns are completed by intersecting a constrained subspace, as in our model, with this manifold. However, PSOM has a disadvantage because it assumes that the patterns lie on a manifold which is connected and sufficiently smooth. In contrast, our model is usable for arbitrary distributions.

6.4 Reaching and grasping in robotics

There exist several studies that present alternative robot models for reaching and grasping. However, we think there are crucial differences to our work. Alternative models differ in at least one of the following three points: first, in many studies, the target's position or its orientation in space is given in world coordinates (Cipolla and Hollinghurst, 1997; Molina-Vilaplana et al., 2004; Salganicoff et al., 1996; Fuentes and Nelson, 1998; Oztop et al., 2004). We avoided these kind of representation because it is not be accessible to an animal either. Second, there are differences in the complexity of the task of our model compared to others since often reaching and grasping are not combined (Ritter et al., 1989; Walter et al., 2000; Molina-Vilaplana et al., 2004; Uno et al., 1995; Fuentes and Nelson, 1998). Only combining reaching and grasping requires knowledge on the target's position *and* orientation. Third, some existing studies either do not deal or cannot deal with the one-to-many-mapping problem. On the one hand, arm controllers that produce incremental changes only deal with one-to-one mappings (Distante et al., 2000; Molina-Vilaplana et al., 2004). On the other hand, feed-forward networks, like Kuperstein's neural controller, cannot map from one to many as a matter of principle (Kuperstein, 1988, 1990).

6.5 Relation to human physiology

In the remainder of the discussion, we show how our model relates to human physiology. The input-output flow of the saccade controller is related to the human oculomotor system in three ways. First, the position of the fixation target is provided in retinal coordinates (salience-image coordinates in our model). For example, the striate cortex and the dorsal layers of the superior colliculus hold similar information about the location of visual stimuli in form of retinotopic maps (Leigh and Zee, 1999).

Second, the motor output is encoded in absolute coordinates. This encoding might also hold for the human saccade control. An incremental update of the eyes' position would often lead to rotations that violate Listing's law (Sparks and Mays, 1990). Accordingly, Crawford and Guitton (1997) showed that a neural implementation of Listing's law requires absolute coordinates (see also Klier and Crawford (1998)).

Third, for the generation of motor output in absolute coordinates, the target position in retinal coordinates is not sufficient. Additionally, the current eye position is necessary, either provided by an efference copy or proprioception. In our model, this information is the kinesthetic input to the saccade controller.

Reaching and grasping were achieved by associating final arm postures, rather than by planning trajectories. This is consistent with the finding that in the monkey, the stimulation of certain motor cortex neurons leads to hand locations independent of the initial arm posture (Graziano et al., 2002). A further study speaks for an association of an object's image with a suitable grasping posture. Rizzolatti and Fadiga (1998) discovered so-called 'canonical neurons' in the premotor-cortex area F5 in monkeys. These neurons fire both when an object is presented only visually and when the same object is grasped by the monkey.

The raw image data is a thin distribution in a high-dimensional space. Therefore, we needed to preprocess the images. The position of an object was extracted by simply blurring the image. Analogue to preprocessing in the brain, blurring is parallel, and neighboring neurons (pixels in the resulting image) respond similar to a given stimulus (Blasdel and Salama, 1986). The orientation of the target object was extracted by applying four different compass filters. These filters extract edges, and act therefore like the simple cells in the primary visual cortex (Hubel and Wiesel, 1962).

For the associative mapping, the joints angles of our robot arm were population coded: four neurons encoded the value of each angle. Population codes are widespread in nature. Tuning curves can be observed, for example, in the monkey for the direction of a moving stimulus (Treue and Trujillo, 1999) and in the cricket for the airflow direction (Miller et al., 1991). Interestingly, in our model, the introduction of tuning curves improved the performance. In biological nervous systems, the purpose of population codes might be to reduce noise (Latham et al., 2003).

Our future work will aim at extending the grasping to manipulation tasks.

7 Acknowledgments

We thank Henryk Milewski for implementing a software collision detection for the robot arm, Janos Kovats for writing server interfaces for the pan-tilt unit and the cameras, and for helping to rebuild the robot, Fiorello Banci for helping to set up the robot, and the anonymous reviewer for his helpful comments on the manuscript.

A The recall mechanism

The goal of the recall is to complete a pattern $\mathbf{p}$ whose components are only partially given (Hoffmann and Möller, 2003). The resulting pattern $\mathbf{z}$ shares the components of $\mathbf{p}$ that are defined as input.

After learning, the data distribution is represented by a collection of hyper-ellipsoids with centers $\mathbf{c}_k$, direction vectors $\mathbf{w}_{ki}$ (principal axes), half-axes lengths $\sqrt{\lambda_{ki}}$, and a residual variance σ_k^2 (in the space orthogonal to the span of the principal axes). The $\mathbf{w}_{ki}$ are the eigenvectors of a local principal components analysis, and the λ_{ki} are the corresponding eigenvalues. The hyper-ellipsoids are iso-potential surfaces of the normalized Mahalanobis distance plus reconstruction error (Hinton et al., 1997),

$$d_k(\mathbf{z}) = \mathbf{y}_k^T \mathbf{\Lambda}_k^{-1} \mathbf{y}_k + (\boldsymbol{\xi}_k^T \boldsymbol{\xi}_k - \mathbf{y}_k^T \mathbf{y}_k)/\sigma_k^2 + \ln\det \mathbf{\Lambda}_k + (n-m)\ln\sigma_k^2 \ . \quad (1)$$

The dimensionality of the pattern space is n, and m is the number of principal components. $\boldsymbol{\xi}_k$ is the distance to the center, $\boldsymbol{\xi}_k = \mathbf{z} - \mathbf{c}_k$. Its representation in the local coordinate system of the ellipsoid is $\mathbf{y}_k = \mathbf{W}_k^T \boldsymbol{\xi}_k$. The eigenvectors $\mathbf{w}_{ki}$ are the columns of $\mathbf{W}_k$. $\mathbf{\Lambda}_k$ is a diagonal matrix containing the eigenvalues λ_{ki}.

An input to the network defines the offset of a constrained space $\mathbf{z}(\boldsymbol{\eta})$ spanning the space of all possible output values:

$$\mathbf{z}(\boldsymbol{\eta}) = \mathbf{M}\boldsymbol{\eta} + \mathbf{p} \ . \quad (2)$$

$\boldsymbol{\eta}$ is a collection of q free parameters (q being the dimension of the network output) in the subspace. $\mathbf{M}$ is a $n \times q$ matrix which is chosen such that the constrained space is in line with the coordinate axes. $\mathbf{p}$ is the offset from zero, which contains the input in its components.

The recall of the complete pattern happens in two steps. First, for each ellipsoid k, determine the point $\hat{\mathbf{z}}_k$ on the constrained subspace with smallest distance (1) to the center $\mathbf{c}_k$. Second, choose the unit k^* resulting in the smallest distance $d_{k^*}(\hat{\mathbf{z}}_{k^*})$. The corresponding $\hat{\mathbf{z}}_{k^*}$ yields the desired output values.

The distance d_k as a function of the free parameters $\boldsymbol{\eta}$ can be written as:

$$\begin{aligned} d_k(\mathbf{z}(\boldsymbol{\eta})) = {} & (\mathbf{M}\boldsymbol{\eta} + \boldsymbol{\pi}_k)^T \\ & \cdot \ (\mathbf{W}_k \mathbf{\Lambda}_k^{-1} \mathbf{W}_k^T + \{\mathbf{I} - \mathbf{W}_k \mathbf{W}_k^T\}/\sigma_k^2) \\ & \cdot \ (\mathbf{M}\boldsymbol{\eta} + \boldsymbol{\pi}_k) \\ & + \ln\det \mathbf{\Lambda}_k + (n-m)\ln\sigma_k^2 \ , \end{aligned} \quad (3)$$

with $\boldsymbol{\pi}_k = \mathbf{p} - \mathbf{c}_k$. We derive with respect to $\boldsymbol{\eta}$:

$$\partial d_k / \partial \boldsymbol{\eta} = 2\,\mathbf{M}^T \mathbf{D}_k \mathbf{M}\boldsymbol{\eta} + 2\,\mathbf{M}^T \mathbf{D}_k \boldsymbol{\pi}_k \quad (4)$$

with

$$\mathbf{D}_k = \mathbf{W}_k \mathbf{\Lambda}_k^{-1} \mathbf{W}_k^T + \{\mathbf{I} - \mathbf{W}_k \mathbf{W}_k^T\}/\sigma_k^2 \ . \quad (5)$$

Setting the derivative equal to zero yields,

$$\hat{\boldsymbol{\eta}}_k = -(\mathbf{M}^T \mathbf{D}_k \mathbf{M})^{-1} \mathbf{M}^T \mathbf{D}_k (\mathbf{p} - \mathbf{c}_k) \ . \quad (6)$$

The function d is convex. Therefore, $\hat{\boldsymbol{\eta}}_k$ is closest to the center. Thus, $\hat{\mathbf{z}}_k = \mathbf{M}\hat{\boldsymbol{\eta}}_k + \mathbf{p}$. For each input, the presented algorithm gives a unique output, which is as close

as possible to the mixture of ellipsoids. In contrast, a recall mechanism using gradient descent along a constraint in a potential field may end in a local minimum that is distant to all training data points.

References

Batista AP, Buneo CA, Snyder LH, Andersen RA (1999) Reach plans in eye-centered coordinates. Science 285:257–260

Blasdel GG, Salama G (1986) Voltage-sensitive dyes reveal a modular organization in monkey striate cortex. Nature 321:579–585

Bridgeman B (1995) A review of the role of efference copy in sensory and oculomotor control systems. Annals of Biomedical Engineering 23:409–422

Buneo CA, Jarvis MR, Batista AP, Andersen RA (2002) Direct visuomotor transformations for reaching. Nature 416:632–636

Carey DP, Coleman RJ, Sala SD (1997) Magnetic misreaching. Cortex 33:639–652

Carey DP, Sala SD, Ietswaart M (2002) Neuropsychological perspectives on eye-hand coordination in visually-guided reaching. In: Progress in Brain Research, Elsevier Science, vol. 140, pp. 311–327

Cipolla R, Hollinghurst N (1997) Visually guided grasping in unstructured environments. Robotics and Autonomous Systems 19:337–346

Crawford JD, Guitton D (1997) Visual-motor transformations required for accurate and kinematically correct saccades. Journal of Neurophysiology 78:1447–1467

Crawford JD, Medendorp WP, Marotta JJ (2004) Spatial transformation for eye-hand coordination. Journal of Neurophysiology 92:10–19

Cruse H, Steinkühler U (1993) Solution of the direct and inverse kinematic problems by a common algorithm based on the mean of multiple computations. Biological Cybernetics 69:345–351

Diamantaras KI, Kung SY (1996) Principal Component Neural Networks. John Wiley & Sons, New York

Distante C, Anglani A, Taurisano F (2000) Target reaching by using visual information and Q-learning controllers. Autonomous Robots 9:41–50

Fuentes O, Nelson RC (1998) Learning dextrous manipulation skills for multifingered robot hands using the evolution strategy. Machine Learning 31:223–237

Graziano MS, Taylor CS, Moore T (2002) Complex movements evoked by microstimulation of precentral cortex. Neuron 34:841–851

Hinton GE, Dayan P, Revow M (1997) Modeling the manifolds of images of handwritten digits. IEEE Transactions on Neural Networks 8(1):65–74

Hoffmann H, Möller R (2003) Unsupervised learning of a kinematic arm model. In: Kaynak O, Alpaydin E, Oja E, Xu L (eds.) Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003, LNCS, Springer, Berlin, vol. 2714, pp. 463–470

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the USA 79:2554–2558

Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology 160:106–154

Jordan MI, Rumelhart DE (1992) Forward models: Supervised learning with a distal teacher. Cognitive Science 16:307–354

Kawato M (1990) Feedback-error-learning neural network for supervised motor learning. In: Eckmiller R (ed.) Advanced Neural Computers, Elsevier, North-Holland, pp. 365–372

Kawato M, Furukawa K, Suzuki R (1987) A hierarchical neural-network model for control and learning of voluntary movement. Biological Cybernetics 57:169–185

Klier EM, Crawford JD (1998) Human oculomotor system accounts for 3-D eye orientation in the visual-motor transformation for saccades. Journal of Neurophysiology 80:2274–2294

Kuperstein M (1988) Neural model of adaptive hand-eye coordination for single postures. Science 239:1308–1311

Kuperstein M (1990) INFANT neural controller for adaptive sensory-motor coordination. Neural Networks 4:131–145

Lacquaniti F, Caminiti R (1998) Visuo-motor transformations for arm reaching. European Journal of Neuroscience 10:195–203

Latham PE, Deneve S, Pouget A (2003) Optimal computation with attractor networks. Journal of Physiology 97:683–694

Leigh RJ, Zee DS (1999) The Neurology of Eye Movements. Oxford University Press: New York, Oxford

Martinetz TM, Berkovich SG, Schulten KJ (1993) "Neural-Gas" network for vector quantization and its application to time-series prediction. IEEE Transactions on Neural Networks 4(4):558–569

Miller JP, Jacobs GA, Theunissen FE (1991) Representation of sensory information in the cricket cercal sensory system. I. Response properties of the primary interneurons. Journal of Neurophysiology 66(5):1680–1689

Molina-Vilaplana J, Pedreño-Molina JL, López-Coronado J (2004) Hyper RBF model for accurate reaching in redundant robotic systems. Neurocomputing 61:495–501

Möller R, Hoffmann H (2004) An extension of neural gas to local PCA. Neurocomputing 62:305–326

Movellan JR, McClelland JL (1993) Learning continuous probability distributions with symmetric diffusion networks. Cognitive Science 17:463–496

Ouyang S, Bao Z, Liao GS (2000) Robust recursive least squares learning algorithm for principal compo-

nent analysis. IEEE Transactions on Neural Networks 11(1):215–221

Oztop E, Bradley NS, Arbib MA (2004) Infant grasp learning: A computational model. Experimental Brain Research 158:480–503

Riedmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, pp. 586–591

Ritter HJ (1993) Parametrized self-organizing maps. In: Gielen S, Kappen B (eds.) Proceedings of the International Conference on Artificial Neural Networks, Springer, Berlin, pp. 568–575

Ritter HJ, Martinetz TM, Schulten KJ (1989) Topology-conserving maps for learning visuo-motor-coordination. Neural Networks 2:159–168

Rizzolatti G, Fadiga L (1998) Grasping objects and grasping action meanings: The dual role of monkey rostroventral premotor cortex (area F5). Novartis Foundation Symposium 218:81–103

Ruskell GL (1999) Extraocular muscle proprioceptors and proprioception. Progress in Retinal and Eye Research 18:269–291

Salganicoff M, Ungar LH, Bajcsy R (1996) Active learning for vision-based robot grasping. Machine Learning 23:251–278

Schenck W, Möller R (2004) Staged learning of saccadic eye movements with a robot camera head. In: Bowman H, Labiouse C (eds.) Connectionist Models of Cognition and Perception II, World Scientific, London, NJ, pp. 82–91

Schenck W, Hoffmann H, Möller R (2003) Learning internal models for eye-hand coordination in reaching and grasping. In: Proceedings of the European Cognitive Science Conference, Erlbaum, Mahwah, NJ, pp. 289–294

Snyder LH (2000) Coordinate transformations for eye and arm movements in the brain. Current Opinion in Neurobiology 10:747–754

Snyder LH, Batista AP, Andersen RA (2000) Saccade-related activity in the parietal reach region. Journal of Neurophysiology 83:1099–1102

Sparks DL, Mays LE (1990) Signal transformations required for the generation of saccadic eye movements. Annual Review of Neuroscience 13:309–336

Steinkühler U, Cruse H (1998) A holistic model for an internal representation to control the movement of a manipulator with redundant degrees of freedom. Biological Cybernetics 79:457–466

Tipping ME, Bishop CM (1999) Mixtures of probabilistic principal component analyzers. Neural Computation 11:443–482

Treue S, Trujillo JCM (1999) Feature-based attention influences motion processing gain in macaque visual cortex. Nature 399:575–579

Umbaugh SE (1998) Computer Vision and Image Processing: A Practical Approach Using CVIPtools. Prentice Hall, Upper Saddle River, NJ

Uno Y, Fukumura N, Suzuki R, Kawato M (1995) A computational model for recognizing objects and planning hand shapes in grasping movements. Neural Networks 8:839–851

Walter JA, Nölker C, Ritter H (2000) The PSOM algorithm and applications. In: Proceedings of the Symposium on Neural Computation, pp. 758–764

Wolpert DM, Ghahramani Z, Jordan MI (1995) Are arm trajectories planned in kinematic or dynamic coordinates? An adaptation study. Experimental Brain Research 103:460–470

Zipser D, Andersen RA (1988) A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature 331:679–684