Learning Internal Models for Eye-Hand Coordination in Reaching and Grasping

Wolfram Schenck, Heiko Hoffmann, Ralf Möller
(schenck,hoffmann,moeller@psy.mpg.de)
Cognitive Robotics, Max Planck Institute for Psychological Research; Amalienstr. 33
80799 Munich, Germany

Abstract

A computational model of sensorimotor transformations underlying eye-hand coordination in reaching and grasping is presented and tested in a robot-arm setup. We suggest solutions for both the problem of the missing teacher signal and for the problem of one-to-many mappings encountered when learning inverse models and controllers.

Introduction

Moving the eyes such that the image of an object appears in both foveae, extending the arm towards the foveated object, and finally grasping it with the appropriate hand shape and orientation requires a complex transformation from the sensory (vision, proprioception) to the motor domain (muscles of eye, arm, and hand). The questions which neural mechanisms are underlying such sensorimotor transformations and in which brain structures they are implemented are intensively studied in neurophysiological experiments. There is abundant evidence for a strong participation of parietal cortical areas in this task, which seem to be involved in spatial attention processes and object shape recognition (Gottlieb, 2002), in the transformation from retinal to hand-centered coordinates (Buneo et al., 2002), and generally in the integration of eye and arm movements (Carey, 2000). Here we present a computational model that addresses the question how the sensorimotor transformations for eye-hand coordination in reaching and grasping can be learned. Sensorimotor transformations are a special case of the more general notion of "internal models" (Sabes, 2000), computational functions of the brain that establish relations between sensory signals of different modalities and between sensory and motor signals. Internal models are "trained" by collecting examples of such relationships, for example by observing the effects that self-generated actions have on the sensory inputs. Artificial neural networks are often used as an abstract computational description of the capability of internal models to learn generalized relationships from examples (Jordan and Rumelhart, 1992).

Internal models that integrate sensory and motor information are classified as "inverse models" (with the special case of "feedforward controllers") which provide the motor commands required to achieve some sensory goal, and "forward models" which predict the sensory consequences of movements (Jordan and Rumelhart, 1992; Sabes, 2000). Here we focus on learning of inverse models and controllers, i.e. sensorimotor transformations from the current sensory state and a given goal to the motor command required to accomplish the goal (in the special case of a controller, the goal is specified *implicitly*). The training of inverse models and controllers is burdened with two problems. Firstly, without an external teacher, the motor signal required to achieve a goal in a given situation is unknown, so that the training examples can only be collected by some kind of directed search in the sensorimotor space. Secondly, sensorimotor relationships usually are "one-to-many mappings": for the same current state, different motor commands can be appropriate to achieve the same goal, an example being a manipulator with redundant degrees of freedom. Solutions for both problems are presented in this paper.

Overall system architecture

Figure 1 shows the overall architecture of the learning system. It comprises two internal models encoding the sensorimotor relationships of a 6-degree-of-freedom robot arm with two-finger gripper and a 2-degree-of-freedom pan-tilt unit with stereo camera system (figure 6). The task of the system is to foveate and grasp objects lying on a table. Since the objects have an elongated block shape, they can only be grasped in certain gripper orientations. *Input* to the learning system are two images from the stereo camera system (top left) and proprioceptive information describing the current gaze angles of the pan-tilt unit in combination with a software vergence angle. As *output*, the system generates angular changes for pan, tilt, and vergence, as well as a joint angle vector describing the arm posture (including gripper orientation). The internal model has two parts. The first part is a *foveation controller*. According to a goal category, it selects a salient object for foveation and produces the appropriate eye movement. The second part, the *eye-arm model*, receives the image of the foveated object and the corresponding pro-

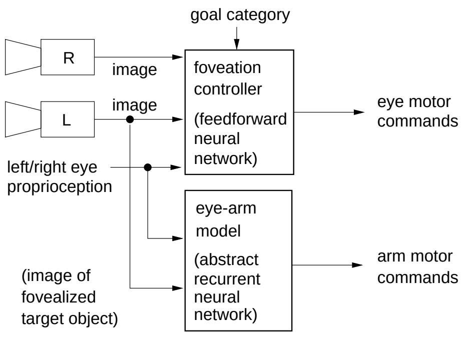

goal category
R
L
image
image
foveation controller
(feedforward neural network)
eye motor commands
left/right eye proprioception
eye-arm model
(abstract recurrent neural network)
arm motor commands
(image of fovealized target object)

Figure 1: Overall system architecture.

prioceptive signals from the vision system, and produces a sequence of two arm and gripper postures to grasp the object.

The goal category (presently a color information) is used as a simple means to model task-specific gating signals for feature selectivity which are supposed to influence neural activity in the parietal cortex (Gottlieb, 2002). Following neurophysiological evidence that saccades are preceding and guiding arm movements, and that limb movements, at least in the early processing stages, are described in eye-centered coordinates (Carey, 2000), the input to the eye-arm model is the result of the saccade sequence generated in the loop of the foveation controller. This also entails a staged training procedure where the foveation controller is trained before the eye-arm model.

Reaching and grasping are not treated as separate processes in our controller, taking into account recent results that indicate a closer relation between the two phases and thus contradict the older hypothesis of separated "visuomotor channels" (Jeannerod, 1999). The eye-arm model includes a simple mechanism of shape pre-processing, which is in accordance with the fact that neurons in some parts of the parietal cortex are sensitive to two- or three-dimensional shape parameters (Gottlieb, 2002).

Foveation controller

The goal of foveation is to fixate a chosen target within two saccades (on average), a performance similar to that of humans (Deubel et al., 1996). The core of the *foveation controller* is implemented as a *feedforward neural network*. It receives two inputs: Kinesthetic information about the current pan, tilt, and vergence position, and the coordinates of the selected foveation target in the left and right input image (see figure 2). These input images are regions in the left and right camera image which are chosen according to the current vergence position (the resulting input images are shown in the left column in figure 3). In this way, vergence movements are simulated in software, while pan and tilt movements are carried out through the hardware. When foveation

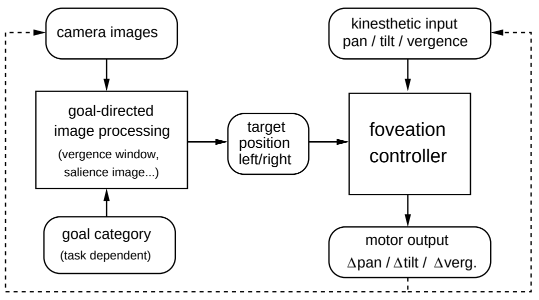
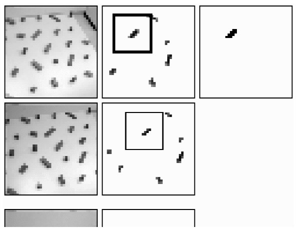

camera images
kinesthetic input pan / tilt / vergence
goal-directed image processing
(vergence window, salience image...)
target position left/right
foveation controller
goal category
(task dependent)
motor output
Δpan / Δtilt / Δverg.

Figure 2: Architecture of the foveation controller.



region surrounding the target object (bold arrow between upper left and right salience image in figure 3). After a saccade, the foveated target has to be re-identified in the images. For this purpose, the same correlation approach is applied. There is evidence that humans recover visual stability after a saccade also through re-identification of the foveation target (Deubel et al., 1996).

By this method, the coordinates of the target object in the left and right input image are computed as input for the foveation controller. Since the height values are nearly the same for the left and right "eye" due to the geometry of the setup, these values are combined in only one input parameter. Thus, together with the three kinesthetic input parameters, there are six input values to the foveation controller. The output of the foveation controller comprises a change value for each of the three motor parameters pan, tilt, and vergence (see figure 2).

Since the controller is implemented by an adaptive multi-layer perceptron, one of the most important issues concerns network training and the collection of good learning samples. The system has no means to determine the optimum motor output for computing error signals so that straightforward supervised learning is not applicable. Thus, instead of building up a set of perfect learning samples, the system starts with random movements and includes every movement in the training set as long as it fulfills a minimum quality criterion, namely, a shift of the target position towards the center in both the left and right input image. In doing so, optimum vergence is determined for each movement since it can be computed easily from the camera images. The solution offered here for the above-mentioned (first) problem of the missing teacher signal exploits the averaging property of feedforward neural networks. While the training set will rarely include perfect examples for saccades, it will include some examples in which the saccade undershoots, and other examples in which it overshoots the fovea. Since a feedforward neural network is forced to average such examples if the input information is similar, it will produce a close-to-optimal saccade. At the moment, training is restricted to target objects on a table surface (see figure 6 for a picture of the robot-arm setup).

The foveation controller used for the combined grasping system presented in this paper is a multi-layer perceptron with one hidden layer of 30 units. In around 350.000 random movements, a training set of 40.000 samples was collected. 12.000 epochs of training were carried out using resilient propagation (Riedmiller and Braun, 1993) which shows a much better performance than standard back-propagation.

Eye-arm model

The *eye-arm model* is implemented as an abstract *recurrent neural network* (RNN). Unlike feedforward neural networks — which are "function approximators" and thus only can learn one-to-one or many-to-one, but not one-to-many mappings (Jordan and Rumelhart, 1992) —, recurrent neural networks are pattern storages which can encode arbitrary data manifolds and thus also reproduce one-to-many mappings (Cruse, 2003). In our setup, an elongated block-shaped object (a 74 mm long red brick, see figure 4) can at least be grasped in two gripper orientations; in addition, even though our 6-degree-of-freedom manipulator is not "redundant" with respect to end-effector position and orientation, multiple (up to 8) discrete postures bring the end-effector into the same position and orientation. We show that the RNN can actually solve this (second) problem of learning one-to-many mappings in inverse models. Similar approaches were suggested by Steinkühler and Cruse (1998) and Walter (1998). However, these studies rely on knowledge about the structure of the training data, while we aim for a general unsupervised learning method.

We assume that recurrent networks could be employed in the parietal cortex to bi-directionally associate visual and motor information; neurons that respond to both manipulation of an object in the dark and to the fixation of the same object can be interpreted as positive evidence (Sakata et al., 1995).

The abstract RNN is implemented by a combination of a density model, describing the structure of the training data, and a recall mechanism working solely on the density model.

A Gaussian mixture model (a sum of Gaussian functions) is used to approximate the density distribution of the training pattern vectors. For fitting the Gaussian functions, we first set their centers using the vector quantization method Neural Gas (Martinetz et al., 1993). Then, we used the algorithm "mixture of probabilistic principal component analyzers" (PPCA) (Tipping and Bishop, 1999) to adjust the shape of the Gaussians to the local distribution of the data. Here, we used a mixture of 160 Gaussians (modules). The PPCA determines 3 eigenvectors for each module.

The resulting Gaussians can be regarded as a set of hyper-ellipses (with axis directions = eigenvectors and squared axis lengths = eigenvalues as calculated by the local PPCA) in the sensorimotor space that "enclose" all training examples and thus represent the manifold of the sensorimotor relationship in a generalized way.

Recall starts by defining a hyper-plane spanning over all output dimensions with an offset in the input dimensions set to the input values. In this hyper-plane a point is chosen which has the smallest normalized Mahalanobis distances to an ellipsoid center. This point is thought to be closest to the described data manifold. Its position in the output dimensions defines the output uniquely. Figure 5 illustrates this process, equations can be found in Hoffmann and

Möller (2003).

Here, the abstract RNN is only used in one direction of recall (as "inverse model"): image and proprioceptive signals are "clamped" to the values coming from the eyes, while the arm motor signals are "free", so that our "network" can find the appropriate motor commands for the given input.

Training data are collected with a combination of random exploration and guided movement. The goal is to collect images of the unobstructed brick together with corresponding arm postures allowing to grasp the brick. A training cycle starts with randomly choosing a position on the table, and an orientation of the gripper (between 0° and 360°). In the grasping position the gripper is always facing downward. Given position and orientation of the gripper, the corresponding arm angles are obtained as the solution of the inverse kinematics (this is the guided movement). Since there exist up to 8 solutions, one solution is selected at random. For each gripper position on the table, a second one 6 cm above is chosen as "pre-grasping" posture. A training sequence consists of moving from a home-position (allowing unobstructed view onto the table) to the pre-grasping position, from there to the grasping position, releasing the brick (initially placed in the gripper), moving back to the pre-grasping position, from there to the home-position, and then recording the left camera image of the brick (to do so the camera was always facing in the same direction). The sequence is repeated to pick up the brick again, and to get back into the initial state. Every movement from one position to another is performed by linearly and simultaneously changing all joint angles. Sequences that would result in a collision are skipped. Apart from the image, the joint angles of the pre-grasping and grasping position (12 angles in total) are stored. Furthermore, the pan and tilt angle required for foveating the brick are computed and stored. To save time in collecting data, for each brick position and orientation, a second solution (different from the first) of the inverse kinematics is chosen (again randomly) without executing it with the robot (since it is not necessary to replace the brick), and stored as a training pattern. Thus, the data clearly forms a one-to-many mapping. Totally, 4069 patterns were recorded.

Joint angles, pan-tilt values, and image are preprocessed, and the result is stored in a single 60-dimensional pattern vector. Each of the 12 angles, pan, and tilt were coded topologically by four Gaussian neurons, which have receptor fields equally spaced in the value range of the variable.

Shape information is extracted from the left camera image (see figure 4). A red filter and a binomial filter are applied. Edges in four different orientations are extracted by a set of compass filters. Active edge pixels are selected by a thesholding operation. A summation over these pixels gives a value

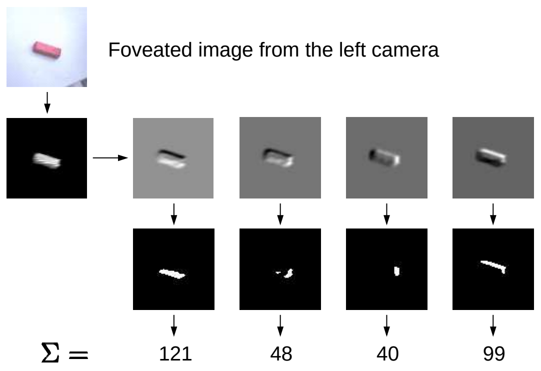


Figure 4: Preprocessing of foveated image to extract shape information.

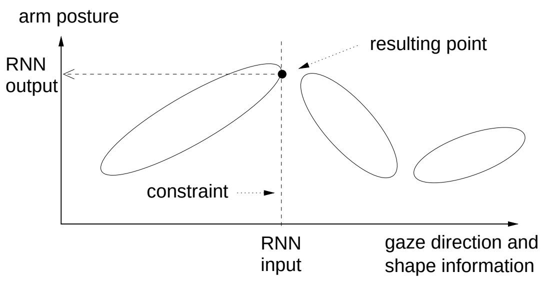


Figure 5: Recall in the abstract recurrent neural network. Ellipses describe points having the same normalized Mahalanobis distance to their respective centers.

for the orientation. The variance in each dimension was normalized to 1.

In connection with the foveation controller, the input is gained from left/right eye proprioception and from the foveated image after successful foveation. The proprioceptive input (pan and tilt) and the image are preprocessed in the same way as described above. The output of the network are the topologically coded joint-angles. These are transformed back analytically by fitting a Gaussian function to the neurons' output, and we obtain 6 angles for the pre-grasping and 6 angles for the grasping posture. The brick is grasped by moving from the home-position to the pre-grasping, and from there to the grasping position in the same way as described for training.

Results

Before being combined, the foveation controller and the eye-arm model were tested separately.

The foveation controller was tested using the same setup as in training. More than 30 objects were distributed over the table surface (800×800 mm). Starting from a random position with a randomly selected target object, is was determined how many saccades were needed until the center of the target

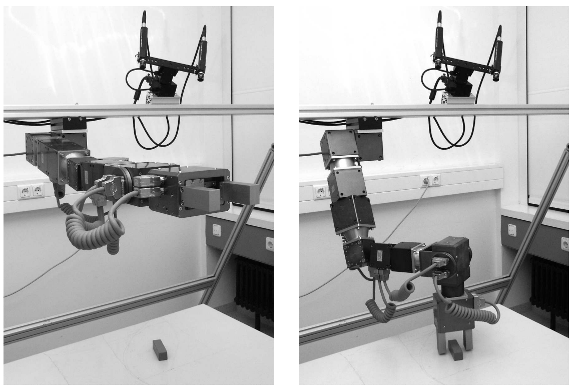


Figure 6: Grasping the brick: after foveation (left image), after moving to grasping posture (right image).

object was located in both input images within the center. If this center was defined as a 1%-region of the overall image, 99 from 100 trials succeeded. 2.05 saccades were performed in average (SD: 1.42). A center region defined as 0.25% of the overall image results in a considerably worse performance: 69 of 100 trials succeeded, the average number of needed saccades was 3.61 (SD: 2.19).

The eye-arm model was tested with the help of a test pattern file which was generated the same way as the training pattern file. It contained 200 patterns. The mean absolute position error for grasping was (3.5, 4.2) mm on the table and 1.6 mm vertical. For comparison, the grasping positions cover a rectangle 250×400 mm. The mean absolute orientation error was 2.9°. This performance is better then the one gained by using the whole training pattern set as a look-up table. In that case, the horizontal position error was (6.5, 7.5) mm, and the orientation error 3.7°. In addition, grasping performance was analyzed with a geometric model of brick and arm. In 92.5% of all tests, the brick could be grasped. This demonstrates the successful application of recurrent neural networks to solve the one-to-many mapping problem.

The combined model was tested by 10 times placing the red brick at arbitrary positions and with arbitrary orientations on the table, within an accessible region. Figure 6 illustrates one trial. In nine out of the 10 trails foveation was successful. On average, 2.9 saccades were needed to match the foveation criterion (here 0.25%, since precise proprioceptive information was needed as input of the eye-arm model). In 6 trials the brick was grasped successfully.

Discussion

We presented a computational model which exhibits some of the visuomotor competences attributed to the parietal cortex. We suggest solutions for two problems encountered when learning inverse models and controllers. As solution for the first problem — the lack of a teacher signal — we suggest to exploit the averaging property of feedforward neural networks and demonstrate the principle for the example of a foveation controller. As solution of the second problem, we presented a general learning method for one-to-many mappings, here shown for a grasping task.

The foveation approach is based on the idea of generalizing over an easily available set of imperfect samples. Compared to other approaches from the field of inverse model acquisition, this procedure seems to be better suited for controller training. For example, Jordan and Rumelhart (1992) propose “distal supervised learning” and “direct inverse modeling”. Both approaches suffer from the fact that the whole sensorimotor parameter space has to be sampled intensively to find some examples related to the controller task. In our approach, all samples which fulfill a minimum quality criterion can be included in the training set and used for training. “Feedback error learning” (Bruske et al., 1997; Kawato, 1999) does not offer a general solution for controller training either, because it relies on the existence of a suitable feedback controller. For foveation movements, such a controller can easily be specified, but for more complex movements, this is difficult and biologically implausible. The work of Kuperstein (1991) is also restricted: Learning samples were collected by directly producing the desired sensory effect. For foveation movements, such an approach is not applicable.

On the other hand, the generalization approach used for learning pan and tilt movements relies on some prerequisites. First, the larger the dimensionality of the sensorimotor space, the more difficult is the acquisition of a sufficient number of learning samples. This restriction holds for most of the aforementioned approaches as well. Second, it has to be guaranteed that averaging over data points in output space actually produces the optimum output. This is only the case when learning samples with similar input values are distributed in output space in a way that the average of the output vectors equals the optimum for the task at hand. For the proposed foveation controller, this is approximately true. There is only one optimal saccade for a given target. Moreover, the mapping between controller input space (including target coordinates) and pan-tilt change space is almost linear. In the set of selected learning samples for a given input, all patterns are distributed symmetrically in target coordinate space around the image centers. Due to the linearity, these patterns are also almost symmetrically distributed in pan-tilt change space. The center of this distribution is close to the optimal saccade. Thus, averaging of output data in the multi-layer perceptron will result in optimal performance. However, as the results with the 0.25% criterion show, the current training data do not allow a precise foveation for the whole table. Because of this, the accessible region for the tests of the combined model was limited by both the arm geometry and the table area where precise foveation is possible with the current controller network.

For training of the eye-arm model, the proprioceptive information of the eyes was generated analytically. Alternatively, we could use the pre-trained foveation controller to produce these values. So far, for training of the eye-arm model, we took all the brick images using the same camera orientation. This could be improved by using the foveation controller, because turning the cameras alters the shape information in the images. Here, the effect was only small, a maximum deviation of about 10 degrees of the brick orientation could be observed for the region of operation of the combined model.

Future work

In the next steps of this work, we will strive to overcome the limitations of the recent implementation. We will attempt to improve the foveation controller by an iterative training procedure where training examples are generated by adding noise to the motor command produced by the controller of the previous iteration step. So far, the eye-arm controller only uses a single image of an object which is sufficient for the simple block objects used but will have to be extended to a stereo image for more complex object shapes. Similarly, the simple pre-processing method of the eye-arm model has to be extended. In the data collection, arm postures are currently generated based on an (analytical) inverse kinematic model, which will be eliminated.

We see the inverse models described in this work as a stepping stone towards more complex manipulation tasks. Similar to Cruse (2003), our ultimate goal is to demonstrate that cognitive abilities — specifically an understanding of the shape and the physical properties of objects — have their origin in basic sensorimotor capabilities, an approach contrasting with the classical "cognitivist" paradigm. Extensions of this work will therefore also include "forward models" that allow a prediction of sensorimotor consequences and thus, an interpretation of a visual scene without overt motor responses.

References

Bruske, J., Hansen, M., Riehn, L., and Sommer, G. (1997). Biologically inspired calibration-free adaptive saccade control of a binocular camera-head. *Biological Cybernetics*, 77:433–446.

Buneo, C. A., Jarvis, M. R., Batista, A. P., and Andersen, R. A. (2002). Direct visuomotor transformations for reaching. *Nature*, 416:632–636.

Carey, D. P. (2000). Eye-hand coordination: Eye to hand or hand to eye? *Current Biology*, 10:R416–R419.

Cruse, H. (2003). The evolution of cognition — a hypothesis. *Cognitive Science*, 27:135–155.

Deubel, H., Schneider, W. X., and Bridgeman, B. (1996). Postsaccadic target blanking prevents saccadic suppression of image displacement. *Vision Research*, 36:985–996.

Gottlieb, J. (2002). Parietal mechanisms of target representation. *Current Opinion in Neurobiology*, 12:134–140.

Hoffmann, H. and Möller, R. (2003). Unsupervised learning of a kinematic arm model. In *ICANN/ICONIP 2003, to appear in Lecture Notes in Computer Science*. Springer.

Jeannerod, M. (1999). Visuomotor channels: Their integration in goal-directed prehension. *Human Movement Science*, 18:201–218.

Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9:718–727.

Kuperstein, M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks*, 4:131–145.

Martinetz, T. M., Berkovich, S. G., and Schulten, K. J. (1993). "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *International Conference on Neural Networks*, pages 586–591.

Sabes, P. N. (2000). The planning and control of reaching movements. *Current Opinion in Neurobiology*, 10:740–746.

Sakata, H., Taira, M., Murata, A., and Mine, S. (1995). Neural mechanisms of visual guidance of hand action in the parietal cortex of the monkey. *Cerebral Cortex*, 5(5):429–438.

Steinkühler, U. and 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, M. E. and Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482.

Walter, J. A. (1998). PSOM network: Learning with few examples. In *Proc. Int. Conf. on Robotics and Automation*, pages 2054–2059. IEEE.